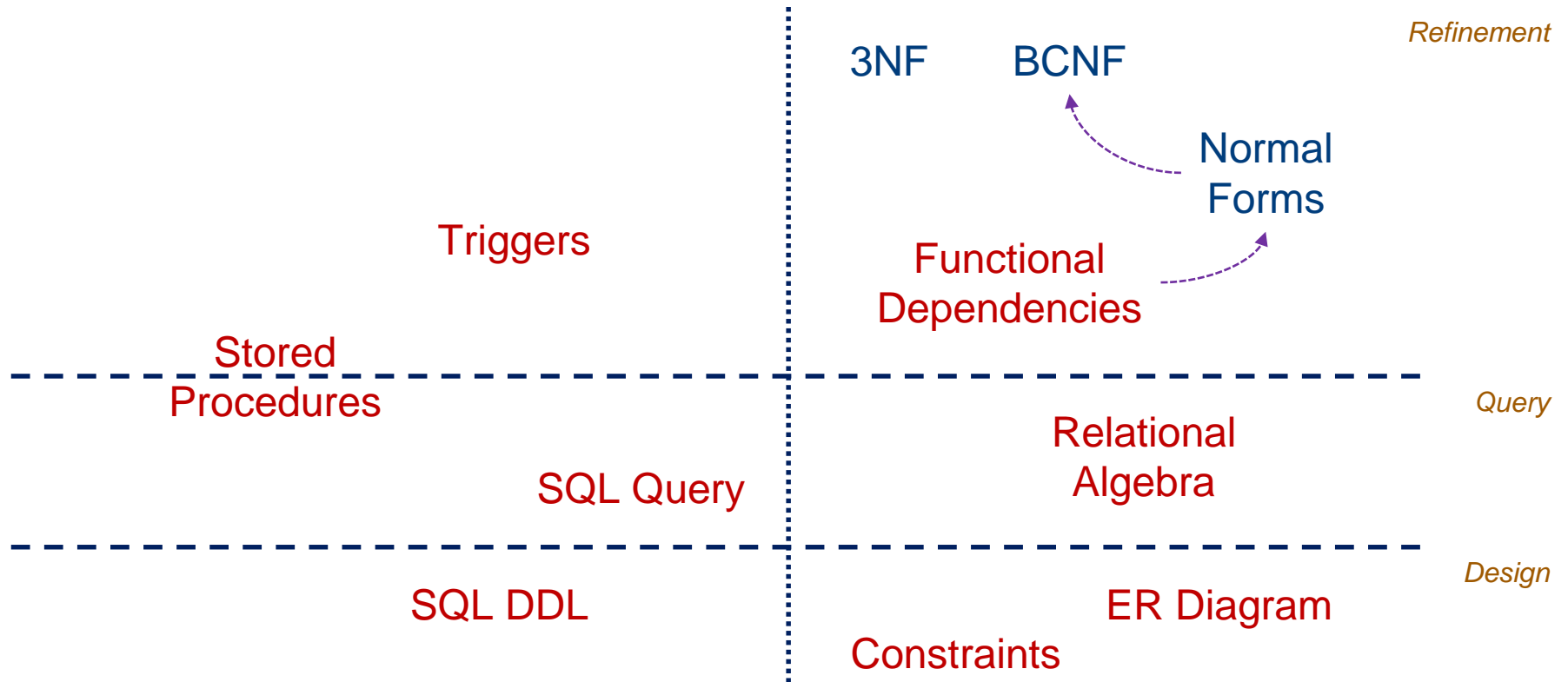


# **CS2102 Database Systems**

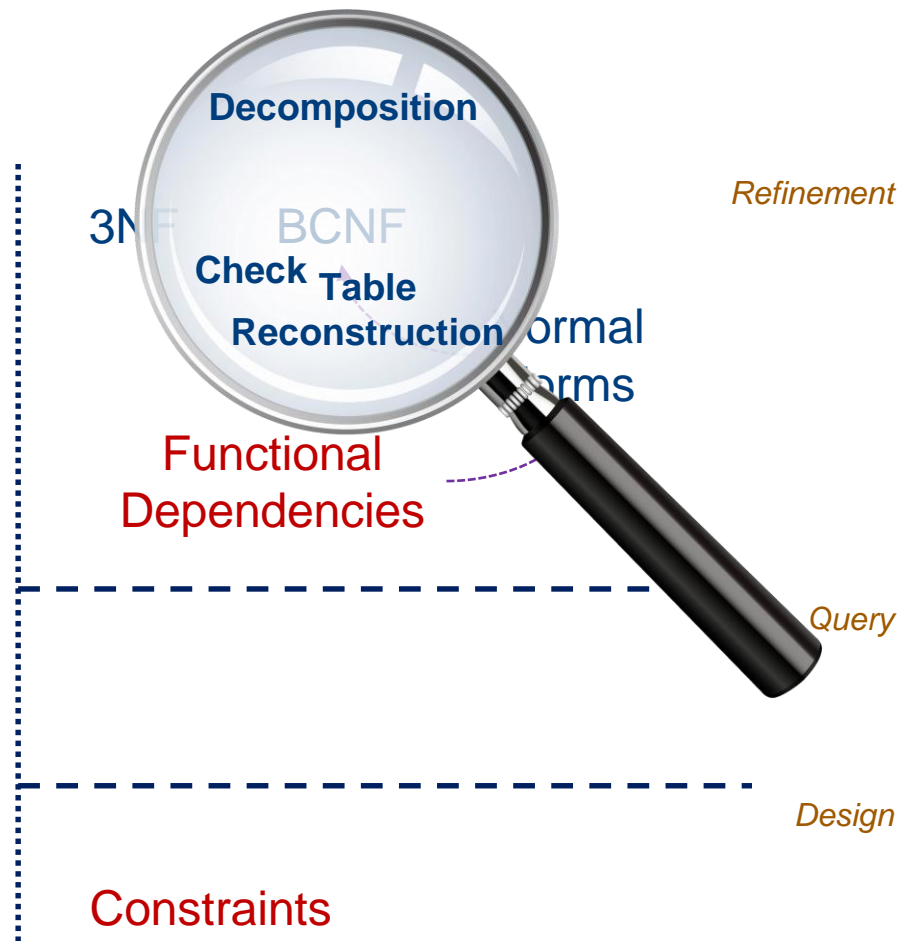
## Lecture 11 – Boyce-Codd Normal Form

# Roadmap

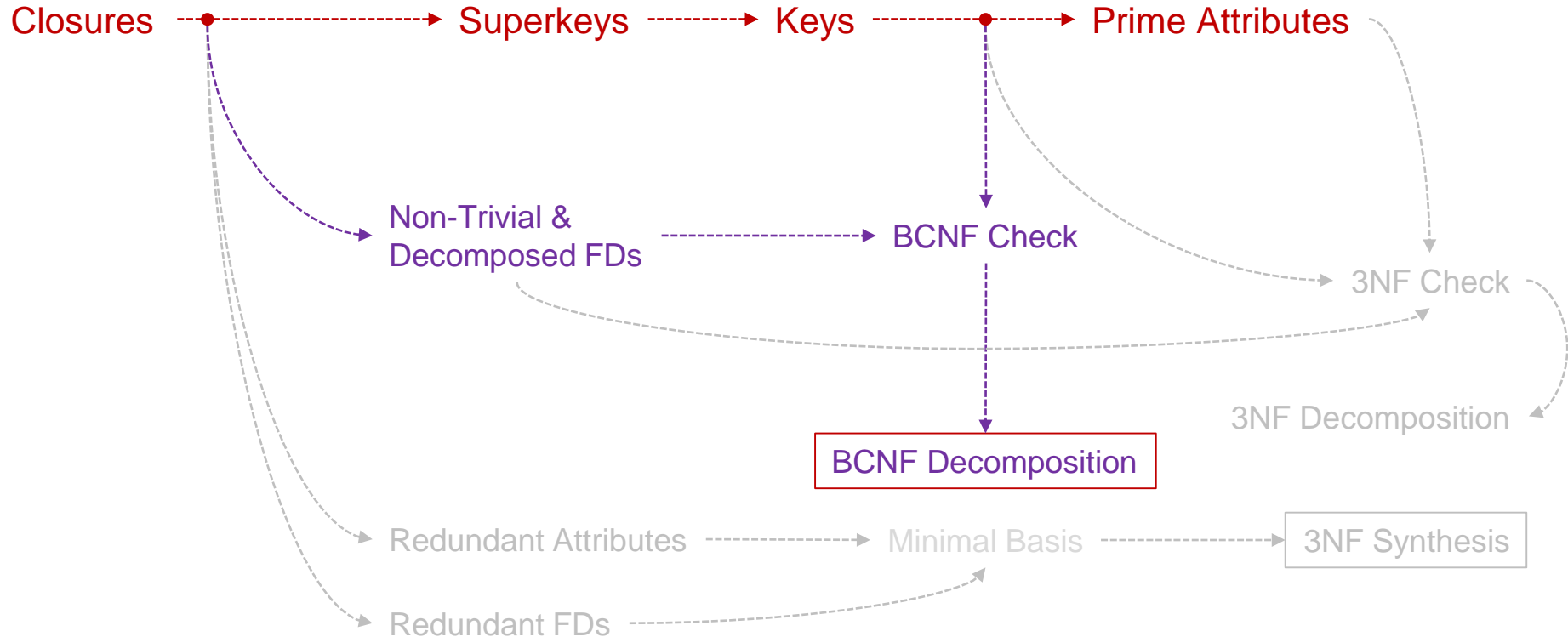


# Roadmap

- We will do this step by step
  - Recap FD *(with non-trivial & decomposed FD)*  
↓
  - Define BCNF  
↓
  - Check BCNF  
↓
  - Normalize to BCNF
  - ❖ *Show that we can reconstruct the original table*



# Algorithm Roadmap



# Normal Form(s)

- What is it?
  - Conditions that a “good” table should satisfy
  - Various NFs in increasing order of strictness

Easy to satisfy.  
May have high  
redundancy.

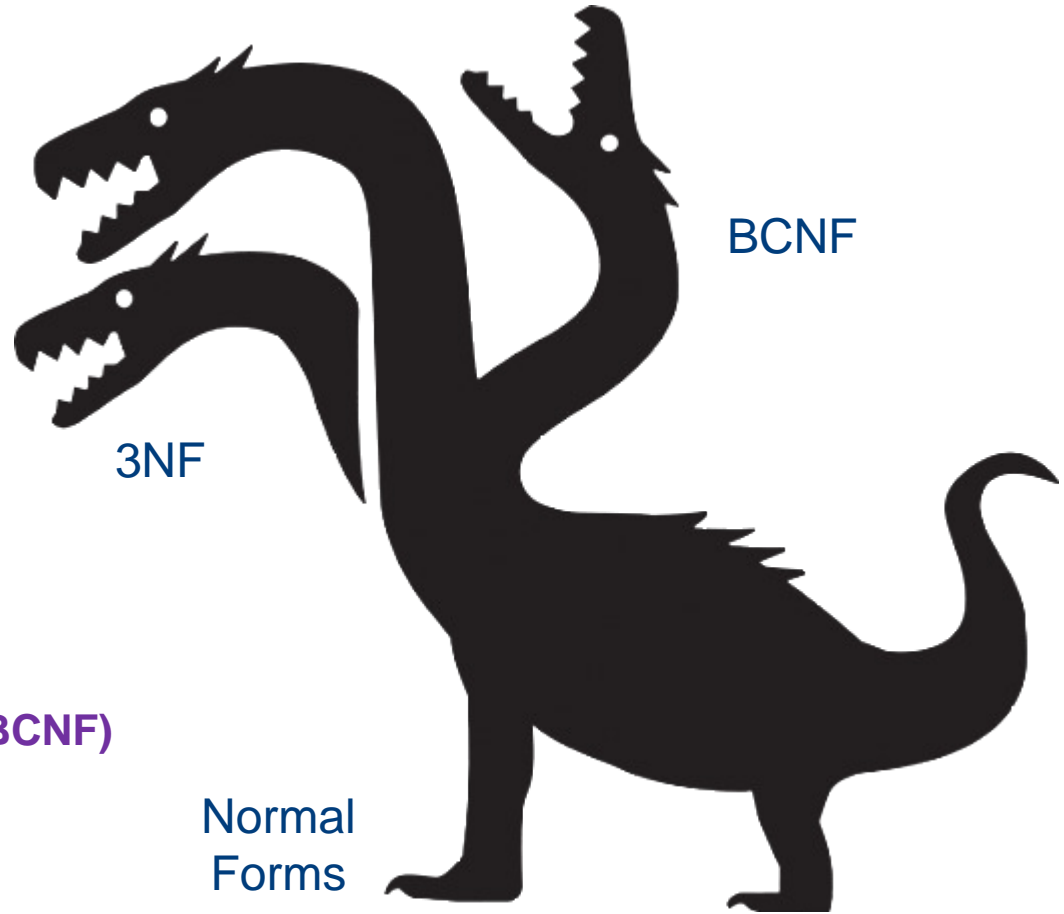
1. 1<sup>st</sup> NF
2. 2<sup>nd</sup> NF

3. **3<sup>rd</sup> NF (3NF)**

4. **Boyce-Codd NF (BCNF)**

Very little  
redundancy.  
Not always  
possible to  
satisfy.

5. 4<sup>th</sup> NF
6. 5<sup>th</sup> NF
7. 6<sup>th</sup> NF



# RECAP: FD

non-trivial and decomposed functional dependencies



# Non-Trivial and Decomposed FD

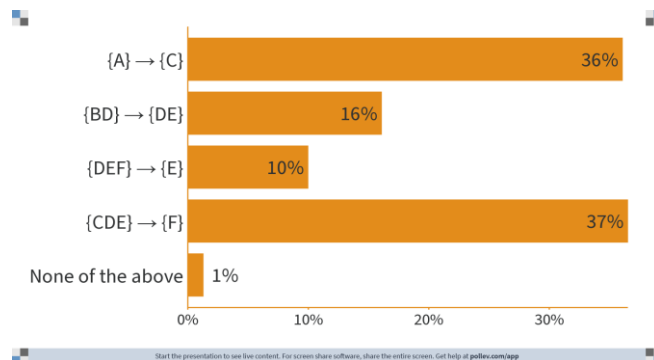
- **Simplifying Discussion**

- **Non-trivial**  $\{A\} \rightarrow \{B\}$  where  $\{B\} \not\subseteq \{A\}$  *(literally, not trivial)*
- **Decomposed**  $\{A\} \rightarrow \{B\}$  where B is a single attribute *(singular)*
- ❖ A non-decomposed FD can always be transformed into the equivalent set of decomposed FD by *decomposition rule*
  - $\{BC\} \rightarrow \{DE\} \Leftrightarrow \{BC\} \rightarrow \{D\}$  and  $\{BC\} \rightarrow \{E\}$

# Non-Trivial and Decomposed FD

- Simplifying Discussion

- **Non-trivial**  $\{A\} \rightarrow \{B\}$  where  $\{B\} \not\subseteq \{A\}$  *(literally, not trivial)*
- **Decomposed**  $\{A\} \rightarrow \{B\}$  where B is a single attribute *(singular)*
- ❖ A non-decomposed FD can always be transformed into the equivalent set of decomposed FD by *decomposition rule*
  - $\{BC\} \rightarrow \{DE\} \Leftrightarrow \{BC\} \rightarrow \{D\}$  and  $\{BC\} \rightarrow \{E\}$
- **Exercise:** Which of these are non-trivial decomposed FDs?





