

# Database Systems

## Part-3

### Normalization

**Arijit Khan**

Associate Professor  
Department of Computer Science  
Aalborg University, Denmark

# Schedule for first half

Intro & ER	Sep 5 Monday (12:30-14:15), Lecture - FRB 7H	No exercise
Relation, FD, Keys	Sep 12 Monday (12:30-14:15), Lecture - FRB 7H	Exercise-1 on Intro & ER (14:30-16:30)
Normalization	Sep 19 Monday (12:30-14:15), Lecture - FRB 7H	Exercise-2 on Relation, FD, Keys (14:30-16:30)
Relational Algebra	Exercise-3 on Normalization (8:15-10:00)	Sep 23 Friday (10:15-12:00), Lecture - NOVI8
Release of Self- Study-1	Sep 26 Monday, No lecture, No exercise	
SQL – Part I	Oct 3 Monday (12:30-14:15), Lecture - FRB 7H	Exercise-4 on Relational Algebra (14:30-16:30)
Feedback of Self- Study-1	Oct 10 Monday (12:30-14:15), During lecture - FRB 7H	Exercise-5 on SQL – Part I (14:30-16:30)

# Ask Questions!



The important thing is not to  
stop questioning.

Albert Einstein

# Functional Dependency

- Schema:  $R = \{A:D_A, B:D_B, C:D_C, D:D_D\}$
- Instance:  $R$
- Let  $\alpha \subseteq R, \beta \subseteq R$
- $\alpha \rightarrow \beta$  iff  $\forall r, s \in R: r.\alpha = s.\alpha \Rightarrow r.\beta = s.\beta$

$R$			
$A$	$B$	$C$	$D$
a4	b2	c4	d3
a1	b1	c1	d1
a1	b1	c1	d2
a2	b2	c3	d2
a3	b2	c4	d3

$\{A\} \rightarrow \{B\}$

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

Not:  $\{B\} \rightarrow \{C\}$

Convention:

$CD \rightarrow B$

# Example

Family Tree				
Child	Father	Mother	Grandma	Grandpa
Sofie	Alfons	Sabine	Lothar	Linde
Sofie	Alfons	Sabine	Hubert	Lisa
Niklas	Alfons	Sabine	Lothar	Linde
Niklas	Alfons	Sabine	Hubert	Lisa
...	...	...	Lothar	Martha
...	...	...	...	...

- Child → Father, Mother
- Child, Grandpa → Grandma
- Child, Grandma → Grandpa

# Analogy to functions

- $f1 : \text{Child} \rightarrow \text{Father}$ 
  - E.g.,  $f1(\text{Niklas}) = \text{Alfons}$
- $f2: \text{Child} \rightarrow \text{Mother}$ 
  - E.g.,  $f2(\text{Niklas}) = \text{Sabine}$
- $f3: \text{Child} \times \text{Grandpa} \rightarrow \text{Grandma}$
- $\text{FD}: \text{Child} \rightarrow \text{Father, Mother}$ 
  - represents two functions ( $f1, f2$ )
  - Komma on right side indicates multiple functions
- $\text{FD}: \text{Child, Grandpa} \rightarrow \text{Grandma}$ 
  - Komma on the left side indicates Cartesian product

# Keys



- $\alpha \subseteq R$  is a superkey iff
  - $\alpha \rightarrow R$
- $\alpha \rightarrow \beta$  is minimal iff
  - $\forall A \in \alpha: \neg((\alpha - \{A\}) \rightarrow \beta)$
- Notation for minimal functional dependencies:  
 $\alpha \rightarrow^{\cdot} \beta$
- $\alpha \subseteq R$  is a key (or candidate key) iff
  - $\alpha \rightarrow^{\cdot} R$

# Armstrong Axioms: Inference of FDs



- Reflexivity
  - $(\beta \subseteq \alpha) \Rightarrow \alpha \rightarrow \beta$
  - Special case:  $\alpha \rightarrow \alpha$
- Augmentation
  - $\alpha \rightarrow \beta \Rightarrow \alpha\gamma \rightarrow \beta\gamma.$
  - (Notation  $\alpha\gamma := \alpha \cup \gamma$ )
- Transitivity
  - $\alpha \rightarrow \beta \wedge \beta \rightarrow \gamma \Rightarrow \alpha \rightarrow \gamma.$
- These three axioms are complete. All possible other rules can be implied from these axioms.



# Questions?



# Minimal Basis

- Given a set  $S$  of FDs, the **minimal basis** of  $S$  is a **simplified** version of  $S$
- Example:
  - $S = \{A \rightarrow BD, AB \rightarrow C, C \rightarrow D, BC \rightarrow D\}$
  - A minimal basis:  $\{A \rightarrow B, A \rightarrow C, C \rightarrow D\}$
- How simplified?
- Three conditions.
- **Condition 1:** For any FD in the minimal basis, its right hand side has only one attribute.
- Example in  $S$ :  $A \rightarrow BD$  does not satisfy this condition
- That is why  $A \rightarrow BD$  is not in the minimal basis

# Minimal Basis

- Example:
  - $S = \{A \rightarrow BD, AB \rightarrow C, C \rightarrow D, BC \rightarrow D\}$
  - A minimal basis:  $\{A \rightarrow B, A \rightarrow C, C \rightarrow D\}$
- **Condition 2:** No FD in the minimal basis is redundant.
- That is, no FD in the minimal basis can be derived from the other FDs.
- Example in S:  $BC \rightarrow D$  can be derived from  $C \rightarrow D$
- That is why  $BC \rightarrow D$  is not in the minimal basis

# Minimal Basis

- Example:
  - $S = \{A \rightarrow BD, AB \rightarrow C, C \rightarrow D, BC \rightarrow D\}$
  - A minimal basis:  $\{A \rightarrow B, A \rightarrow C, C \rightarrow D\}$
- **Condition 3:** For each FD in the minimal basis, none of the attributes on the left hand side is redundant
- That is, for any FD in the minimal basis, if we remove an attribute from the left hand side, then the resulting FD is a new FD that cannot be derived from the original set of FDs
- Example:
  - S contains  $AB \rightarrow C$
  - If we remove B from the left hand side, we have  $A \rightarrow C$
  - $A \rightarrow C$  can be derived from S, as  $\{A\}^+ = \{ABDC\}$
  - This indicates that  $A \rightarrow C$  is “hidden” in S
  - Hence, we can replace  $AB \rightarrow C$  with  $A \rightarrow C$ , as  $A \rightarrow C$  is “simpler”
  - This is why  $AB \rightarrow C$  is not in the minimal basis

