

# CSCI1270

# Introduction to Database Systems

## Normalization

# Another Use for FD's: Schema Design

## Schema Design: Approach #1

1. Construct E/R diagram
  2. Translate into tables
- Subjective:** How do we know if any good?

## Schema Design: Approach #2

1. Start with universal relation
2. Determine FD's
3. "Decompose" UR using FD's as guide

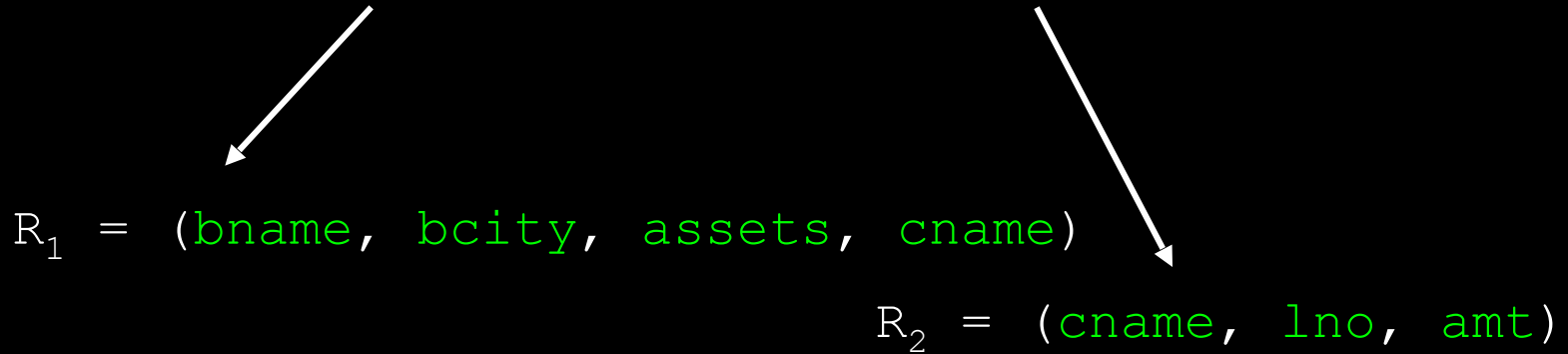
## Schema Design: Approach #3

1. Construct E/R diagram to come up with 1<sup>st</sup> cut design
2. Use FD's to verify or refine

# Decomposition

## 1. Decomposing the Schema

$R = (\text{bname}, \text{bcity}, \text{assets}, \text{cname}, \text{lno}, \text{amt})$



### Notation:

$$R = R_1 \cup R_2$$

# Decomposition

## 2. Decomposing the Instance

R =

bname	bcity	assets	cname	lno	amt
Dntn	Bkln	9M	Jones	L-17	1000
Dntn	Bkln	9M	Johnson	L-23	2000
Mianus	Hnck	1.7M	Jones	L-93	500
Dntn	Bkln	9M	Hayes	L-17	1000

R<sub>1</sub> =

bname	bcity	assets	cname
Dntn	Bkln	9M	Jones
Dntn	Bkln	9M	Johnson
Mianus	Hnck	1.7M	Jones
Dntn	Bkln	9M	Hayes

R<sub>2</sub> =

cname	lno	amt
Jones	L-17	1000
Johnson	L-23	2000
Jones	L-93	500
Hayes	L-17	1000

BTW: Not a Good Decomposition

# Goals of Decomposition

## 1. Lossless Joins

Want to be able to reconstruct big relation by joining smaller ones (Natural join) (i.e.:  $R_1 \bowtie R_2 = R$ ?)

## 2. Dependency Preservation

Want to minimize the cost of global integrity constraints based on FD's (i.e.: Avoid big joins in assertions)

## 3. Redundancy Avoidance

Avoid unnecessary data dupl. (motivation for decomposition)

### Summary:

LJ: Information loss

DP: Efficiency (time)

RA: Efficiency (space), update anomalies

# Another Use for FD's: Schema Design

## Example:

R =

bname	bcity	assets	cname	lno	amt
Dntn	Bkln	9M	Jones	L-17	1000
Dntn	Bkln	9M	Johnson	L-23	2000
Mianus	Hnck	1.7M	Jones	L-93	500
Dntn	Bkln	9M	Hayes	L-17	1000

## R: “Universal Relation”

Tuple meaning: Jones has a loan (L-17) for \$1000 taken out of the Dntn branch in Bkln which has assets of \$9M

**Design:** pro : Fast queries (No need for joins!)  
con : Redundancy, update anomalies, deletion anomalies

# Decomposition Goal #1: Lossless Joins

## A Bad Decomposition

<u>bname</u>	<u>bcity</u>	<u>assets</u>	<u>cname</u>
Dntn	Bkln	9M	Jones
Dntn	Bkln	9M	Johnson
Mianus	Hnck	1.7M	Jones
Dntn	Bkln	9M	Hayes