# Non-Trivial and Decomposed FD

- **Why?**

- We will check normal forms based on the non-trivial and decomposed FDs on a table
  - Trivial FDs are not interesting (*i.e., trivial*) because they convey no information
  - Decomposed FDs are easier to reason and they are still valid

- **Deriving Non-Trivial and Decomposed FD from a Table**

- Closure! *... non-trivialize, then decompose ...*
  1. Consider all subset of attributes in R *(do we need to consider R?)*
  2. Compute the closure of each subset
  3. Remove the “trivial” attributes
  4. Derive the decomposed FD from each closure

# Non-Trivial and Decomposed FD

- Deriving Non-Trivial and Decomposed FD from a Table

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

1. Consider all subset of attributes in R

$\{A\}$                        $\{B\}$                        $\{C\}$                        $\{AB\}$                        $\{AC\}$                        $\{BC\}$                        $\{ABC\}$

2. Compute the closure of each subset

$\{A\}^+ = \{ABC\}$      $\{B\}^+ = \{ABC\}$      $\{C\}^+ = \{C\}$      $\{AB\}^+ = \{ABC\}$      $\{AC\}^+ = \{ABC\}$      $\{BC\}^+ = \{ABC\}$

3. Remove the “trivial” attributes

$\{A\}^+ = \{A\}BC$      $\{B\}^+ = \{A\}B\{C\}$      $\{C\}^+ = \{C\}$      $\{AB\}^+ = \{A\}B\{C\}$      $\{AC\}^+ = \{A\}B\{C\}$      $\{BC\}^+ = \{A\}B\{C\}$

4. Derive the decomposed FD from each closure

$\{A\} \rightarrow \{B\}$      $\{A\} \rightarrow \{C\}$      $\{B\} \rightarrow \{A\}$      $\{B\} \rightarrow \{C\}$      $\{AB\} \rightarrow \{C\}$      $\{AC\} \rightarrow \{B\}$      $\{BC\} \rightarrow \{A\}$

# Non-Trivial and Decomposed FD

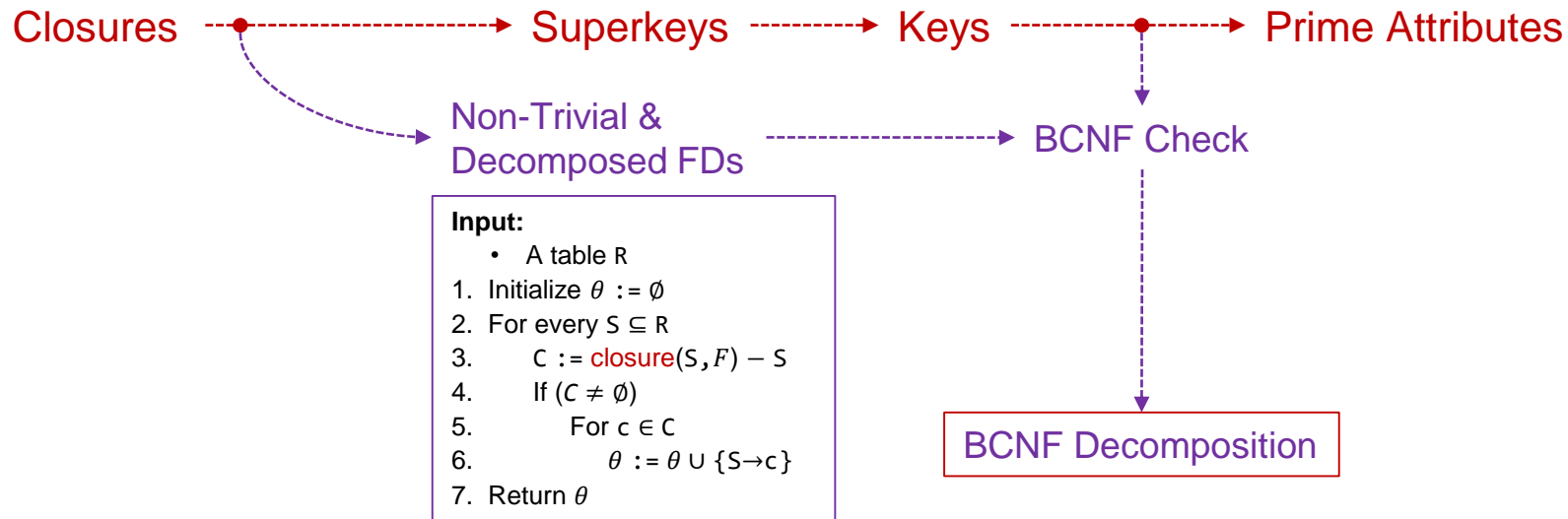
- Deriving Non-Trivial and Decomposed FD from a Table

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

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

$\{C\} \rightarrow \{A\}$	$\{AB\} \rightarrow \{C\}$	$\{AC\} \rightarrow \{D\}$	$\{ABC\} \rightarrow \{D\}$
$\{C\} \rightarrow \{D\}$	$\{AB\} \rightarrow \{D\}$	$\{BD\} \rightarrow \{A\}$	$\{ABD\} \rightarrow \{C\}$
$\{D\} \rightarrow \{A\}$	$\{BC\} \rightarrow \{A\}$	$\{BD\} \rightarrow \{C\}$	$\{BCD\} \rightarrow \{A\}$
	$\{BC\} \rightarrow \{D\}$	$\{CD\} \rightarrow \{A\}$	

# Algorithm Roadmap



# BCNF

## Boyce-Codd Normal Form



# BCNF

- **Definition**

- A table R is in BCNF if *every non-trivial and decomposed FD has a superkey as its left hand side*
- **Example:** R(A, B, C) with  $\{A\} \rightarrow \{B\}$ ,  $\{B\} \rightarrow \{A\}$  and  $\{B\} \rightarrow \{C\}$ 
  - Key:  $\{A\}$  and  $\{B\}$
  - Non-trivial and decomposed FDs on R:
    - $\{A\} \rightarrow \{B\}$     -  $\{A\} \rightarrow \{C\}$     -  $\{B\} \rightarrow \{A\}$     -  $\{B\} \rightarrow \{C\}$
    - $\{AB\} \rightarrow \{C\}$     -  $\{AC\} \rightarrow \{B\}$     -  $\{BC\} \rightarrow \{A\}$
  - For each of the above FD, the left hand side is a superkey  
 $\therefore$  R satisfies BCNF