# Algorithm for Minimal Basis



- Given: a set  $S$  of FDs
- Example:  $S = \{A \rightarrow BD, AB \rightarrow C, C \rightarrow D, BC \rightarrow D\}$
- **Step 1:** Transform the FDs, so that each right hand side contains only one attribute
- Result:  $S = \{A \rightarrow B, A \rightarrow D, AB \rightarrow C, C \rightarrow D, BC \rightarrow D\}$
- Reason:
  - Condition 1 for minimal basis:  
The right hand side of each FD contains only one attribute

# Algorithm for Minimal Basis



- Result of Step 1:
  - $S = \{A \rightarrow B, A \rightarrow D, AB \rightarrow C, C \rightarrow D, BC \rightarrow D\}$
- **Step 2:** Remove redundant FDs
- Is  $A \rightarrow B$  redundant?
- i.e., is  $A \rightarrow B$  implied by other FDs in  $S$ ?
- Let's check
- Without  $A \rightarrow B$ , we have  $\{A \rightarrow D, AB \rightarrow C, C \rightarrow D, BC \rightarrow D\}$
- Given those FDs, we have  $\{A\}^+ = \{AD\}$ , which does not contain  $B$
- Therefore,  $A \rightarrow B$  is not implied by the other FDs

# Algorithm for Minimal Basis



- Result of Step 1:
  - $S = \{A \rightarrow B, A \rightarrow D, AB \rightarrow C, C \rightarrow D, BC \rightarrow D\}$
- **Continue Step 2:** Remove redundant FDs
- Is  $A \rightarrow D$  redundant?
- i.e., is  $A \rightarrow D$  implied by other FDs in  $S$ ?
- Let's check
- Without  $A \rightarrow D$ , we have  $\{A \rightarrow B, AB \rightarrow C, C \rightarrow D, BC \rightarrow D\}$
- Given those FDs, we have  $\{A\}^+ = \{ABCD\}$ , which contains  $D$
- Therefore,  $A \rightarrow D$  is implied by the other FDs
- Hence,  $A \rightarrow D$  is redundant and should be removed
- Result:  $S = \{A \rightarrow B, AB \rightarrow C, C \rightarrow D, BC \rightarrow D\}$

# Algorithm for Minimal Basis



- Result of the last step:
  - $S = \{A \rightarrow B, AB \rightarrow C, C \rightarrow D, BC \rightarrow D\}$
- **Continue Step 2:** Remove redundant FDs
- Is  $AB \rightarrow C$  redundant?
- i.e., is  $AB \rightarrow C$  implied by other FDs in  $S$ ?
- Let's check
- Without  $AB \rightarrow C$ , we have  $\{A \rightarrow B, C \rightarrow D, BC \rightarrow D\}$
- Given those FDs, we have  $\{AB\}^+ = \{ABD\}$ , which does not contain  $C$
- Therefore,  $AB \rightarrow C$  is NOT implied by the other FDs
- Hence,  $AB \rightarrow C$  is not redundant and should not be removed



# Algorithm for Minimal Basis



- Result of the last step:
  - $S = \{A \rightarrow B, AB \rightarrow C, C \rightarrow D, BC \rightarrow D\}$
- **Continue Step 2:** Remove redundant FDs
- Is  $C \rightarrow D$  redundant?
- i.e., is  $C \rightarrow D$  implied by other FDs in  $S$ ?
- Let's check
- Without  $C \rightarrow D$ , we have  $\{A \rightarrow B, AB \rightarrow C, BC \rightarrow D\}$
- Given those FDs, we have  $\{C\}^+ = \{C\}$ , which does not contain  $D$
- Therefore,  $C \rightarrow D$  is NOT implied by the other FDs and should not be removed

# Algorithm for Minimal Basis



- Result of the last step:
  - $S = \{A \rightarrow B, AB \rightarrow C, C \rightarrow D, BC \rightarrow D\}$
- **Continue Step 2:** Remove redundant FDs
- Is  $BC \rightarrow D$  redundant?
- i.e., is  $BC \rightarrow D$  implied by other FDs in  $S$ ?
- Let's check
- Without  $BC \rightarrow D$ , we have  $\{A \rightarrow B, AB \rightarrow C, C \rightarrow D\}$
- Given those FDs, we have  $\{BC\}^+ = \{BCD\}$ , which contains  $D$
- Therefore,  $BC \rightarrow D$  is implied by the other FDs
- Hence,  $BC \rightarrow D$  is redundant and should be removed
- Result:  $S = \{A \rightarrow B, AB \rightarrow C, C \rightarrow D\}$

# Algorithm for Minimal Basis



- Result of Step 2:
  - $S = \{A \rightarrow B, AB \rightarrow C, C \rightarrow D\}$
- **Step 3:** Remove redundant attributes on the left hand side (lhs) of each FD
- Only  $AB \rightarrow C$  has more than one attribute on the lhs
- Let's check
- Is A redundant?
- If we remove A, then  $AB \rightarrow C$  becomes  $B \rightarrow C$
- Whether this removal is OK depends on whether  $B \rightarrow C$  is "hidden" in S already
- If  $B \rightarrow C$  is "hidden" in S, then the removal of A is OK, (since the removal does not add extraneous information into S)
- Is  $B \rightarrow C$  "hidden" in S?
- Check: Given S, we have  $\{B\}^+ = \{B\}$ , which does NOT contain C
- Therefore,  $B \rightarrow C$  is not "hidden" in S, and hence, A is NOT redundant

# Algorithm for Minimal Basis



- Result of Step 2:
  - $S = \{A \rightarrow B, AB \rightarrow C, C \rightarrow D\}$
- **Step 3:** Remove redundant attributes on the left hand side (lhs) of each FD
- Only  $AB \rightarrow C$  has more than one attribute on the lhs
- Let's check
- Is B redundant?
- If we remove B, then  $AB \rightarrow C$  becomes  $A \rightarrow C$
- Whether this is OK depends on whether  $A \rightarrow C$  is "hidden" in S
- Is  $A \rightarrow C$  "hidden" in S?
- Check: Given S, we have  $\{A\}^+ = \{ABCD\}$ , which contains C
- Therefore,  $A \rightarrow C$  is "hidden" in S
- Hence, we can simplify  $AB \rightarrow C$  to  $A \rightarrow C$
- Final minimal basis:  $S = \{A \rightarrow B, A \rightarrow C, C \rightarrow D\}$

# Minimal Basis is not always unique



- For given set of FDs, its minimal basis may not be unique
- Example:
  - Given  $R(A, B, C)$  and  $\{A \rightarrow B, A \rightarrow C, B \rightarrow C, B \rightarrow A, C \rightarrow A, C \rightarrow B\}$
  - Minimal basis 1:  $\{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$
  - Minimal basis 2:  $\{A \rightarrow C, B \rightarrow C, C \rightarrow A, C \rightarrow B\}$