<u>cname</u>	<u>lno</u>	<u>amt</u>
Jones	L-17	1000
Johnson	L-23	2000
Jones	L-93	500
Hayes	L-17	1000

=

<u>bname</u>	<u>bcity</u>	<u>assets</u>	<u>cname</u>	<u>lno</u>	<u>amt</u>
Dntn	Bkln	9M	Jones	L-17	1000
Dntn	Bkln	9M	Jones	L-93	500
Dntn	Bkln	9M	Johnson	L-23	3000
Mianus	Hnck	1.7M	Jones	L-17	1000
Mianus	Hnck	1.7M	Jones	L-93	500
Dntn	Bkln	9M	Hayes	L-17	1000

# Decomposition Goal #1: Lossless Joins

## A Bad Decomposition

=

bname	bcity	assets	cname	lno	amt
Dntn	Bkln	9M	Jones	L-17	1000
Dntn	Bkln	9M	Jones	L-93	500
Dntn	Bkln	9M	Johnson	L-23	3000
Mianus	Hnck	1.7M	Jones	L-17	1000
Mianus	Hnck	1.7M	Jones	L-93	500
Dntn	Bkln	9M	Hayes	L-17	1000

**Problem:** ⋈ adds meaningless tuples

**“Lossy join”:** By adding noise, have lost meaningful information as a result of decomposition



# Lossless Joins

Is the Following Decomposition Lossless or Lossy?

$R_1 =$

bname	bcity	assets	cname
Dntn	Bkln	9M	Jones
Dntn	Bkln	9M	Johnson
Mianus	Hnck	1.7M	Jones
Dntn	Bkln	9M	Hayes

$R_2 =$

bname	lno	amt
Dntn	L-17	1000
Dntn	L-23	2000
Mianus	L-93	500

**A:** Lossy.

$R_1 \bowtie R_2$  includes:

bname	bcity	assets	cname	lno	amt
...					
Dntn	Bkln	9M	Jones	L-23	2000
Dntn	Bkln	9M	Johnson	L-17	1000
Dntn	Bkln	9M	Hayes	L-23	2000
...					

( $R_1 \bowtie R_2$  has 7  
tuples, whereas R has 4)

# Lossless Joins

Is the Following Decomposition Lossless or Lossy?

$R_1 =$

bname	assets	cname	lno
Dntn	9M	Jones	L-17
Dntn	9M	Johnson	L-23
Mianus	1.7M	Jones	L-93
Dntn	9M	Hayes	L-17

$R_2 =$

lno	bcity	amt
L-17	Bkln	1000
L-23	Bkln	2000
L-93	Hnck	500

**A:** Lossless.  $R_1 \bowtie R_2$  has 4 tuples

# Lossless Joins

## Lossless or Lossy?

$R_1 =$

bname	bcity	assets
Dntn	Bkln	9M
Mianus	Bkln	1.7M

$R_2 =$

bname	lno	amt	cname
Dntn	L-17	1000	Jones
Dntn	L-23	2000	Johnson
Mianus	L-93	500	Jones
Dntn	L-17	1000	Hayes

**A:** Lossless.  $R_1 \bowtie R_2$  has 4 tuples

**Q:** When is decomposition lossless?

# Ensuring Lossless Joins

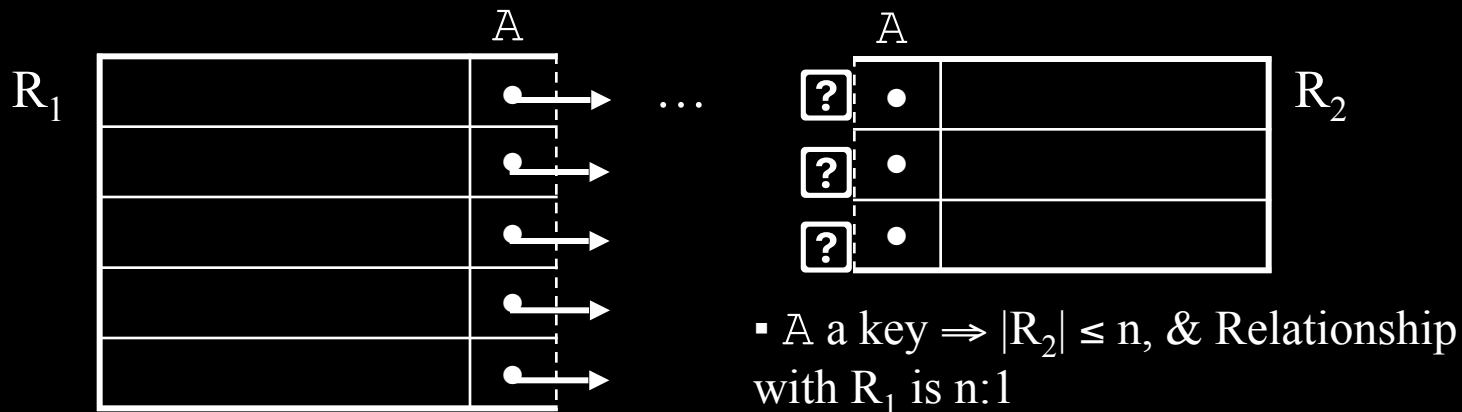
A Decomposition of  $R$ ,  $R = R_1 \cup R_2$  is Lossless iff