# BCNF

- **Definition**

- A table R is in BCNF if *every non-trivial and decomposed FD has a superkey as its left hand side*
- **Example:** R(A, B, C) with  $\{A\} \rightarrow \{B\}$  and  $\{B\} \rightarrow \{C\}$ 
  - Key:  $\{A\}$
  - Non-trivial and decomposed FDs on R:
    - $\{A\} \rightarrow \{B\}$     -  $\{A\} \rightarrow \{C\}$     -  $\{B\} \rightarrow \{C\}$
    - $\{AB\} \rightarrow \{C\}$     -  $\{AC\} \rightarrow \{B\}$
  - B is **NOT** superkey in  $\{B\} \rightarrow \{C\}$   
∴ R does **NOT** satisfy BCNF

# BCNF

- **Intuition**

- A table  $R$  is in BCNF if *every non-trivial and decomposed FD has a superkey as its left hand side*
- **In other words**, any attribute  $B$  can depend **only** on superkeys
  - “In superkeys we trust”
  - Any dependency on non-superkeys is **prohibited**
- **Why?**
  - Suppose  $B$  depends on non-superkey  $C_1C_2 \dots C_n$
  - Since  $C_1C_2 \dots C_n$  is not a superkey, the same  $C_1C_2 \dots C_n$  may appear multiple times in the table *(why?)*
  - Whenever this happens, the same  $B$  would appear multiple times in the table

Table "Example"

$C_1$	$C_2$	$\dots$	$C_n$	$B$
c1	c2	...	cn	b
c1	c2	...	cn	b



# BCNF

- **Intuition**

- A table R is in BCNF if *every non-trivial and decomposed FD has a superkey as its left hand side*

- **Example:** The table shown on the right

- Key: {NRIC, Phone}

- FD that violates BCNF

- {NRIC}  $\rightarrow$  {Name}
- {NRIC}  $\rightarrow$  {Address}

- Since {NRIC} is not superkey, the same NRIC can appear multiple times in the table

- Every time NRIC is repeated, the corresponding Name and Address are also repeated

Table "Student\_Data"

Name	<u>NRIC</u>	<u>Phone</u>	Address
Alice	1234	67899876	Jurong East
Alice	1234	83838484	Jurong East
Bob	5678	98765432	Pasir Ris

# BCNF

- **Checking BCNF**

- A table R is **NOT** in BCNF if *there exists at least one non-trivial and decomposed FD such that its left hand side is NOT superkey*

- **By Counterexample:**

Can be done  
at the same time.  
Generate + check!

- 1. Consider all non-trivial and decomposed FDs of R
- 2. For each non-trivial and decomposed FDs of R, check that the left hand side is superkey
  - If not, then we have a counterexample
- 3. If no counterexample found, then R satisfies BCNF

# BCNF

## • Checking BCNF

- A table R is **NOT** in BCNF if *there exists at least one non-trivial and decomposed FD such that its left hand side is NOT superkey*

- **By Counterexample:**

1. Consider all subset of attributes of R

2. Compute the closure of each subset

Non superkey { 3. Consider only the subset that are not superkey (*closure  $\neq$  R*)

Non-trivial FD { 4. Remove the “trivial” attributes

5. If the resulting set is non-empty, we have a counterexample

### More but NOT All

Alternatively,  $\{S\} \subset \{S\}^+ \subset R$

The FD violating BCNF are  $\{S\} \rightarrow (\{S\}^+ - \{S\})$   
and can be decomposed if needed

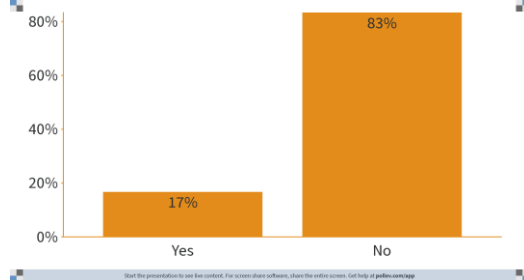
# BCNF

## • Checking BCNF

- A table R is **NOT** in BCNF if *there exists at least one non-trivial and decomposed FD such that its left hand side is NOT superkey*
- **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. If the resulting set is non-empty, we have a counterexample

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

# BCNF



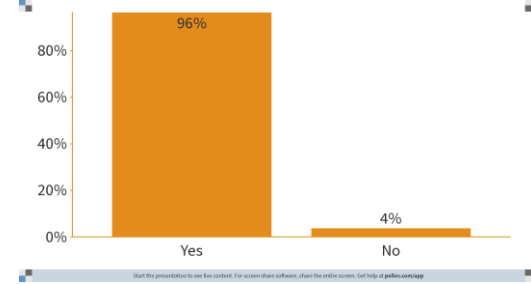
- **Checking BCNF**

- A table R is **NOT** in BCNF if *there exists at least one non-trivial and decomposed FD such that its left hand side is NOT superkey*

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

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

# BCNF



- **Checking BCNF**

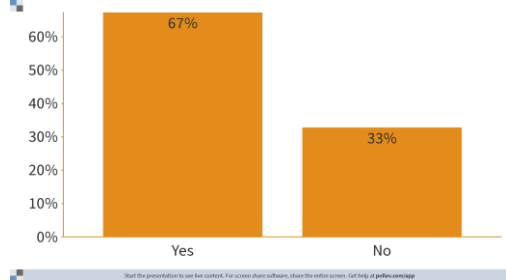
- A table R is **NOT** in BCNF if *there exists at least one non-trivial and decomposed FD such that its left hand side is NOT superkey*
- **Exercise:** R(A, B, C, D) with  $\{A\} \rightarrow \{B\}$ ,  $\{B\} \rightarrow \{C\}$ ,  $\{C\} \rightarrow \{D\}$  and  $\{D\} \rightarrow \{A\}$

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

## Note

Whenever we have  $\{A\} \rightarrow \{B\}$  and  $\{B\} \rightarrow \{A\}$ , we say that the attributes A and B are functionally equivalent. In such cases, we may actually remove either attribute A or attribute B (not the FD, but the attribute) from the database. For instance, eid and email in project. email can always be derived! Maybe as something like eid@company.name

# BCNF



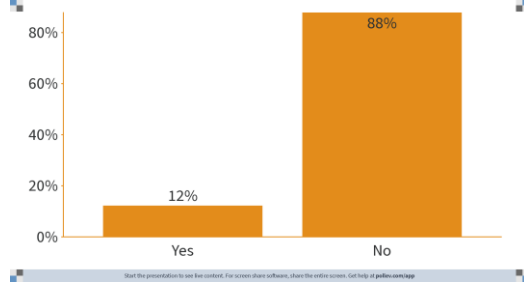
- **Checking BCNF**

- A table R is **NOT** in BCNF if *there exists at least one non-trivial and decomposed FD such that its left hand side is NOT superkey*

- **Exercise:** R(A, B, C, D) with  $\{AB\} \rightarrow \{D\}$ ,  $\{BD\} \rightarrow \{C\}$ ,  $\{CD\} \rightarrow \{A\}$  and  $\{AC\} \rightarrow \{B\}$

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

# BCNF



- **Checking BCNF**

- A table R is **NOT** in BCNF if *there exists at least one non-trivial and decomposed FD such that its left hand side is NOT superkey*
- **Exercise:** R(A, B, C, D, E) with  $\{AB\} \rightarrow \{C\}$ ,  $\{C\} \rightarrow \{E\}$ ,  $\{E\} \rightarrow \{A\}$  and  $\{E\} \rightarrow \{D\}$

$$\{A\}^+ = \{A\} \quad \bigg| \quad \{B\}^+ = \{B\} \quad \bigg| \quad \{C\}^+ = \{A, C, D, E\}$$

## Note

Remember the trick from before, we can always start from smallest subset. This way, the *chance* of the closure being non-trivial is greater.