# Questions?



# Classroom Exercise-1



## Minimal Basis Exercise:

- $S = \{A \rightarrow C, AC \rightarrow D, AD \rightarrow B\}$ 
  1. Transform the FDs to ensure that the right hand side of each FD has only one attribute
  2. See if any FD can be derived from the other FDs. Remove those FDs one by one
  3. Check if we can remove any attribute from the left hand side of any FD

# Classroom Exercise-1: Solution



## Minimal Basis Exercise:

- $S = \{A \rightarrow C, AC \rightarrow D, AD \rightarrow B\}$ 
  1. Transform the FDs to ensure that the right hand side of each FD has only one attribute
- **Results:**  $M = \{A \rightarrow C, AC \rightarrow D, AD \rightarrow B\}$ 
  2. See if any FD can be derived from the other FDs. Remove those FDs one by one



# Classroom Exercise-1: Solution



## Minimal Basis Exercise:

- $M = \{A \rightarrow C, AC \rightarrow D, AD \rightarrow B\}$
- 2. See if any FD can be derived from the other FDs.  
Remove those FDs one by one.
- Try  $A \rightarrow C$  first
  - If  $A \rightarrow C$  is removed, then the ones left would be  $AC \rightarrow D, AD \rightarrow B$
  - With the remaining FDs, we have  $\{A\}^+ = \{A\}$
  - Since  $\{A\}^+$  does not contain  $C$ , we know that  $A \rightarrow C$  cannot be derived from the remaining FDs
  - Therefore,  $A \rightarrow C$  cannot be removed

# Classroom Exercise-1: Solution



## Minimal Basis Exercise:

- $M = \{A \rightarrow C, AC \rightarrow D, AD \rightarrow B\}$
- 2. See if any FD can be derived from the other FDs.  
Remove those FDs one by one.
- Next, try  $AC \rightarrow D$ 
  - If  $AC \rightarrow D$  is removed, then the ones left would be  $A \rightarrow C, AD \rightarrow B$
  - With the remaining FDs, we have  $\{AC\}^+ = \{AC\}$
  - Since  $\{AC\}^+$  does not contain  $D$ , we know that  $AC \rightarrow D$  cannot be derived from the remaining FDs
  - Therefore,  $AC \rightarrow D$  cannot be removed

# Classroom Exercise-1: Solution



## Minimal Basis Exercise:

- $M = \{A \rightarrow C, AC \rightarrow D, AD \rightarrow B\}$

2. See if any FD can be derived from the other FDs.  
Remove those FDs one by one.

- Next, try  $AD \rightarrow B$ 
  - If  $AD \rightarrow B$  is removed, then the ones left would be  $A \rightarrow C, AC \rightarrow D$
  - With the remaining FDs, we have  $\{AD\}^+ = \{AD\}$
  - Since  $\{AD\}^+$  does not contain  $B$ , we know that  $AD \rightarrow B$  cannot be derived from the remaining FDs
  - Therefore,  $AD \rightarrow B$  cannot be removed

# Classroom Exercise-1: Solution



## Minimal Basis Exercise:

- $M = \{A \rightarrow C, AC \rightarrow D, AD \rightarrow B\}$
- 3. Check if we can remove any attribute from the left hand side of any FD
- First, try to remove A from  $AC \rightarrow D$ 
  - It results in  $C \rightarrow D$
  - Can  $C \rightarrow D$  be derived from M?
  - $\{C\}^+ = \{C\}$  given M.
  - Since  $\{C\}^+$  does not contain D, we know that  $C \rightarrow D$  cannot be derived from M
  - Therefore, A cannot be removed from  $AC \rightarrow D$

# Classroom Exercise-1: Solution



## Minimal Basis Exercise:

- $M = \{A \rightarrow C, AC \rightarrow D, AD \rightarrow B\}$

3. Check if we can remove any attribute from the left hand side of any FD

- Next, try to remove C from  $AC \rightarrow D$ 
  - It results in  $A \rightarrow D$
  - Can  $A \rightarrow D$  be derived from M?
  - $\{A\}^+ = \{ABCD\}$  given M.
  - Since  $\{A\}^+$  contains D, we know that  $A \rightarrow D$  can be derived from M
  - Therefore, C can be removed from  $AC \rightarrow D$
- New  $M = \{A \rightarrow C, A \rightarrow D, AD \rightarrow B\}$

# Classroom Exercise-1: Solution



## Minimal Basis Exercise:

- New  $M = \{A \rightarrow C, A \rightarrow D, AD \rightarrow B\}$
- 3. Check if we can remove any attribute from the left hand side of any FD
- Next, try to remove  $A$  from  $AD \rightarrow B$ 
  - It results in  $D \rightarrow B$
  - Can  $D \rightarrow B$  be derived from  $M$ ?
  - $\{D\}^+ = \{D\}$  given  $M$ .
  - Since  $\{D\}^+$  does not contain  $B$ , we know that  $D \rightarrow B$  cannot be derived from  $M$
  - Therefore,  $A$  cannot be removed from  $AD \rightarrow B$

# Classroom Exercise-1: Solution



## Minimal Basis Exercise:

- $M = \{A \rightarrow C, A \rightarrow D, AD \rightarrow B\}$
- 3. Check if we can remove any attribute from the left hand side of any FD
- Next, try to remove D from  $AD \rightarrow B$ 
  - It results in  $A \rightarrow B$
  - Can  $A \rightarrow B$  be derived from M?
  - $\{A\}^+ = \{ABCD\}$  given M.
  - Since  $\{A\}^+$  contains B, we know that  $A \rightarrow B$  can be derived from M
  - Therefore, D can be removed from  $AD \rightarrow B$
- New  $M = \{A \rightarrow C, A \rightarrow D, A \rightarrow B\}$ ; done

# Bad Schemas



ProfLecture						
PersNr	Name	Level	Room	Nr	Title	CP
2125	Sokrates	FP	226	5041	Ethik	4
2125	Sokrates	FP	226	5049	Mäeutik	2
2125	Sokrates	FP	226	4052	Logik	4
...	...	...	...	...	...	...
2132	Popper	AP	52	5259	Der Wiener Kreis	2
2137	Kant	FP	7	4630	Die 3 Kritiken	4

- Update-Anomaly
  - What happens when Sokrates moves to a different room?
- Insert-Anomaly
  - What happens if Roscoe is elected as a new professor?
- Delete-Anomaly
  - What happens if Popper does not teach this semester?