$$R_1 \cap R_2 \rightarrow R_1 \text{ or}$$

$$R_1 \cap R_2 \rightarrow R_2$$

(i.e.: Intersecting atts must form a super key for one of the resulting smaller relations)

Intuition: Original relation  $R$  has  $n$  tuples



▪  $A$  not a key  $\Rightarrow |R_1| = n$

▪  $A$  a key  $\Rightarrow |R_2| \leq n$ , & Relationship with  $R_1$  is  $n:1$

**$\therefore n$  tuples in result**

# Decomposition Goal #2: Dependency Preservation

**Goal:** Efficient integrity checks of FD's

**An Example With No Dependency Preservation:**

$R = (\text{bname}, \text{bcity}, \text{assets}, \text{cname}, \text{lno}, \text{amt})$   
 $\text{bname} \rightarrow \text{bcity}$   
 $\text{assets}$   
 $\text{lno} \rightarrow \text{amt}$

**Decomposition:**  $R = R_1 \cup R_2$

$R_1 = (\text{bname}, \text{assets}, \text{cname}, \text{lno})$

$R_2 = (\text{lno}, \text{bcity}, \text{amt})$

**Lossless, but Not DP. Why?**

# Decomposition Goal #2: Dependency Preservation (cont.)

**Decomposition (cont.):**  $R = R_1 \cup R_2$

$R_1 = (\text{bname}, \text{assets}, \text{cname}, \text{lno})$

$R_2 = (\text{lno}, \text{bcity}, \text{amt})$

**Lossless, but Not DP. Why?**

**A:**  $\text{bname} \rightarrow \text{bcity}$  crosses 2 tables

```
CREATE ASSERTION bname-bcity
CHECK NOT EXISTS
  (SELECT *
   FROM R1 AS x1, R2 AS y1, R1 AS x2, R2 AS y2
   WHERE x1.lno = y1.lno AND x2.lno = y2.lno AND
        x1.lno = x2.lno AND x1.bname = x2.bname AND
        y1.bcity <> y2.bcity)
```

# Decomposition Goal #2: Dependency Preservation

## To Ensure Best Possible Efficiency of FD Checks

Ensure that only a SINGLE table be examined for each FD

i.e.: Ensure that  $A_1, \dots, A_n \rightarrow B_1, \dots, B_m$  can be checked by examining one table as in:

$R_i =$

...	$A_1$	...	$A_n$	$B_1$	...	$B_m$	...
	—	—	—	—	—	—	
	—	—	—	—	—	—	
	—	—	—	—	—	—	

**Above:**  $R_i$  “covers” the FD,  $A_1, \dots, A_n \rightarrow B_1, \dots, B_m$

## To Test if Decomposition $R = R_1 \cup \dots \cup R_n$ is DP,

1. See which FD's of  $R$  are covered by  $R_1, \dots, R_n$
2. Compare closure of (1) to closure of FD's of  $R$

# Decomposition Goal #2: Dependency Preservation

## More Formally:

To test if  $R = R_1 \cup \dots \cup R_n$  is dependency preserving wrt  $R$ 's FD set,  $F$ :

1. Compute  $F^+$
2. Compute  $G$

$G \leftarrow \emptyset$

For  $i \leftarrow 1$  to  $n$  DO

    Add to  $G$  those FD's in  $F^+$  covered by  $R_i$

3. Compute  $G^+$

4. If  $F^+ = G^+$ : Decomposition is DP

    If  $F^+ \neq G^+$ : Decomposition is not DP



# Decomposition Goal #2: Dependency Preservation (cont.)

## More Formally (cont.):

To test if  $R = R_1 \cup \dots \cup R_n$  is dependency preserving wrt  $R$ 's FD set,  $F$ :

1. Compute  $F^+$
2. Compute  $G$
3. Compute  $G^+$
4. Compute  $F^+ - G^+$

## Example:

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

$R_1 = (A, B, C); R_2 = (C, D)$

Is this decomposition of  $(A, B, C, D)$  DP?