# Algorithm Roadmap

Closures  $\dashrightarrow$  Superkeys  $\dashrightarrow$  Keys  $\dashrightarrow$  Prime Attributes

Non-Trivial &  
Decomposed FDs

**Input:**

- A table R
1. Initialize  $\theta := \emptyset$
  2. For every  $S \subseteq R$
  3.      $C := \text{closure}(S, F) - S$
  4.     If ( $C \neq \emptyset$ )
  5.         For  $c \in C$
  6.              $\theta := \theta \cup \{S \rightarrow c\}$
  7. Return  $\theta$

BCNF Check

BCNF Decomposition

**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 ( $S \subset C \subset R$ )
  4.         Return False
  5. Return True

# BCNF Decomposition

How to normalize a table to satisfy BCNF property



# BCNF Decomposition

- Normalization

Table "Student\_Data"

Name	<u>NRIC</u>	<u>Phone</u>	Address
Alice	1234	67899876	Jurong East
Alice	1234	83838484	Jurong East
Bob	5678	98765432	Pasir Ris

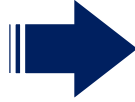


Table "Student\_Info"

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

Table "Student\_Contact"

<u>NRIC</u>	<u>Phone</u>
1234	67899876
1234	83838484
5678	98765432

- Idea:

- Decompose until all are in BCNF
  - Split the attributes into another table and remove any duplicates
  - The splitting need to ensure that BCNF violations are removed

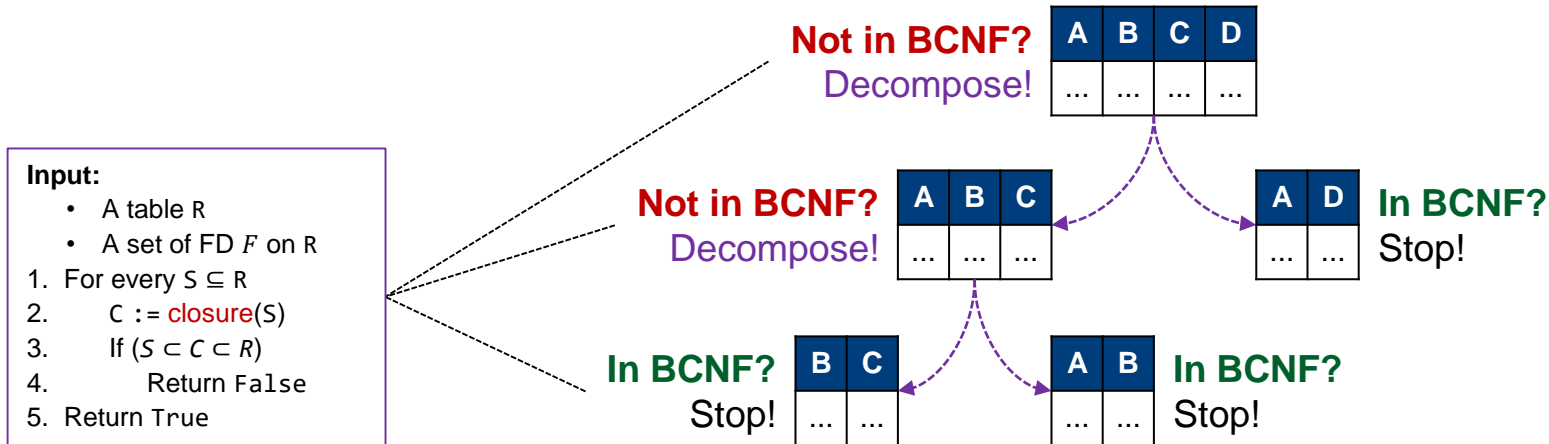
# BCNF Decomposition

- **Normalization**

- **Idea:**

- Decompose until all are in BCNF

- Split the attributes into another table and remove any duplicates
      - The splitting need to ensure that BCNF violations are removed



# BCNF Decomposition

- Normalization

- Problem:

- How to decompose such that we remove at least one BCNF violation?

- Idea:

- Let's look at the violation again
      - Say  $\{B\} \rightarrow \{CD\}$  violates BCNF property of  $R(A, B, C, D)$
    - In what table does  $\{B\} \rightarrow \{CD\}$  not violate BCNF property?
      - Make B a superkey!
      - $R_1(B, C, D)$
      - ❖ We need to return the *violation* and not just True/False

Input:

- A table R
  - A set of FD  $F$  on R
1. For every  $S \subseteq R$
  2.      $C := \text{closure}(S)$
  3.     If  $(S \subset C \subset R)$
  4.         Return **False**
  5. Return **True**

This is True/False,  
no information about  
violations

**More but NOT All**

Alternatively,  $\{S\} \subset \{S\}^+ \subset R$

The FD violating BCNF are  $\{S\} \rightarrow (\{S\}^+ - \{S\})$   
but  $\{S\} \rightarrow \{S\}^+$  is sufficient

# BCNF Decomposition

- Normalization Algorithm

- Input:

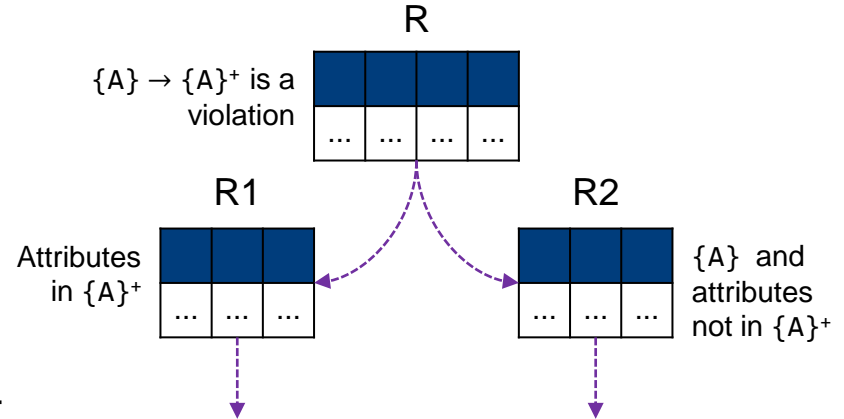
- A table  $R$
- A set of FD  $F$  on  $R$

1. Check if  $R$  is in BCNF

- a. If not, there is an FD  $\{A\} \rightarrow \{A\}^+$  that violates it then  $R$  decompose into
  - i.  $R_1(\text{everything in } \{A\}^+)$
  - ii.  $R_2(\text{everything in } \{A\} \text{ as well as attributes in } R \text{ but not in } \{A\}^+)$
  - iii. Recursively check  $R_1$  and  $R_2$
  - iv. Return the **union** of the result from recursive check
- b. Otherwise, stop
  - i. Return  **$\{R\}$**

Input:

- A table  $R$
  - A set of FD  $F$  on  $R$
1. For every  $S \subseteq R$
  2.  $C := \text{closure}(S)$
  3. If  $(S \subset C \subset R)$
  4. Return  **$\{S\} \rightarrow C$**
  5. Return **NULL**