# Decomposition of Relations

- Bad relations combine several concepts
  - decompose them so that each concept in one relation
  - $R \rightarrow R_1, \dots, R_n$

## 1. Valid Decomposition

Decomposition of  $R$  into  $R_1$  and  $R_2$  is valid if  $R = R_1 \cup R_2$ , i.e., no attributes in  $R$  get lost.

$R_1 := \pi_{R_1}(R)$

$R_2 := \pi_{R_2}(R)$

## 2. Lossless Decomposition

$$R = R_1 \bowtie R_2 \bowtie \dots \bowtie R_n$$

Lossless decomposition: all data in the original schema must be reconstructible with a natural join from the instances of the new schemas

## 3. Preservation of Dependencies

- $FD(R)^+ = (FD(R_1) \cup \dots \cup FD(R_n))^+$

# When is a decomposition lossless?

- Let  $R = R1 \cup R2$ 
  - $R1 := \pi_{R1}(R)$
  - $R2 := \pi_{R2}(R)$
- Lemma: The decomposition is lossless if
  - $(R1 \cap R2) \rightarrow R1$  or
  - $(R1 \cap R2) \rightarrow R2$
- 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 the key of  $R2$
  - $R(A, B, C, D)$  decomposed into  $R1(A, B, C)$  and  $R2(B, C, D)$ , with  $BC$  being the key of  $R1$

# Lossless Join Example

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

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

# Lossy Join Example

<i>Drinker</i>		
<i>Pub</i>	<i>Guest</i>	<i>Beer</i>
Kowalski	Kemper	Pils
Kowalski	Eickler	Hefeweizen
Innsteg	Kemper	Hefeweizen

<i>Visitor</i>	
<i>Pub</i>	<i>Guest</i>
Kowalski	Kemper
Kowalski	Eickler
Innsteg	Kemper



<i>Drinks</i>	
<i>Guest</i>	<i>Beer</i>
Kemper	Pils
Eickler	Hefeweizen
Kemper	Hefeweizen

<i>Visitor ⋈ Drinks</i>		
<i>Pub</i>	<i>Guest</i>	<i>Beer</i>
Kowalski	Kemper	Pils
Kowalski	Kemper	Hefeweizen
Kowalski	Eickler	Hefeweizen
Innsteg	Kemper	Pils
Innsteg	Kemper	Hefeweizen

≠

# Lossy Join Example



- Drinker has one (non-trivial) functional dependency
  - $\{\text{Pub}, \text{Guest}\} \rightarrow \{\text{Beer}\}$
- But none of the criteria of Lossless join hold
  - $\{\text{Guest}\} \rightarrow \{\text{Beer}\}$  
  - $\{\text{Guest}\} \rightarrow \{\text{Pub}\}$  
- The problem is that Kemper likes different beer in different pubs.

# Classroom Exercise-2



- Let  $R$  be decomposed into  $R_1, \dots, R_n$
- $F_R = (F_{R_1} \cup \dots \cup F_{R_n})$
- ZipCodes: {[Street, City, Canton, Zip]}
- Functional dependencies in ZipCodes
  - $\{\text{Zip}\} \rightarrow \{\text{City, Canton}\}$
  - $\{\text{Street, City, Canton}\} \rightarrow \{\text{Zip}\}$
- What about this decomposition?
  - Streets: {[Zip, Street]}
  - Cities: {[Zip, City, Canton]}
- Is it lossless? Does it preserve functional depend.?

## Classroom Exercise-2: Solution



- Let  $R$  be decomposed into  $R_1, \dots, R_n$
- $F_R = (F_{R_1} \cup \dots \cup F_{R_n})$
- ZipCodes: {[Street, City, Canton, Zip]}
- Functional dependencies in ZipCodes
  - $\{\text{Zip}\} \rightarrow \{\text{City, Canton}\}$
  - $\{\text{Street, City, Canton}\} \rightarrow \{\text{Zip}\}$
- What about this decomposition?
  - Streets: {[Zip, Street]}
  - Cities: {[Zip, City, Canton]}
- Is it lossless? Does it preserve functional depend.?
- **Lossless  $\rightarrow$  YES; preserve functional depend.  $\rightarrow$  NO**

# Decomposition of ZipCodes

<i>ZipCodes</i>			
<i>City</i>	<i>Canton</i>	<i>Street</i>	<i>Zip</i>
Buchs	AG	Goethestr.	5033
Buchs	AG	Schillerstr.	5034
Buchs	SG	Goethestr.	8107

$\Pi_{\text{Zip, Street}}$

$\Pi_{\text{City, Canton, Zip}}$

<i>Streets</i>	
<i>Zip</i>	<i>Street</i>
8107	Goethestr.
5033	Goethestr.
5034	Schillerstr.

<i>Cities</i>		
<i>City</i>	<i>Canton</i>	<i>Zip</i>
Buchs	AG	5033
Buchs	AG	5034
Buchs	SG	8107

$\{\text{Street, City, Canton}\} \rightarrow \{\text{Zip}\}$  not checkable in decomp. schema

**It is possible to insert inconsistent tuples**



# Violation of City,Canton,Street→Zip

<i>ZipCodes</i>			
<i>City</i>	<i>Canton</i>	<i>Street</i>	<i>Zip</i>
Buchs	AG	Goethestr.	5033
Buchs	AG	Schillerstr.	5034
Buchs	SG	Goethestr.	8107

$\Pi_{\text{Zip,Street}}$

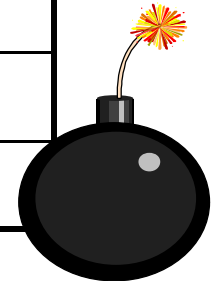
$\Pi_{\text{City,Canton,Zip}}$

<i>Streets</i>	
<i>Zip</i>	<i>Street</i>
8107	Goethestr.
5033	Goethestr.
5034	Schillerstr.
8108	Goethestr.

<i>Cities</i>		
<i>City</i>	<i>Canton</i>	<i>Zip</i>
Buchs	AG	5033
Buchs	AG	5034
Buchs	SG	8107
Buchs	SG	8108

# Violation of City,Canton,Street→Zip

<i>ZipCodes</i>			
<i>City</i>	<i>Canton</i>	<i>Street</i>	<i>Zip</i>
Buchs	AG	Goethestr.	5033
Buchs	AG	Schillerstr.	5034
Buchs	SG	Goethestr.	8107
Buchs	SG	Goethestr.	8108



<i>Streets</i>	
<i>Zip</i>	<i>Street</i>
8107	Goethestr.
5033	Goethestr.
5034	Schillerstr.
8108	Goethestr.