# Decomposition Goal #2: Dependency Preservation

## Example:

$$F = \{A \rightarrow B, AB \rightarrow D, C \rightarrow D\}$$
$$R_1 = (A, B, C); R_2 = (C, D)$$

Is  $R = R_1 \cup R_2$  DP?

- A:**
1.  $F^+ = \{A \rightarrow B, AB \rightarrow D, C \rightarrow D\}^+$   
Note:  $(A \rightarrow D) \in F^+$
  2.  $G = \emptyset \cup \{A \rightarrow B, \dots\} \cup \{C \rightarrow D, \dots\}$   
Note:  $(A \rightarrow D) \notin G$
  3.  $G^+ = \{\dots\}$   
Note:  $(A \rightarrow D) \notin G^+$
  4.  $F^+ \neq G^+$  because  $(A \rightarrow D) \in (F^+ - G^+)$

**$\therefore$  Decomposition is not DP**

# Decomposition Goal #2: Dependency Preservation

## Example:

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

What is a DP decomposition of  $F$ ?

**A:**  $R = R_1 \cup R_2$  s.t.  $R_1 = (A, B, D)$ ;  $R_2 = (C, D)$

1.  $F^+ = \{A \rightarrow B, AB \rightarrow D, C \rightarrow D\}^+$

2.  $G^+ = \{A \rightarrow B, AB \rightarrow D, C \rightarrow D\}^+$

3.  $F^+ = G^+$

Note:  $G^+$  cannot introduce FD's not in  $F^+$

$\therefore$  Decomposition is DP

**Q:** Does it satisfy lossless joins?

**A:** No

# Decomposition Goals Summary

## Lossless Joins

**Motivation:** Avoid information loss

**Idea:** No noise introduced when reconstitution universal relation via joins

**Test:** At each decomposition test:  $R = R_1 \cup R_2$   
 $(R_1 \cap R_2) \rightarrow R_1$  or  $(R_1 \cap R_2) \rightarrow R_2$

**Ensured for:** BCNF, 3NF

## Dependency Preservation

**Motivation:** Efficient FD assertions

**Idea:** No gic's require joins of more than 1 table with itself

**Test:**  $R = R_1 \cup \dots \cup R_n$  is DP if closure of FD's covered by each  $R_i =$  closure of FD's covered by  $R = F^+$

**Ensured for:** 3NF

# Decomposition Goal #3

## Redundancy Avoidance

A	B	C
a	x	1
e	x	1
g	y	2
h	y	2
m	y	2
n	z	1
p	z	1

### Redundancy:

1. Name FD of this relation?

**Ans:**  $B \rightarrow C$

2. Name the super keys of this relation

**A:** All sets of atts that include A

3. When do we have redundancy?

**A:** When  $\exists$  some FD,  $X \rightarrow Y$  covered by relation & X not a super key

# Decomposition Goals Summary (cont.)

## Redundancy Avoidance

**Motivation:** Avoid update, deletion anomalies

**Idea:** Avoid update anomalies, wasted space

**Test:** For any  $X \rightarrow Y$  covered by  $R_i$ ,

$X$  should be a superkey of  $R_i$

**Ensured for:** BCNF

# Boyce-Codd Normal Form

## What is a Normal Form?

Characterization of schema decomposition in terms of properties it satisfies

## BCNF:

Guarantees no redundancy and lossless joins (Not DP!)

**Defined:** Relation schema  $R$ , with FD set  $F$ , is in BCNF if:

For all nontrivial  $X \rightarrow Y$  in  $F^+$ :  $X \rightarrow R$   
(i.e.:  $X$  is a super key)

# BCNF

## Example:

$$R = (A, B, C)$$
$$F = \{A \rightarrow B, B \rightarrow C\}$$

Is R in BCNF?

**A:** Consider the nontrivial dependencies in  $F^+$ :

1.  $A \rightarrow B, A \rightarrow C$  (A is a key)
2.  $A \rightarrow C, A \rightarrow B$  (A is a key)
3.  $B \not\rightarrow C, B \not\rightarrow A$  (B is not a key)

Therefore, R not in BCNF



# BCNF

## Example:

$$R = R_1 \cup R_2 \quad R_1 = (A, B); \quad R_2 = (B, C)$$

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

Are  $R_1, R_2$  in BCNF?