# BCNF Decomposition

- **Normalization Algorithm**

- **Example:**

- Known FD:

- $\{NRIC\} \rightarrow \{Name, Address\}$

- **Steps:**

1. Check BCNF property

- $\{NRIC\} \rightarrow \{Name, NRIC, Address\}$  violates BCNF property

- a. Decompose Student\_Data into two tables

- i. Student\_Info(Name, NRIC, Address) *( $\{NRIC\}^+$ )*

- ii. Student\_Contact(NRIC, Phone) *( $\{NRIC\}$  and attributes not in  $\{NRIC\}^+$ )*

- iii. Check if Student\_Info(Name, NRIC, Address) and Student\_Contact(NRIC, Phone) are in BCNF *(they are...)*

Table "Student\_Data"

Name	<u>NRIC</u>	<u>Phone</u>	Address
Alice	1234	67899876	Jurong East
Alice	1234	83838484	Jurong East
Bob	5678	98765432	Pasir Ris

# BCNF Decomposition

And that is how  
we got this  
decomposition

Table "Student\_Data"

Name	<u>NRIC</u>	<u>Phone</u>	Address
Alice	1234	67899876	Jurong East
Alice	1234	83838484	Jurong East
Bob	5678	98765432	Pasir Ris



Table "Student\_Info"

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

Table "Student\_Contact"

<u>NRIC</u>	<u>Phone</u>
1234	67899876
1234	83838484
5678	98765432



# BCNF Decomposition

## Collected Answers

1. R1(A, B, C) ✖
2. R2(A, D) ✔

## • Normalization Algorithm

- **Example:** R(A, B, C, D) with  $\{A\} \rightarrow \{B\}$  and  $\{B\} \rightarrow \{C\}$
- **Steps:**
  1. Check BCNF property
    - $\{A\} \rightarrow \{A, B, C\}$  violates BCNF property
  - a. Decompose R into two tables
    - i. R1(A, B, C)  $(\{A\}^+)$
    - ii. R2(A, D)  $(\{A\} \cup (R - \{A\}^+))$
    - iii. Check if R1(A, B, C) and R2(A, D) are in BCNF

**NO!**

**YES!**

# BCNF Decomposition

- **Normalization Algorithm**

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

- **Steps:**

1. Check BCNF property of  $R1(A, B, C)$  with  $\{A\} \rightarrow \{B\}$  and  $\{B\} \rightarrow \{C\}$

- $\{B\} \rightarrow \{B, C\}$  violates BCNF property

- a. Decompose  $R1$  into two tables

- i.  $R3(B, C)$  ( $\{B\}^+$ )

- ii.  $R4(A, B)$  ( $\{B\} \cup (R1 - \{B\}^+)$ )

- iii. Check if  $R3(B, C)$  and  $R4(A, B)$  are in BCNF

**YES!**

**YES!**

## Collected Answers

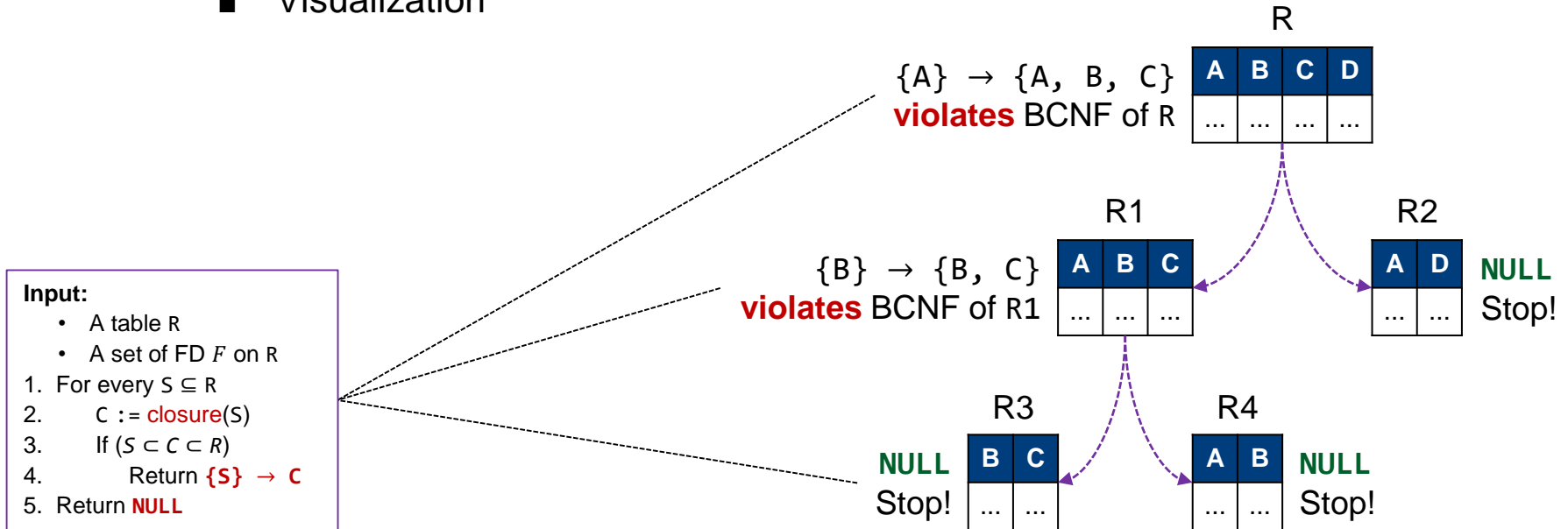
- |  |   |
|--|---|
| 1. <del><math>R1(A, B, C)</math></del> | ✗ |
| a. $R3(B, C)$                          | ✓ |
| b. $R4(A, B)$                          | ✓ |
| 2. $R2(A, D)$                          | ✓ |

# BCNF Decomposition

## • Normalization Algorithm

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

- Visualization



# BCNF Decomposition

- **Normalization Algorithm**

- **Notes:**

- The BCNF decomposition of a table may **NOT** be *unique*
      - Because any FD violating BCNF can be used for decomposition
    - If a table only has two attributes, then it is in BCNF
      - Why?
        - ❖ No need to check tables with only two attributes

- **Issues:**

- What happen if an FD has attributes *across* multiple tables?
    - Compute closure!
      - Then remove attributes that are irrelevant to the current table
    - Let's illustrate this with an example

# BCNF Decomposition

## Collected Answers

1.  $R_1(A, B)$  ✓
2.  $R_2(A, C, D, E)$  ?