<i>Cities</i>		
<i>City</i>	<i>Canton</i>	<i>Zip</i>
Buchs	AG	5033
Buchs	AG	5034
Buchs	SG	8107
Buchs	SG	8108

# 5 Minutes Interval



The important thing is not to  
stop questioning.

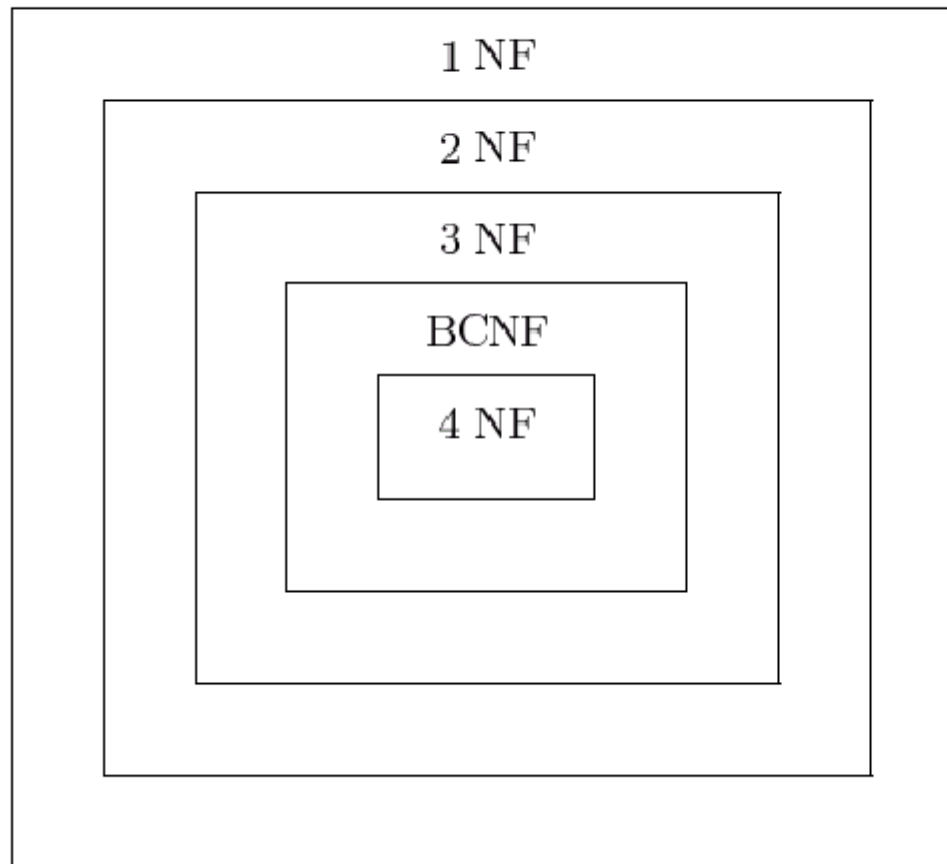
Albert Einstein

# Normal Forms

- Conditions that a “good” table should satisfy
- Various normal forms (in increasing order of strictness)
  - First normal form
  - Second normal form
  - Third normal form (3NF)
  - Boyce-Codd normal form (BCNF)
  - Fourth normal form
  - Fifth normal form
  - Sixth normal form

# Normal Forms

- Lossless decomposition up to 4NF
- Preserving dependencies up to 3NF



# First Normal Form

- Only atomic domains

vs.

<i>Parents</i>		
<i>Father</i>	<i>Mother</i>	<i>Children</i>
Johann	Martha	{Else, Lucie}
Johann	Maria	{Theo, Josef}
Heinz	Martha	{Cleo}

Not in  
1NF

<i>Parents</i>		
<i>Father</i>	<i>Mother</i>	<i>Child</i>
Johann	Martha	Else
Johann	Martha	Lucie
Johann	Maria	Theo
Johann	Maria	Josef
Heinz	Martha	Cleo

in  
1NF

# Second Normal Form

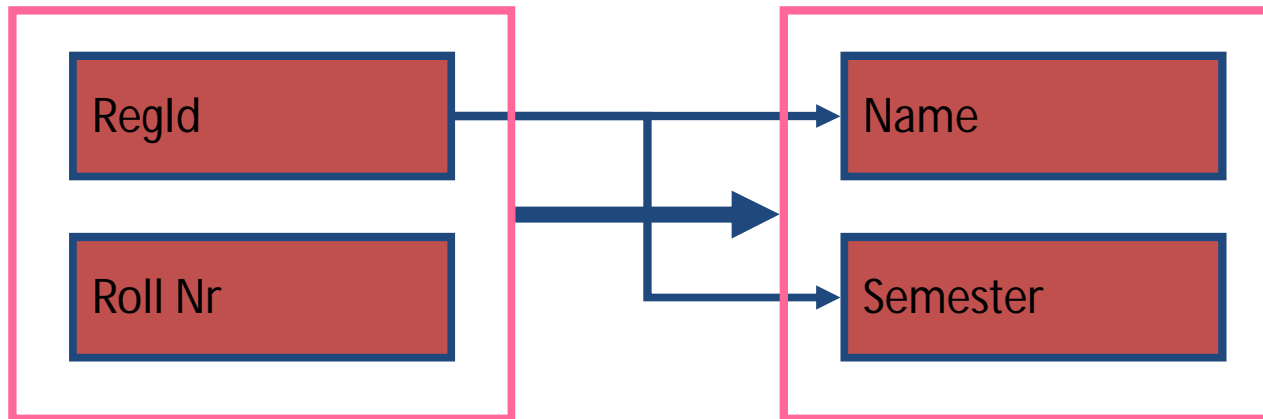


- R is in 2NF iff every non-key attribute is functionally dependent on the whole of every candidate key.

StudentAttends			
RegId	Roll Nr	Name	Semester
26120	5001	Fichte	10
27550	5001	Schopenhauer	6
27550	4052	Schopenhauer	6
28106	5041	John	3
28106	5052	John	3
28106	5216	John	3
28106	5259	John	3
...	...	...	...

- StudentAttends is not in 2NF!!!
  - {RegId}  $\rightarrow$  {Name, Semester}

# Second Normal Form



- Insert Anomaly: What about students who attend no lecture?
- Update Anomaly: Promotion of John to the 4th semester.
- Delete Anomaly: Fichte drops his last course?
- **Solution:** Decompose into two relations
  - attends: {[**RegId**, **Roll Nr**]}
  - Student: {[**RegId**, Name, Semester]}
- Student, attends are in 2NF. The decomposition is lossless and preserves dependencies.



