

CS460: Intro to Database Systems

Class 7: *Decomposition &  
Schema Normalization*

Instructor: Manos Athanassoulis

<https://midas.bu.edu/classes/CS460/>

# Review: Database Design

## Requirements Analysis

user needs; what must database do?

## Conceptual Design

high level description (often done w/ ER model)

## Logical Design

translate ER into DBMS data model

## Schema Refinement

**consistency, normalization**

## Physical Design

indexes, disk layout

# Why schema refinement

what is a bad schema?

*a schema with redundancy!*

why?

redundant storage & insert/update/delete anomalies

how to fix it?



***normalize*** the schema by decomposing  
normal forms: BCNF, 3NF, ...

# Motivating Example

SSN	Name	Salary	Telephone
987-00-8761	John	65K	857-555-1234
987-00-8761	John	65K	857-555-8800
123-00-9876	Anna	80K	617-555-9876
787-00-4321	John	25K	617-555-3761

SSN  $\rightarrow$  Name, Salary

# Motivating Example

SSN	Name	Salary	Telephone
987-00-8761	John	65K	857-555-1234
987-00-8761	John	65K	857-555-8800
123-00-9876	Anna	80K	617-555-9876
787-00-4321	John	25K	617-555-3761

SSN → Name, Salary

SSN	Name	Salary
987-00-8761	John	65K
123-00-9876	Anna	80K
787-00-4321	John	25K

SSN	Telephone
987-00-8761	857-555-1234
987-00-8761	857-555-8800
123-00-9876	617-555-9876
787-00-4321	617-555-3761

# Motivating Example 2

name	category	color	price	department
iPhone	smartphone	black	600	phones
Lenovo Yoga	laptop	grey	800	computers
unifi	networking	white	150	computers
unifi	cables	white	10	stationary
OnePlus	smartphone	silver	450	phones

name, category  $\rightarrow$  price, color

category  $\rightarrow$  department

# Motivating Example 2

name	category	color	price	department
iPhone	smartphone	black	600	phones
Lenovo Yoga	laptop	grey	800	computers
unifi	networking	white	150	computers
unifi	cables	white	10	stationary
OnePlus	smartphone	silver	450	phones



name	category	color	price
iPhone	smartphone	black	600
Lenovo Yoga	laptop	grey	800
unifi	networking	white	150
unifi	cables	white	10
OnePlus	smartphone	silver	450

category	department
laptop	computers
networking	computers
cables	stationary
smartphone	phones

## Reminder: Reasoning for FDs

- (1) reflexivity e.g.,  $A, B \rightarrow B$
- (2) augmentation e.g., if  $A \rightarrow B$  then  $A, C \rightarrow B, C$
- (3) transitivity e.g., if  $A \rightarrow B$  and  $B \rightarrow C$  then  $A \rightarrow C$
- (4) union e.g., if  $X \rightarrow Y$  and  $X \rightarrow Z$  then  $X \rightarrow Y, Z$
- (5) decomposition e.g., if  $X \rightarrow Y, Z$  then  $X \rightarrow Y$  and  $X \rightarrow Z$

**FD closure of  $F$ ,  $F^+$ :** is the set of all FDs that are implied by  $F$

**attr. closure of  $X$ :** the set of all attributes that are determined by  $X$

**minimal cover:** subset  $S$  of  $F^+$  such that  $S^+ = F^+$



*“chopping the relation into pieces using FDs”*

## **DECOMPOSITION**

# Decomposition

## Formally

we decompose  $R(A_1, \dots, A_n)$  by creating:

$R_1(B_1, \dots, B_m)$

$R_2(C_1, \dots, C_k)$

where  $\{B_1, \dots, B_m\} \cup \{C_1, \dots, C_k\} = \{A_1, \dots, A_n\}$

the instance of  $R_1$  is the projection of  $R$  onto  $B_1, \dots, B_m$

the instance of  $R_2$  is the projection of  $R$  onto  $C_1, \dots, C_k$

# “Good” Decomposition

- (1) minimize redundancy
- (2) avoid information loss (***lossless-join***)
- (3) preserve FDs (***dependency preserving***)
- (4) ensure good query performance

# Information Loss

SSN	Name	Salary	Telephone
987-00-8761	John	65K	857-555-1234
987-00-8761	John	65K	857-555-8800
123-00-9876	Anna	80K	617-555-9876
787-00-4321	John	25K	617-555-3761