## • Normalization Algorithm

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

■ **Steps:**

1. Check BCNF property

-  $\{A\} \rightarrow \{A, B\}$  violates BCNF property

a. Decompose  $R$  into two tables

i.  $R_1(A, B)$   $(\{A\}^+)$

ii.  $R_2(A, C, D, E)$   $(\{A\} \cup (R - \{A\}^+))$

iii. Check if  $R_1(A, B)$  and  $R_2(A, C, D, E)$  are in BCNF

**YES!**

**???**

*Do we even know what are  
the FDs that holds on  $R_2$ ?*

$\{A\} \rightarrow \{B\}$ ? But no  $B$

$\{BC\} \rightarrow \{D\}$ ? Also no  $B$

# BCNF Decomposition

## Collected Answers

1. R1(A, B) ✓
2. R2(A, C, D, E) ✗

## • Normalization Algorithm

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

■ **Steps:**

1. Check BCNF property of R2(A, C, D, E) with  $\{A\} \rightarrow \{B\}$  and  $\{BC\} \rightarrow \{D\}$ 
  - But all FD contains B and R2 has no attribute B
  - a. Compute closure of each subset of attributes
  - b. Remove attributes not in the current table
  - Now we can find the violation

(projection)

$\{A\}^+ = \{AB\}$	$\{C\}^+ = \{C\}$	$\{D\}^+ = \{D\}$	$\{E\}^+ = \{E\}$
$\{AC\}^+ = \{ABCD\}$	$\{AD\}^+ = \{ABD\}$	$\{AE\}^+ = \{AE\}$	
$\{CD\}^+ = \{CD\}$	$\{CE\}^+ = \{CE\}$	$\{DE\}^+ = \{DE\}$	
$\{ACD\}^+ = \{ABCD\}$	$\{ACE\}^+ = \{ABCDE\}$	$\{ADE\}^+ = \{ABDE\}$	$\{CDE\}^+ = \{CDE\}$

# BCNF Decomposition

## • Normalization Algorithm

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

■ **Steps:**

1. Check BCNF property of  $R_2(A, C, D, E)$  with  $\{A\} \rightarrow \{B\}$  and  $\{BC\} \rightarrow \{D\}$

-  $\{A, C\} \rightarrow \{A, C, D\}$  violates BCNF property

a. Decompose  $R_2$  into two tables

i.  $R_3(A, C, D)$   $(\{AC\}^+)$

ii.  $R_4(A, C, E)$   $(\{AC\} \cup (R_2 - \{AC\}^+))$

iii. Check if  $R_3(A, C, D)$  and  $R_4(A, C, E)$  are in BCNF

**YES!**

**YES!**

*We used  $\{A\} \rightarrow \{A, B\}$  as our FD violation.*

*But we can also use  $\{B, C\} \rightarrow \{B, C, D\}$  as our FD violation!*

***Let's redo our normalization algorithm***

### Collected Answers

1.  $R_1(A, B)$  ✓

2.  ~~$R_2(A, C, D, E)$~~ \*

a.  $R_3(A, D, E)$  ✓

b.  $R_4(A, C, E)$  ✓

# BCNF Decomposition

## Collected Answers

1. R1(B, C, D) ✓
2. R2(A, B, C, E) ✗

## • Normalization Algorithm

- **Example:** R(A, B, C, D, E) with  $\{A\} \rightarrow \{B\}$  and  $\{BC\} \rightarrow \{D\}$
- **Steps:**
  1. Check BCNF property
    - $\{B, C\} \rightarrow \{B, C, D\}$  violates BCNF property
  - a. Decompose R into two tables
    - i. R1(B, C, D)  $(\{BC\}^+)$
    - ii. R2(A, B, C, E)  $(\{BC\} \cup (R - \{BC\}^+))$
    - iii. Check if R1(B, C, D) and R2(A, B, C, E) are in BCNF  

YES!NO!



# BCNF Decomposition

## • Normalization Algorithm

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

■ **Steps:**

1. Check BCNF property of  $R_2(A, B, C, E)$  with  $\{A\} \rightarrow \{B\}$  and  $\{BC\} \rightarrow \{D\}$

-  $\{A\} \rightarrow \{A, B\}$  violates BCNF property

a. Decompose  $R_2$  into two tables

i.  $R_3(A, B)$   $(\{A\}^+)$

ii.  $R_4(A, C, E)$   $(\{A\} \cup (R_2 - \{A\}^+))$

iii. Check if  $R_3(A, B)$  and  $R_4(A, C, E)$  are in BCNF

**YES!**

**YES!**

**Starting with  $\{B, C\} \rightarrow \{B, C, D\}$**

1.  $R_1(B, C, D)$

2.  $R_3(A, B)$

3.  $R_4(A, C, E)$

**Starting with  $\{A\} \rightarrow \{A, B\}$**

1.  $R_1(A, B)$

2.  $R_3(A, D, E)$

3.  $R_4(A, C, E)$

### Collected Answers

1.  $R_1(B, C, D)$  ✓

2.  ~~$R_2(A, B, C, E)$~~ \*

a.  $R_3(A, B)$  ✓

b.  $R_4(A, C, E)$  ✓

# Algorithm Roadmap

Closures  $\dashrightarrow$  Superkeys  $\dashrightarrow$  Keys  $\dashrightarrow$  Prime Attributes

Non-Trivial &  
Decomposed FDs

BCNF Check

FD Projection

BCNF Decomposition

Input:

- A table  $R$
  - A set of FD  $F$  on  $R$
  - A set of attributes  $\{A\} \subseteq R$
1. Initialize  $\theta := \emptyset$
  2. For every  $S \subseteq R$
  3.      $C := \text{closure}(S, F)$
  4.      $\theta := \theta \cup \{\{S\} \rightarrow (C \cap A)\}$
  5. Return  $\theta$

Input:

- A table  $R$
1. Initialize  $\theta := \emptyset$
  2. For every  $S \subseteq R$
  3.      $C := \text{closure}(S, F) - S$
  4.     If  $(C \neq \emptyset)$
  5.         For  $c \in C$
  6.              $\theta := \theta \cup \{S \rightarrow c\}$
  7. Return  $\theta$

Input:

- A table  $R$
  - A set of FD  $F$  on  $R$
1. For every  $S \subseteq R$
  2.      $C := \text{project}(R, F, S)$
  3.     If  $(S \subset C \subset R)$
  4.         Return  $\{S\} \rightarrow C$
  5. Return **NULL**

Input:

- A table  $R$
  - A set of FD  $F$  on  $R$
1.  $f := \text{BCNF Check}(R, F)$
  2. If  $(f = \text{NULL})$
  3.     Return  $\{R\}$
  4.  $\{A\} \rightarrow \{A\}^+ := f$
  5. Return  $\text{BCNF}(R1(\{A\}^+)) \cup \text{BCNF}(R2(\{A\}^+ (R - \{A\}^+)))$

# BCNF Decomposition

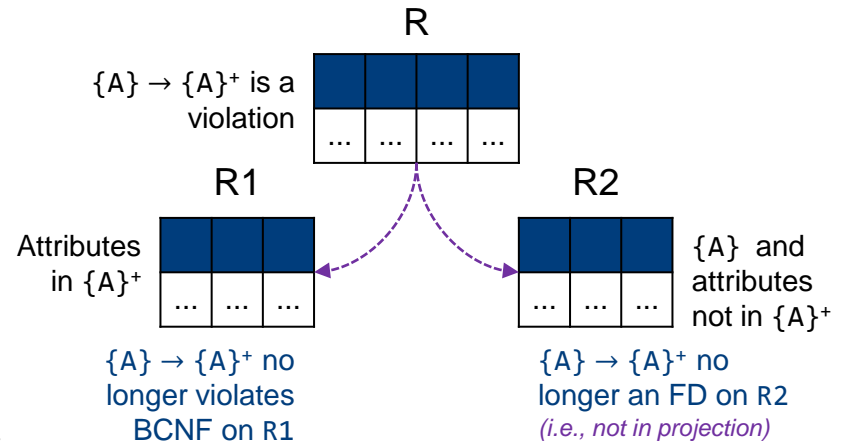
- **Normalization Algorithm**

- Why does the BCNF decomposition algorithm work?
  - In other words, how can it eliminate violations of BCNF?