# Third Normal Form (3NF)



- **Definition:** A table satisfies 3NF, if and only if for every non-trivial  $X \rightarrow Y$ 
  - Either  $X$  is a superkey
  - Or each attribute in  $Y$  is either contained in a key or in  $X$

- **Example:**

- Given FDs:  $C \rightarrow B$ ,  $AB \rightarrow C$ ,  $BC \rightarrow C$
- Keys:  $\{AB\}$ ,  $\{AC\}$
- $AB \rightarrow C$  is OK, since  $AB$  is a key of  $R$
- $C \rightarrow B$  is OK, since  $B$  is in a key of  $R$
- $BC \rightarrow C$  is OK, since  $C$  is in  $BC$
- So  $R$  is in 3NF

R	A	B	C

# Third Normal Form (3NF)

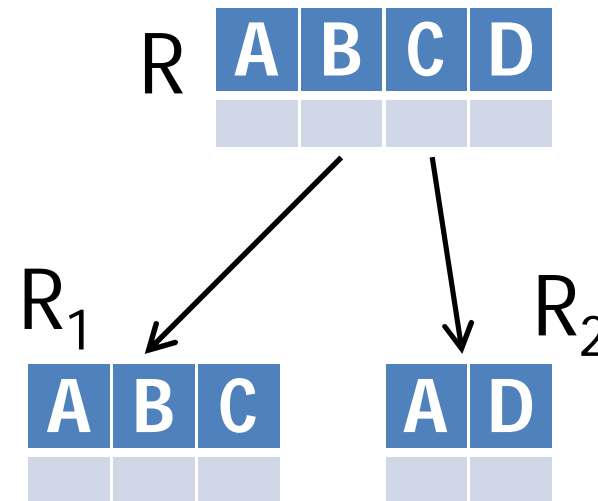


- **Definition:** A table satisfies 3NF, if and only if for every non-trivial  $X \rightarrow Y$ 
  - Either  $X$  is a superkey
  - Or each attribute in  $Y$  is either contained in a key or in  $X$
- **Another Example:**
  - Given FDs:  $A \rightarrow B$ ,  $B \rightarrow C$
  - Keys:  $\{A\}$
  - $A \rightarrow B$  is OK, since  $A$  is a key of  $R$
  - $B \rightarrow C$  is not OK, since  $C$  is NOT in a key of  $R$ , and it is NOT in the left hand side
  - So  $R$  is NOT in 3NF

R	A	B	C

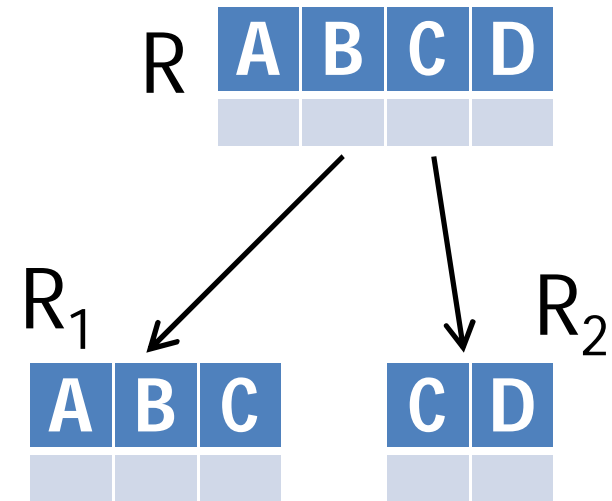
# 3NF Decomposition

- Given: A table NOT in 3NF
- Objective: Decompose it into smaller tables that are in 3NF
- Example
  - Given:  $R(A, B, C, D)$
  - FDs:  $AB \rightarrow C$ ,  $C \rightarrow B$ ,  $A \rightarrow D$
  - Keys:  $\{AB\}$ ,  $\{AC\}$
  - $R$  is not in 3NF, due to  $A \rightarrow D$
  - 3NF decomposition of  $R$ :  
 $R_1(A, B, C)$ ,  $R_2(A, D)$



# 3NF Decomposition Algorithm

- Given: A table R, and a set S of FDs
  - e.g.,  $R(A, B, C, D)$
  - $S = \{A \rightarrow BD, AB \rightarrow C, C \rightarrow D, BC \rightarrow D\}$
- Step 1:** Derive a **minimal basis** of S
  - e.g., a minimal basis of S is  $\{A \rightarrow B, A \rightarrow C, C \rightarrow D\}$
- Step 2:** In the minimal basis, combine the FDs whose left hand sides are the same
  - e.g., after combining  $A \rightarrow B$  and  $A \rightarrow C$ , we have  $\{A \rightarrow BC, C \rightarrow D\}$
- Step 3:** Create a table for each FD remained
  - $R_1(A, B, C), R_2(C, D)$
- Step 4:** If none of the tables contain a key of the original table R, create a table that contains a key of R
- Step 5:** Remove redundant tables



# 3NF Decomposition Algorithm



- Given:
  - Table  $R(A, B, C, D)$
  - A minimal basis  $\{A \rightarrow B, A \rightarrow C, C \rightarrow D\}$
- **Step 1:** Combine those FDs with the same left hand side
  - Result:  $\{A \rightarrow BC, C \rightarrow D\}$
- **Step 2:** For each FD, create a table that contains all attributes in the FD
  - Result:  $R_1(A, B, C), R_2(C, D)$
- **Step 3:** Remove redundant tables (if any)
- Tricky issue: Sometimes we also need to add an additional table (see the next slide)

# 3NF Decomposition Algorithm



- Given:
  - Table  $R(A, B, C, D)$
  - A minimal basis  $\{A \rightarrow B, C \rightarrow D\}$
- **Step 1:** Combine those FDs with the same left hand side
  - Result:  $\{A \rightarrow B, C \rightarrow D\}$
- **Step 2:** For each FD, create a table that contains all attributes in the FD
  - Result:  $R_1(A, B), R_2(C, D)$
- **Step 3:** Remove redundant tables (if any)
- Problem:  $R_1$  and  $R_2$  do not ensure **lossless join**
- Solution: Add a table that contains a key of the original table  $R$
- Key of  $R$ :  $\{A, C\}$
- Additional table to add:  $R_3(A, C)$
- **Final result:**  $R_1(A, B), R_2(C, D), R_3(A, C)$

# 3NF Decomposition Algorithm



- Given:
  - Table  $R(A, B, C, D)$
  - A minimal basis  $\{A \rightarrow B, C \rightarrow D\}$
- **Step 1:** Combine those FDs with the same left hand side
  - Result:  $\{A \rightarrow B, C \rightarrow D\}$
- **Step 2:** For each FD, create a table that contains all attributes in the FD
  - Result:  $R_1(A, B), R_2(C, D)$
- **Step 3:** If no table contain a key of the original table, add a table that contains a key of the original table
  - Result:  $R_1(A, B), R_2(C, D), R_3(A, C)$
- **Step 4:** Remove redundant tables (if any)