SSN	Name	Salary
987-00-8761	John	65K
123-00-9876	Anna	80K
787-00-4321	John	25K



Decompose into:

$R_1(\text{SSN}, \text{Name}, \text{Salary})$

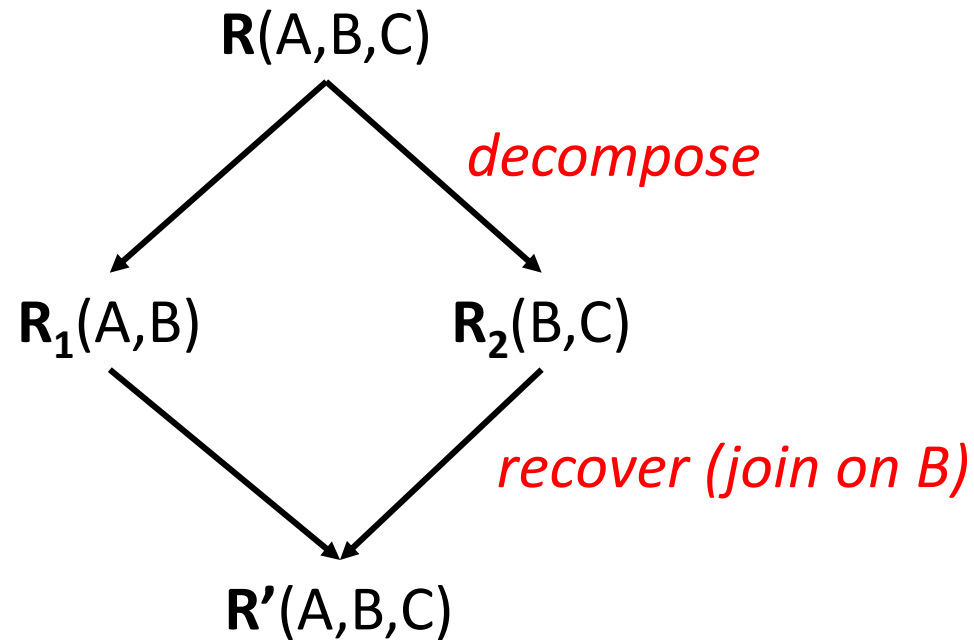
$R_2(\text{Name}, \text{Telephone})$

Name	Telephone
John	857-555-1234
John	857-555-8800
Anna	617-555-9876
John	617-555-3761

can we  
reconstruct R?



# Lossless Decomposition



the decomposition is *lossless-join* if  
for any initial instance  $R$ ,  $R = R'$

# Lossless Criterion

given:

- a relation  $\mathbf{R}(A)$
- a set  $F$  of FDs
- a decomposition of  $\mathbf{R}$  into  $\mathbf{R}_1(A_1)$  and  $\mathbf{R}_2(A_2)$

the decomposition is *lossless-join* **if and only if**  
at least one of the following FDs is in  $F^+$  (closure of  $F$ ):

$$(1) \ A_1 \cap A_2 \rightarrow A_1$$

$$(2) \ A_1 \cap A_2 \rightarrow A_2$$

# Example

Relation  $R(A, B, C, D)$

FD  $A \rightarrow B, C$

$A \rightarrow A \quad A \rightarrow B \quad A \rightarrow C$

$A \rightarrow B, C \quad A \rightarrow A, B \quad A \rightarrow A, C$

$A \rightarrow A, B, C$

what is the  $F^+$ ? 

**lossy**

decomposition into  $R_1(A, B, C)$  and  $R_2(D)$

$A_1 \cap A_2$  empty set

**lossless-join**



decomposition into  $R_1(A, B, C)$  and  $R_2(A, D)$

$A_1 \cap A_2 = A$  and  $A_1 = A, B, C$

$A \rightarrow A, B, C$  is in  $F^+$

# Dependency Preserving

given  $\mathbf{R}$  and a set of FDs  $F$ , we decompose  $\mathbf{R}$  into  $\mathbf{R}_1$  and  $\mathbf{R}_2$ . Suppose:

$\mathbf{R}_1$  has a set of FDs  $F_1$

$\mathbf{R}_2$  has a set of FDs  $F_2$

$F_1$  and  $F_2$  are computed from  $F$

it is dependency preserving if by enforcing  $F_1$  over  $\mathbf{R}_1$  and  $F_2$  over  $\mathbf{R}_2$ , we can enforce  $F$  over  $\mathbf{R}$



# (Good) Example

**Person** (SSN, name, age, canDrink)

$SSN \rightarrow name, age$

$age \rightarrow canDrink$

what is a dependency preserving decomposition?