- A:**
1. Test  $R_1$ :  
 $A \rightarrow B$  covered,  $A \rightarrow R_1$  (all other FD's covered trivial)
  2. Test  $R_2$ :  
 $B \rightarrow C$  covered,  $B \rightarrow R_2$  (all other FD's covered trivial)

$\therefore R_1, R_2$  in BCNF

# BCNF

## Decomposition Algorithm

ALGORITHM BCNF (R: Relation, F: FD set)  
BEGIN

1. Compute  $F^+$
2.  $\text{Result} \leftarrow \{R\}$
3. While some  $R_i \in \text{Result}$  not in BCNF, DO
  - a. Choose  $(X \rightarrow Y) \in F^+$  s.t.  
 $\rightarrow (X \rightarrow Y)$  covered by  $R_i$   
 $\rightarrow X \not\rightarrow R_i$
  - b. Decompose  $R_i$  on  $(X \rightarrow Y)$   
 $R_{i1} \leftarrow X \cup Y$   
 $R_{i2} \leftarrow R_i - Y$
  - c.  $\text{Result} \leftarrow \text{Result} - \{R_i\} \cup \{R_{i1}, R_{i2}\}$
4. Return result

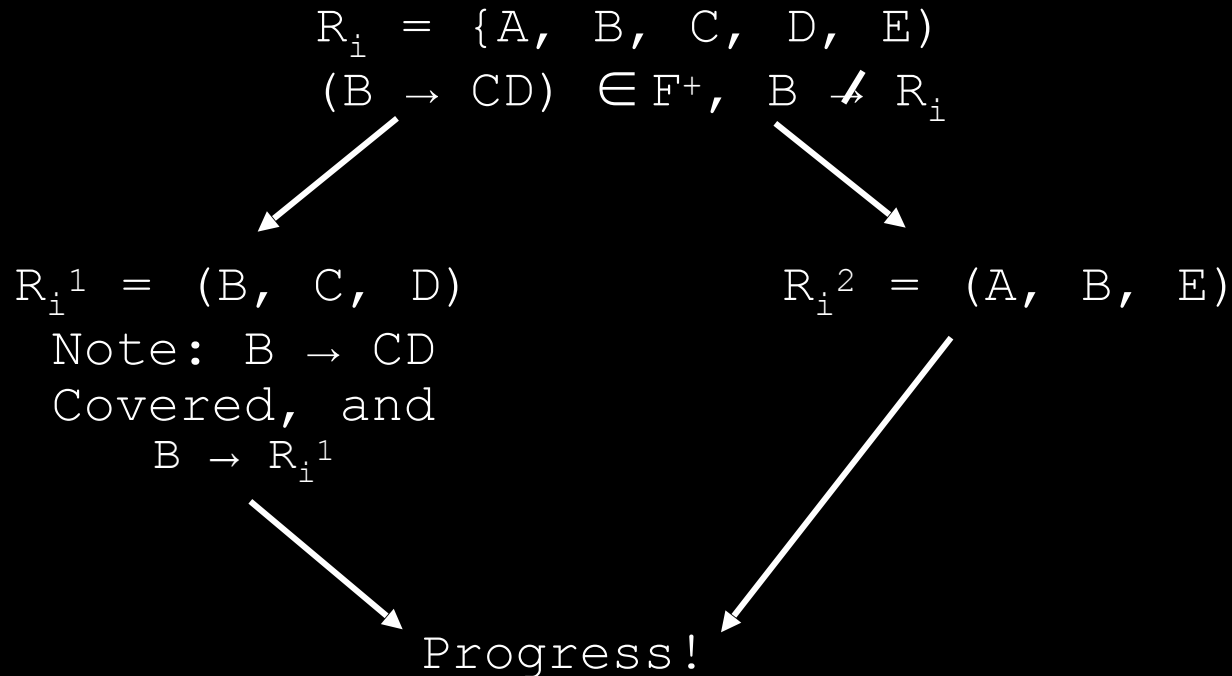
END

# BCNF

## Decomposition Algorithm

Each Step:

Decompose  $R_i$  that is not in BCNF



# BCNF

## Decomposition Algorithm (cont.)

### Example:

$$\begin{aligned} R &= (A, B, C, D) \\ F &= \{A \rightarrow B, AB \rightarrow D, B \rightarrow C\} \end{aligned}$$