# Minimal Basis is not always unique



- For given set of FDs, its minimal basis may not be unique
- Example:
  - Given  $R(A, B, C)$  and  $\{A \rightarrow B, A \rightarrow C, B \rightarrow C, B \rightarrow A, C \rightarrow A, C \rightarrow B\}$
  - Minimal basis 1:  $\{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$
  - Minimal basis 2:  $\{A \rightarrow C, B \rightarrow C, C \rightarrow A, C \rightarrow B\}$
- Different minimal basis may lead to different 3NF decompositions



# Why Does 3NF Preserve All FDs?



- Given: A table  $R$ , and a set  $S$  of FDs
- Step 1: Derive a minimal basis of  $S$
- Step 2: In the minimal basis, combine the FDs whose left hand sides are the same
- Step 3: Create a table for each FD remained
- Step 4: If none of the tables contain a key of the original table  $R$ , create a table that contains a key of  $R$
- Step 5: Remove redundant tables
- Rationale: Because of Step 3

# Questions?



# BCNF (Boyce-Codd Normal Form)

- A table R is in BCNF, if and only if for **every** FD on R,
  - Either the FD is trivial
  - Or the left hand side of the FD is a superkey of R

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

- Key: {NRIC, Phone}
- NRIC  $\rightarrow$  Name, Address
- The FD is non-trivial, and its left hand side is not a superkey
- Therefore, the table is NOT in BCNF

# BCNF Decomposition

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

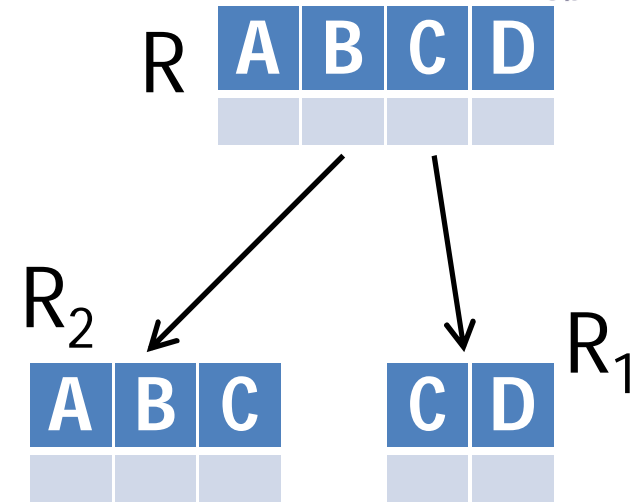
- The above table is not in BCNF
- We transform it into the two tables below, which
  - Provide the same information as the table above
  - Satisfy BCNF

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

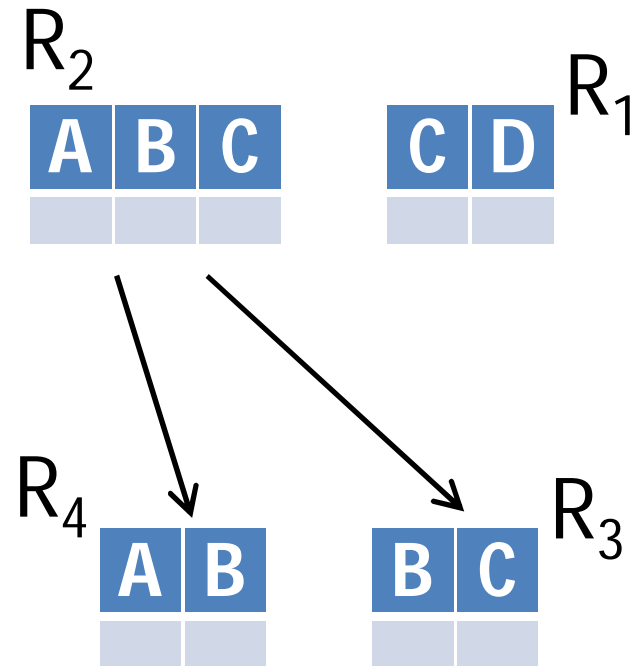
# BCNF Decomposition Algorithm

- Given: Table R, and some FDs on R
  - E.g.,  $B \rightarrow C$ ,  $C \rightarrow D$
- **Step 1:** Find the keys of R
  - Keys: {AB}
- **Step 2:** Find a non-trivial FD whose left hand side (lhs) is not a key
  - $C \rightarrow D$
- **Step 3:** Compute the closure of the lhs
  - $\{C\}^+ = \{C, D\}$
- **Step 4:** Decompose the table into two
  - First one: include all attributes in the closure, i.e., {C, D}
  - Second one: include the lhs and all attributes NOT in the closure, i.e., {A, B, C}
- **Step 5:** Repeat Steps 1-5 on the new tables, until they are all in BCNF



# BCNF Decomposition Algorithm

- Repeat the steps on  $R_2$
- Given FDs:  $B \rightarrow C$ ,  $C \rightarrow D$
- **Step 1:** Find the keys of  $R_2$ 
  - Keys:  $\{AB\}$
- **Step 2:** Find a non-trivial FD whose left hand side (lhs) is not a key
  - $B \rightarrow C$
- **Step 3:** Compute the closure of the lhs on  $R_2$ 
  - $\{B\}^+ = \{B, C\}$
- **Step 4:** Decompose the table into two
  - First one: include all attributes in the closure, i.e.,  $\{B, C\}$
  - Second one: include the lhs and all attributes NOT in the closure, i.e.,  $\{A, B\}$
- **Step 5:** Repeat Steps 1-5 on the new tables, until they are all in BCNF



# BCNF Decomposition Algorithm



- Now we have three tables
- Each of them has only two attributes
- Tables with only two attributes always satisfy BCNF (why?)
- Therefore, we are done
- Final decomposition:
  - $R_1(C, D)$ ,  $R_3(B, C)$ ,  $R_4(A, B)$