$R_1(SSN, name, age)$       and       $R_2(age, canDrink)$

$SSN \rightarrow name, age$                        $age \rightarrow canDrink$

Is it also lossless-join?



**Yes!**  $A_1 \cap A_2 = age$  and  $A_2 = age, canDrink$

$age \rightarrow age, canDrink$  is in  $F^+$

# (Bad) Example

**R (A, B, C)**

$A \rightarrow B$

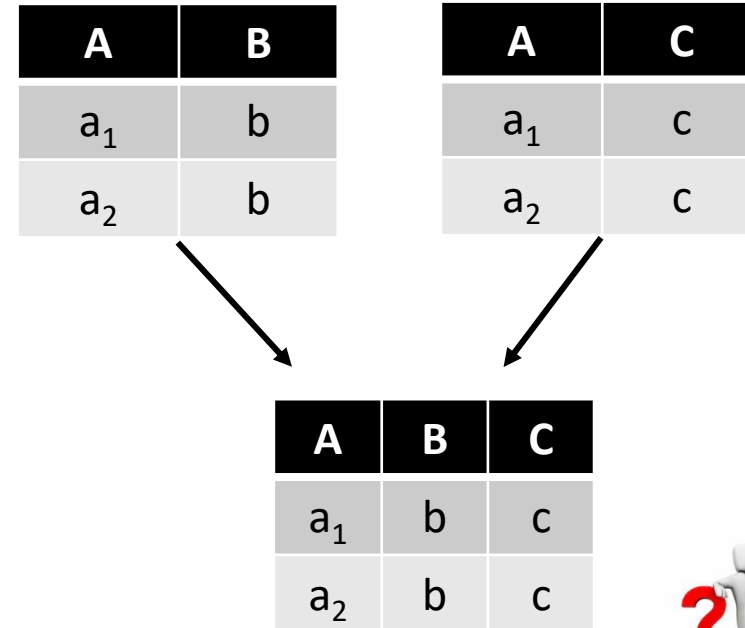
$B, C \rightarrow A$

*not dependency preserving*

**R<sub>1</sub>(A, B)**      and      **R<sub>2</sub>(A, C)**

$A \rightarrow B$

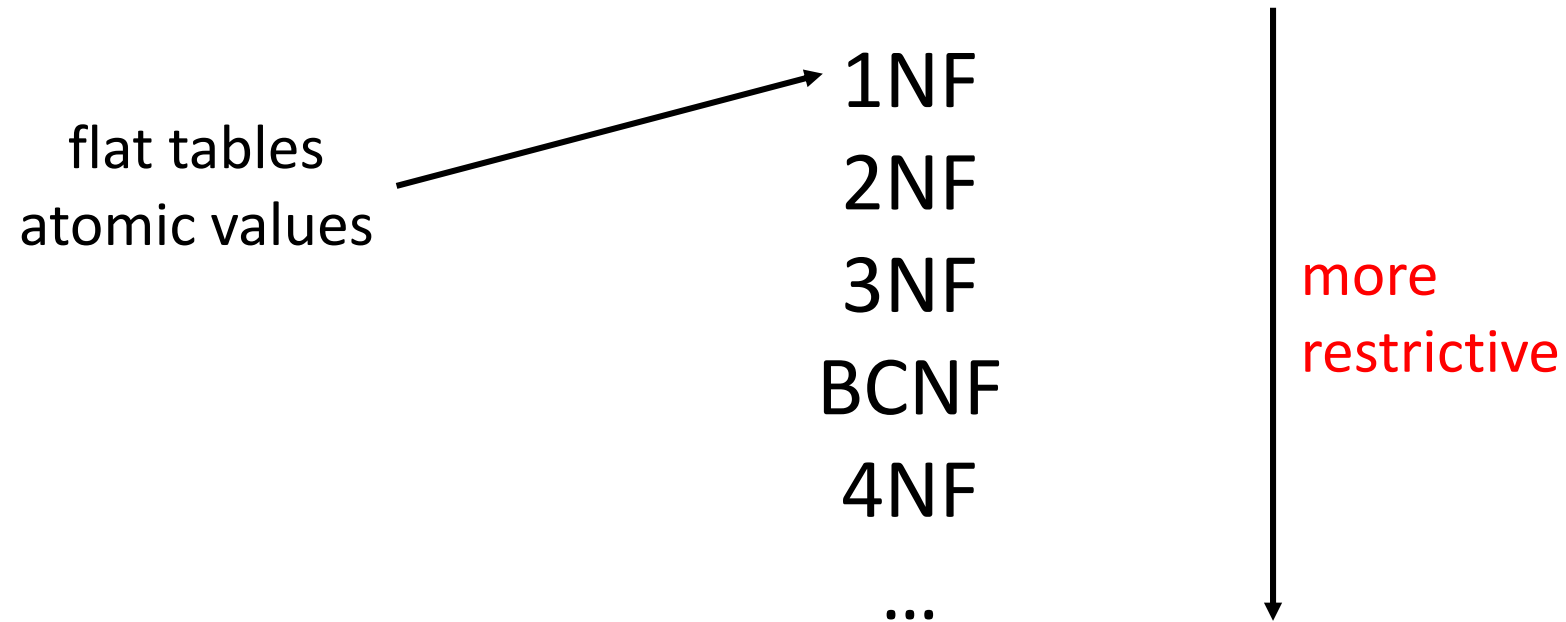
*no FDs!*



the table violates  
 $B, C \rightarrow A$

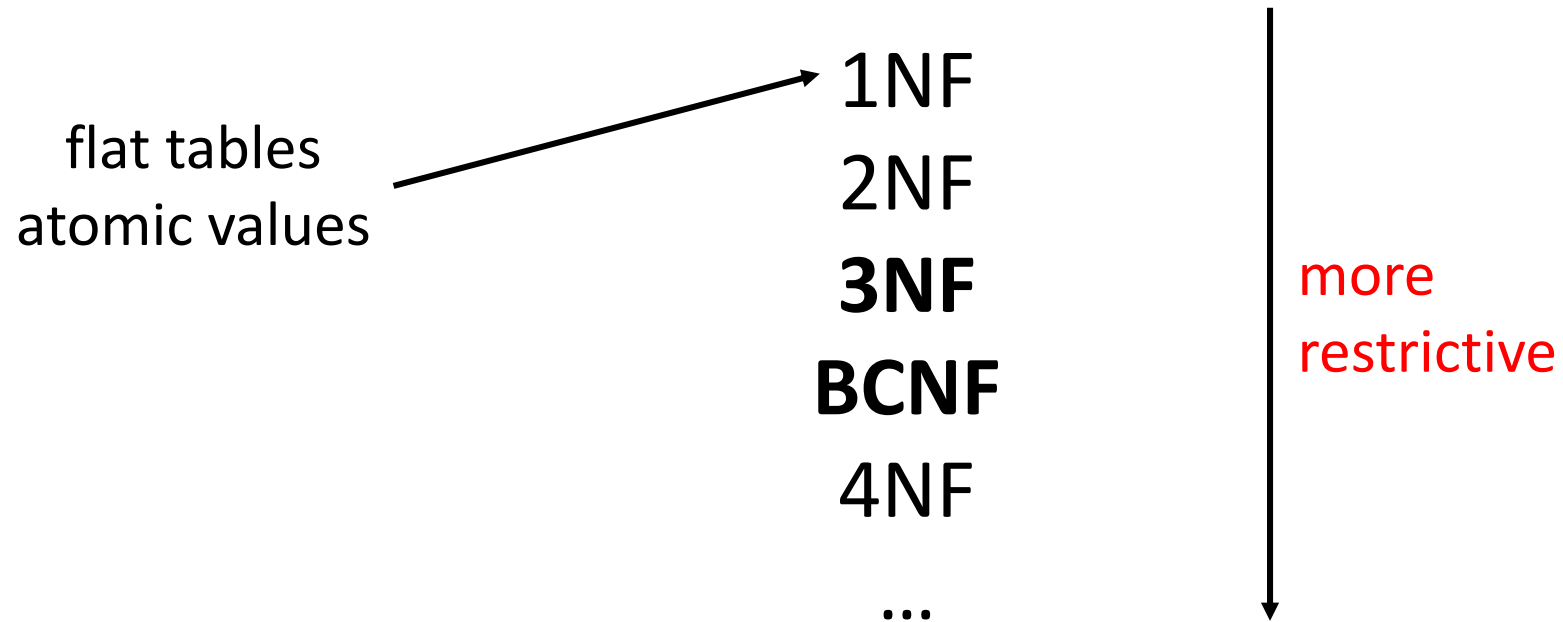
# Normal Forms

How “good” is a schema design?  
follows normal forms



# Normal Forms

How “good” is a schema design?  
follows normal forms



# Boyce-Codd Normal Form (BCNF)

given a relation  $R(A_1, \dots, A_n)$ ,

a set of FDs  $F$ , and  $X \subseteq \{A_1, \dots, A_n\}$

$R$  is in BCNF if  $\forall X \rightarrow A$  one of the two holds:

- $A \in X$  (that is, it is a trivial FD)
- $X$  is a superkey

in other words:  $\forall$  ***non-trivial FD***  $X \rightarrow A$ ,  ***$X$  is a superkey in  $R$***

# BCNF - Example

SSN	Name	Salary	Telephone