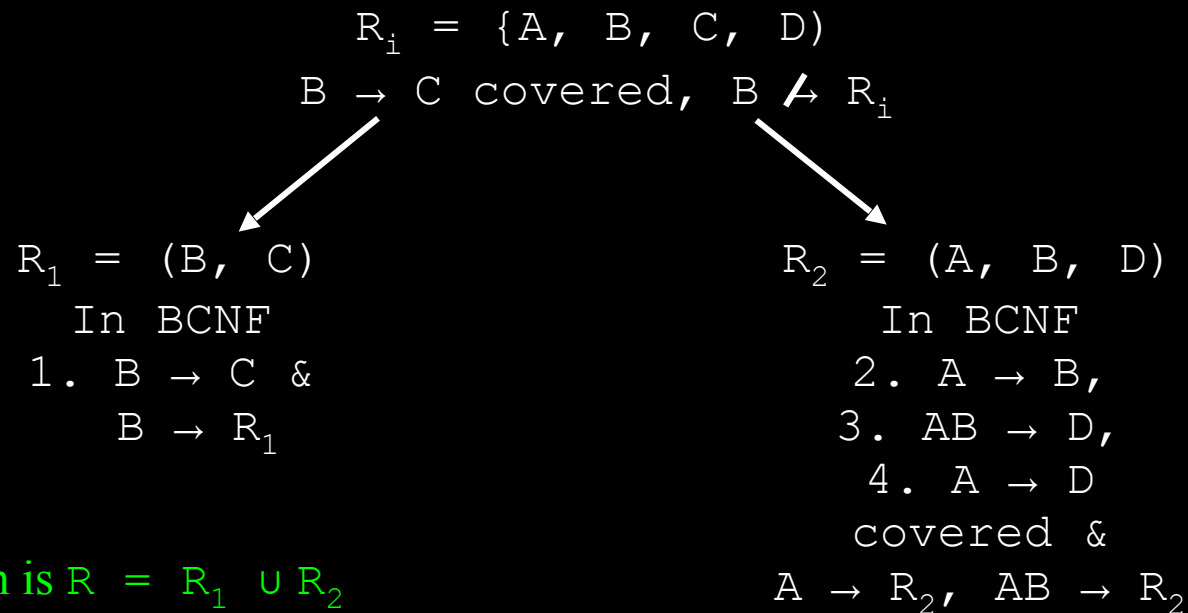
Decompose R into BCNF?

1. Compute  $F^+$ :

$$\begin{aligned} F^+ = \{ & A \rightarrow B, AB \rightarrow D, B \rightarrow C, A \rightarrow C, A \rightarrow D, \\ & AB \rightarrow C, \\ & AC \rightarrow D, \\ & AD \rightarrow C, \\ & ABC \rightarrow D, \\ & ABD \rightarrow C\} + \text{all trivial dep's} \end{aligned}$$

# BCNF

## Decomposition Algorithm (cont.)



# BCNF

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

$F = \{A \rightarrow BC, E \rightarrow HA\}$

## Decompose R into BCNF:

$F^+ = \{A \rightarrow B, A \rightarrow C, A \rightarrow BC$   
 $E \rightarrow H, E \rightarrow A, E \rightarrow HA$   
 $E \rightarrow B, E \rightarrow C, E \rightarrow BC$   
 $E \rightarrow HB, E \rightarrow HC, E \rightarrow AB$   
 $E \rightarrow AC,$   
 $AE \rightarrow \dots,$   
 $ABE \rightarrow \dots,$   
 $ACE \rightarrow \dots,$   
 $ADE \rightarrow \dots,$   
 $\dots\} + \text{all trivial dep's}$

**Note:** This will suffice!

Find 2 decompositions,  
1 DP and 1 not DP

# BCNF Decomposition

$R = (A, B, C, D, E, H)$   
 $F = \{A \rightarrow BC, E \rightarrow HA\}$

(Note:  $F_c = F$ )