C	D

 $R_1$  $R_4$ 

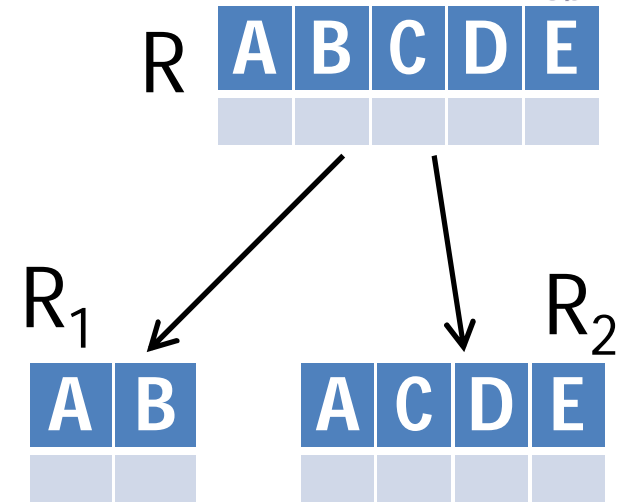
A	B

B	C

 $R_3$

# Tricky Case of BCNF Decomposition

- Given:  $A \rightarrow B$ ,  $BC \rightarrow D$
- **Step 1:** Find the keys of R
  - Keys: {ACE}
- **Step 2:** Find a non-trivial FD whose left hand side (lhs) is not a key
  - $A \rightarrow B$
- **Step 3:** Compute the closure of the lhs
  - $\{A\}^+ = \{A, B\}$
- **Step 4:** Decompose the table into two
  - First one: include all attributes in the closure, i.e., {A, B}
  - Second one: include the lhs and all attributes NOT in the closure, i.e., {A, C, D, E}
- **Step 5:** Repeat Steps 1-5 on the new tables, until they are all in BCNF





# Tricky Case of BCNF Decomposition

- Given:  $A \rightarrow B$ ,  $BC \rightarrow D$
- $R_1$  has only two attribute, so it is in BCNF already
- Repeat the steps on  $R_2$
- **Step 1:** Find the keys of  $R_2$ 
  - ❑ Keys: {ACE}
- **Step 2:** Find a non-trivial FD whose left hand side (lhs) is not a key
  - ❑  $A \rightarrow B$ ?
  - ❑  $BC \rightarrow D$ ?
- Problem here: Some FD contains attributes NOT in  $R_2$
- How to check whether it satisfies BCNF?
- Need to use closures

$R_1$

A	B

$R_2$

A	C	D	E

# Tricky Case of BCNF Decomposition

- Given:  $A \rightarrow B$ ,  $BC \rightarrow D$
- $R_2(A, C, D, E)$
- Keys:  $\{ACE\}$

$R_1$

A	B

$R_2$

A	C	D	E

- Use closures to find non-trivial FDs on  $R_2$ 
  - $\{A\}^+ = \{AB\}$ ,  $\{C\}^+ = \{C\}$ ,  $\{D\}^+ = \{D\}$ ,  $\{E\}^+ = \{E\}$
  - $\{AC\}^+ = \{ACBD\}$ ,
  - This indicates that  $AC \rightarrow D$ , whose lhs is not a superkey of  $R_2$
  - Therefore,  $R_2$  does not satisfy BCNF, and it needs to be decomposed again

# Tricky Case of BCNF Decomposition



- In general, when we check a given  $X \rightarrow Y$  against a table
  - If  $X$  or  $Y$  is not contained in the table
  - Then we need to use closure to check if there is some “hidden” FDs that violates BCNF
- Previous example:
  - $R_2(A, C, D, E)$
  - Given FDs:  $A \rightarrow B, BC \rightarrow D$
  - Keys:  $\{ACE\}$
  - The left hand side of  $BC \rightarrow D$  is  $BC$ , which is not fully contained in  $R_2$
  - Therefore, we need to use closures to check the “hidden” FDs on  $R_2$
  - The check reveals that there is a “hidden”  $AC \rightarrow D$  that violates 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  
(this is called the **lossless join** property)

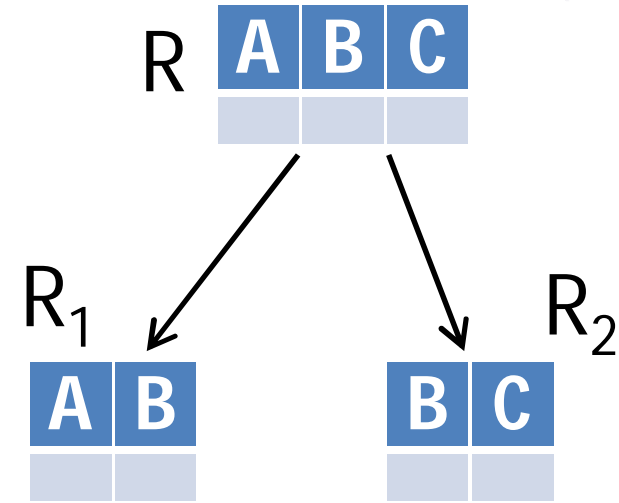
# Properties of BCNF Decomposition



- Good properties
  - No update or deletion anomalies
  - Very small redundancy
  - The original table can always be reconstructed from the decomposed tables  
(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  $AB \rightarrow C, C \rightarrow B$
- Keys:  $\{AB\}, \{AC\}$
- BCNF Decomposition
  - $R_1(B, C)$
  - $R_2(A, B)$
- Non-trivial FDs on  $R_1$ : none
- Non-trivial FDs on  $R_2$ :  $C \rightarrow B$
- The other FD,  $AB \rightarrow C$ , is hold on any individual table, i.e., it is “lost”
- This why we say that a BCNF decomposition does not always **preserve** all FDs

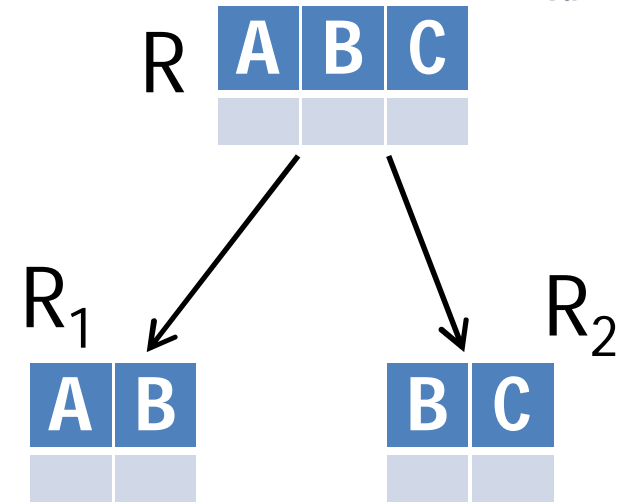


# Dependency Preservation

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

- Previous example

- We have two tables  $R_1(A, B)$ ,  $R_2(B, C)$
- 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



# Summary



- **FD, Minimal Basis**



- **Lossless and dependency-preserving decomposition**



- **Normal forms and decomposition – 1NF, 2NF, 3NF, BCNF**





# Questions ??

---



Thank You !