987-00-8761	John	65K	857-555-1234
987-00-8761	John	65K	857-555-8800
123-00-9876	Anna	80K	617-555-9876
787-00-4321	John	25K	617-555-3761

*SSN  $\rightarrow$  Name, Salary*

*key: {SSN, Telephone}*

*FD is not trivial!*

*so, is SSN a superkey?*

*no! it is **not** in **BCNF***



# BCNF - Example 2

SSN	Name	Salary
987-00-8761	John	65K
123-00-9876	Anna	80K
787-00-4321	John	25K

*SSN  $\rightarrow$  Name, Salary*

*key: {SSN}*

*FD is not trivial!*

*so, is SSN a superkey?*

*yes! it is in **BCNF***



# BCNF - Example 3

SSN	Telephone
987-00-8761	857-555-1234
987-00-8761	857-555-8800
123-00-9876	617-555-9876
787-00-4321	617-555-3761

*key: {SSN, Telephone}    the relation is in **BCNF***

*why?*



*Is it possible a binary relation  
to not be in **BCNF**?*

*no FDs*





# Binary Relations always BCNF

$R(A, B)$

excluding all trivial FDs, there are three cases:

- (1)  $R$  has no FD
- (2)  $R$  has one FD, either  $A \rightarrow B$  or  $B \rightarrow A$ , or,
- (3)  $R$  has two FDs,  $A \rightarrow B$  and  $B \rightarrow A$



- (1) trivially in BCNF
- (2) in either LHS is the key (hence, superkey)
- (3) both,  $A$  and  $B$  candidate keys