**Decomposition #1:**  $R = R_1 \cup R_3 \cup R_4$

$R = (A, B, C, D, E, H)$   
Decompose on  $A \rightarrow BC$

$R_1 = (\underline{A}, B, C)$

$R_2 = (A, D, E, H)$   
Decompose on  $E \rightarrow HA$

$R_3 = (A, \underline{E}, H)$

$R_4 = (\underline{D}, \underline{E})$

**Q:** Is this DP?

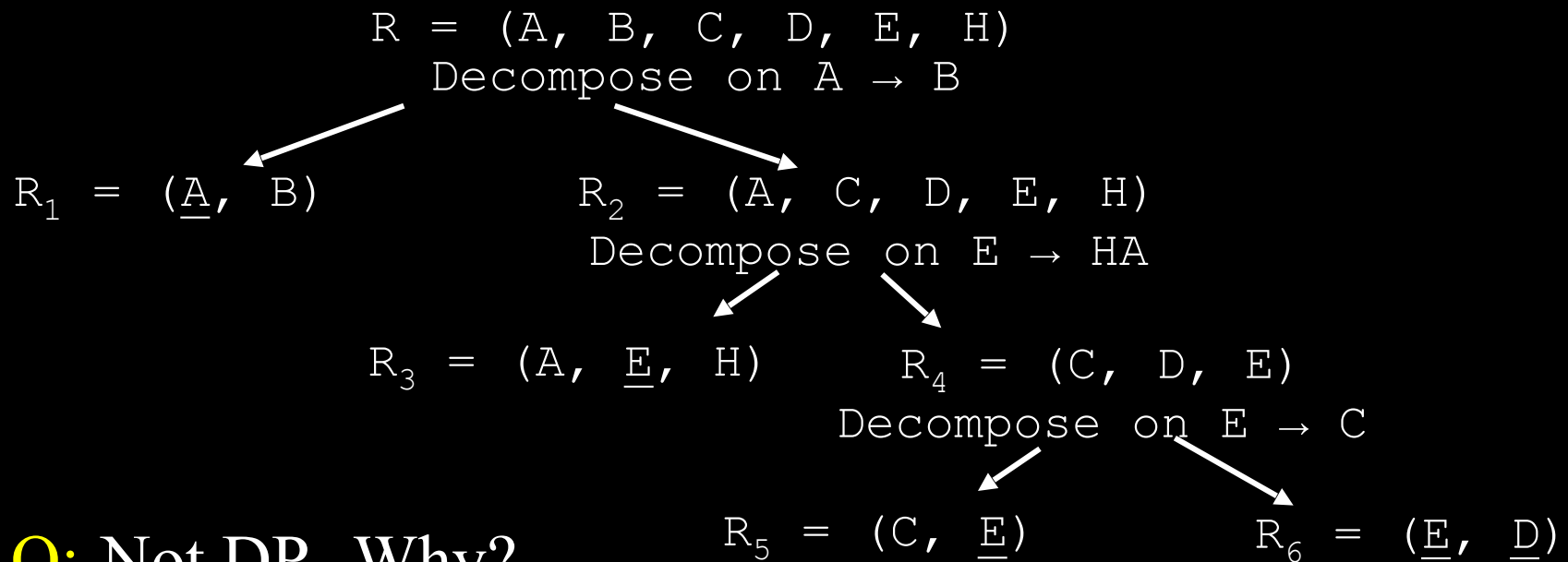
**A:** Yes. All  $F_c$  covered by  $R_1, R_3, R_4$ . Therefore  $F^+$  covered

# BCNF Decomposition (cont.)

$R = (A, B, C, D, E, H)$   
 $F = \{A \rightarrow BC, E \rightarrow HA\}$

(Note:  $F_c = F$ )

**Decomposition #2:**  $R = R_1 \cup R_3 \cup R_5 \cup R_6$



**Q:** Not DP. Why?

**A:**  $A \rightarrow C$  not covered by  $R_1, R_3, R_5, R_6$ .



# More BCNF (cont.)


**Q:** Can we decompose on FD's in  $F_c$  to get a DP BCNF decomposition?

**A:** Sometimes, BCNF + DP not possible

$$R = (J, K, L)$$
$$F = \{JK \rightarrow L, L \rightarrow K\}$$

Decomposition #1:


$R = (J, K, L)$   
Decompose on  $L \rightarrow K$



$$R_1 = (\underline{L}, K) \quad R_2 = (\underline{J}, \underline{L})$$

Not DP:  $JK \rightarrow L$  not covered

Decomposition #2:

$R = (J, K, L)$   
Decompose on  $JK \rightarrow L$


$$R_1 = (\underline{J}, \underline{K}, L) \quad R_2 = (\underline{J}, \underline{L})$$



Still not in BCNF  
( $L$  not a superkey)

# Aside

## Is This a Realistic Example?

$$JK \rightarrow L$$
$$L \rightarrow K$$

A:  $\text{BankerName} \rightarrow \text{BranchName}$   
 $\text{BranchName CustomerName} \rightarrow \text{BankerName}$

Every banker works at one branch

A customer works with the same banker at a given branch

# Testing for FDs Across Relations

- Decomposition not dependency preserving  $\Rightarrow$  an extra **materialized view (MV)** for each dependency  $\alpha \rightarrow \beta$  in  $F_c$  that is not preserved in the decomposition
- The MV is a projection on  $\alpha \beta$  of the join of the relations in the decomposition
- DBMS maintains MV when the relations are updated.
  - $\rightarrow$  *No extra coding effort for programmer.*
- Space overhead: storing MV
- Time overhead: keeping MV up to date

# Multivalued Dependencies

- There are database schemas in BCNF that do not seem to be sufficiently normalized
- Consider a database  
*classes(course, teacher, book)*
- The database lists for each course the set of teachers any one of which can be the course's instructor, and the set of books, all of which are required for the course (no matter who teaches it).