- ❖ In general, each decomposition step removes at least one BCNF violation

- Then we recursively removes all violation
- How do we know it will not add new violations?

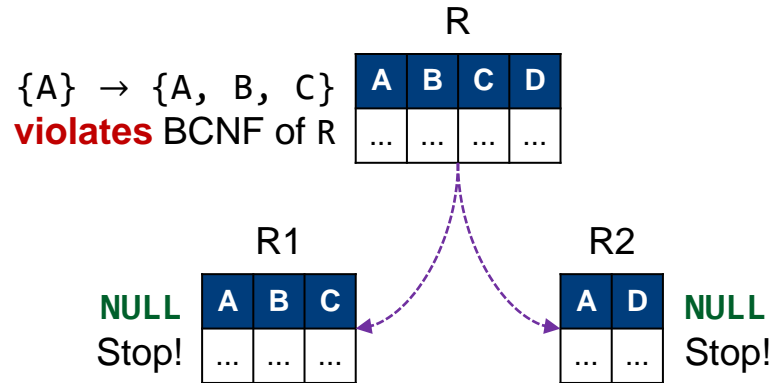
- At the worst case, we have only two attributes  $\Rightarrow$  always BCNF



# BCNF Decomposition

- Normalization Algorithm

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



## Simplified Algorithm

Find violation  $\{A\} \rightarrow \{A\}^+$

a.  $R1(\{A\}^+)$

b.  $R2(\{A\} \cup (R - \{A\}^+))$

Repeat with R1 and R2

# BCNF Decomposition

## Simplified Algorithm

Find violation  $\{A\} \rightarrow \{A\}^+$

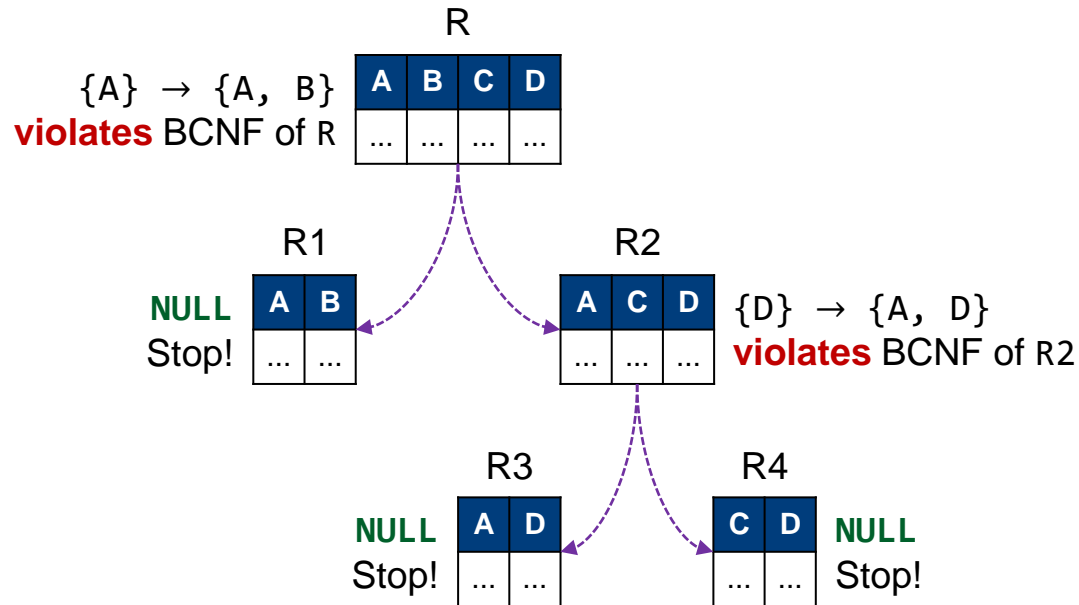
a.  $R1(\{A\}^+)$

b.  $R2(\{A\} \cup (R - \{A\}^+))$

Repeat with  $R1$  and  $R2$

## Normalization Algorithm

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



# BCNF Decomposition

- **Properties of BCNF**

- **Good** properties
  - No update or deletion or insertion anomalies
  - Small redundancies *(very hard to completely remove redundancies)*
  - The original table can always be reconstructed from the decomposed tables
    - How?

This is called **“Lossless Join”**

- ```
SELECT * FROM Student_Info, Student_Contact
WHERE Student_Info.NRIC = Student_Contact.NRIC
```

Table "Student\_Data"

| Name  | <u>NRIC</u> | <u>Phone</u> | Address     |
|-------|-------------|--------------|-------------|
| Alice | 1234        | 67899876     | Jurong East |
| Alice | 1234        | 83838484     | Jurong East |
| Bob   | 5678        | 98765432     | Pasir Ris   |



Table "Student\_Info"

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

Table "Student\_Contact"

| <u>NRIC</u> | <u>Phone</u> |
|-------------|--------------|
| 1234        | 67899876     |
| 1234        | 83838484     |
| 5678        | 98765432     |

# BCNF Decomposition

- **Lossless Join Decomposition**

- Say we decompose a table R into two tables R1 and R2
- The decomposition guarantees *lossless join*, whenever the common attributes in R1 and R2 constitute a superkey of R1 or R2
  - **Example:**
    - R(A, B, C) decomposed into R1(A, B) and R2(B, C) with B being a superkey of R2
    - R(A, B, C, D) decomposed into R1(A, B, C) and R2(B, C, D) with BC being a superkey of R1

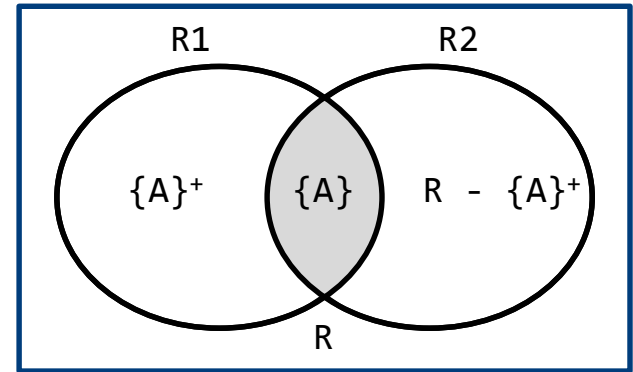
# BCNF Decomposition

## Lossless Join

The decomposition guarantees lossless join, whenever the *common attributes* in R1 and R2 constitute a superkey of R1 or R2

- **Lossless Join Decomposition of BCNF**

- BCNF decomposition **guarantee** lossless join decomposition
- **Why?**
  - Consider a violation  $\{A\} \rightarrow \{A\}^+$
  - Decomposed into  $R1(\{A\}^+)$  and  $R2(\{A\} \cup (R - \{A\}^+))$
  - **Common attributes?**
    - $\{A\}^+ \cap (\{A\} \cup (R - \{A\}^+))$   
 $\Rightarrow \{A\}$
  - But  $\{A\} \rightarrow \{A\}^+$  and  $R1(\{A\}^+)$ 
    - So  $\{A\}$  is a superkey of R1
    - $\therefore$  This decomposition guarantees lossless join decomposition





# BCNF Decomposition

- **Properties of BCNF**

- **Good** properties

- No update or deletion or insertion anomalies
    - Small redundancies *(very hard to completely remove redundancies)*
    - The original table can always be reconstructed from the decomposed tables

- **Bad** properties

- Dependencies may not be preserved *(non dependency-preserving decomposition)*
      - *Topic for next week lecture*

# QUESTION?