course	teacher	book
database	Avi	DB Concepts
database	Avi	Ullman
database	Hank	DB Concepts
database	Hank	Ullman
database	Sudarshan	DB Concepts
database	Sudarshan	Ullman
operating systems	Avi	OS Concepts
operating systems	Avi	Shaw
operating systems	Jim	OS Concepts
operating systems	Jim	Shaw

classes

*(course, teacher, book)* is the only key, and therefore the relation is in BCNF

Insertion anomalies – i.e., if Sara is a new teacher that can teach database, two tuples need to be inserted

(database, Sara, DB Concepts)

(database, Sara, Ullman)

Therefore, it is better to decompose *classes* into:

course	teacher
database	Avi
database	Hank
database	Sudarshan
operating systems	Avi
operating systems	Jim

teaches

course	book
database	DB Concepts
database	Ullman
operating systems	OS Concepts
operating systems	Shaw

text

We shall see that these two relations are in Fourth Normal Form (4NF)

# Multivalued Dependencies (MVDs)

Let  $R$  be a relation schema and let  $\alpha \subseteq R$  and  $\beta \subseteq R$ . The *multivalued dependency*

$$\alpha \twoheadrightarrow \beta$$

holds on  $R$  if in any legal relation  $r(R)$ , for all pairs of tuples  $t_1$  and  $t_2$  in  $r$  such that  $t_1[\alpha] = t_2[\alpha]$ , there exist tuples  $t_3$  and  $t_4$  in  $r$  such that:

$$t_1[\alpha] = t_2[\alpha] = t_3[\alpha] = t_4[\alpha]$$

$$t_3[\beta] = t_1[\beta]$$

$$t_3[R - \beta] = t_2[R - \beta]$$

$$t_4[\beta] = t_2[\beta]$$

$$t_4[R - \beta] = t_1[R - \beta]$$

# MVD (Cont.)

## Tabular representation of $\alpha \twoheadrightarrow \beta$

	$\alpha$	$\beta$	$R - \alpha - \beta$
$t_1$	$a_1 \dots a_i$	$a_{i+1} \dots a_j$	$a_{j+1} \dots a_n$
$t_2$	$a_1 \dots a_i$	$b_{i+1} \dots b_j$	$b_{j+1} \dots b_n$
$t_3$	$a_1 \dots a_i$	$a_{i+1} \dots a_j$	$b_{j+1} \dots b_n$
$t_4$	$a_1 \dots a_i$	$b_{i+1} \dots b_j$	$a_{j+1} \dots a_n$



# Example

- Let  $R$  be a relation schema with a set of attributes that are partitioned into 3 nonempty subsets.

$Y, Z, W$

- We say that  $Y \twoheadrightarrow Z$  ( $Y$  multidetermines  $Z$ ) if and only if for all possible relations  $r(R)$

$\langle y_1, z_1, w_1 \rangle \in r$  and  $\langle y_1, z_2, w_2 \rangle \in r$

implies

$\langle y_1, z_1, w_2 \rangle \in r$  and  $\langle y_1, z_2, w_1 \rangle \in r$

- Note that since the behavior of  $Z$  and  $W$  are identical it follows that  $Y \twoheadrightarrow Z$  if  $Y \twoheadrightarrow W$

# Example (Cont.)

- In our example:

*course*  $\rightarrow\rightarrow$  *teacher*

*course*  $\rightarrow\rightarrow$  *book*

- The above formalizes the notion that a particular value of  $Y$  (*course*) has associated with it a set of values of  $Z$  (*teacher*) and a set of values of  $W$  (*book*), and these two sets are in some sense *independent* of each other.

## Note:

*If  $Y \rightarrow Z$  then  $Y \rightarrow\rightarrow Z$*

*Indeed we have (in above notation)  $Z_1 = Z_2$*

*The claim follows.*

# Use of Multivalued Dependencies

- We use multivalued dependencies in two ways:
  1. To test relations to determine whether they are legal under a given set of functional and multivalued dependencies
  2. To specify constraints on the set of legal relations. We shall thus concern ourselves only with relations that satisfy a given set of functional and multivalued dependencies.
- If a relation  $r$  fails to satisfy a given multivalued dependency, we can construct a relation  $r'$  that does satisfy the multivalued dependency by adding tuples to  $r$ .

# Fourth Normal Form

- A relation schema  $R$  is in 4NF with respect to a set  $D$  of functional and multivalued dependencies if for all multivalued dependencies in  $D^+$  of the form  $\alpha \twoheadrightarrow \beta$ , where  $\alpha \subseteq R$  and  $\beta \subseteq R$ , at least one of the following hold:
  - $\alpha \twoheadrightarrow \beta$  is trivial (i.e.,  $\beta \subseteq \alpha$  or  $\alpha \cup \beta = R$ )
  - $\alpha$  is a superkey for schema  $R$
- If a relation is in 4NF it is in BCNF