# BCNF Decomposition Algorithm

(1) find a FD that violates BCNF:

$$A_1, \dots, A_n \rightarrow B_1, \dots, B_m$$

(2) decompose **R** to **R<sub>1</sub>** and **R<sub>2</sub>**

$$\begin{aligned} &\mathbf{R_1}(A_1, \dots, A_n, B_1, \dots, B_m) \\ &\mathbf{R_2}(A_1, \dots, A_n, \text{all other attributes}) \end{aligned}$$

(3) repeat until no BCNF violations are left  
(in new tables as well)

# Our favorite example!

SSN	Name	Salary	Telephone
987-00-8761	John	65K	857-555-1234
987-00-8761	John	65K	857-555-8800
123-00-9876	Anna	80K	617-555-9876
787-00-4321	John	25K	617-555-3761

*SSN*  $\rightarrow$  *Name, Salary* violates BCNF

$A_1 = \text{SSN}, B_1 = \text{Name}, B_2 = \text{Salary}$

Split in two relations:

$R_1(\text{SSN}, \text{Name}, \text{Salary})$

$R_2(\text{SSN}, \text{Telephone})$

# Our favorite example!

SSN	Name	Salary	Telephone
987-00-8761	John	65K	857-555-1234
987-00-8761	John	65K	857-555-8800
123-00-9876	Anna	80K	617-555-9876
787-00-4321	John	25K	617-555-3761



SSN	Name	Salary
987-00-8761	John	65K
123-00-9876	Anna	80K
787-00-4321	John	25K

SSN	Telephone
987-00-8761	857-555-1234
987-00-8761	857-555-8800
123-00-9876	617-555-9876
787-00-4321	617-555-3761

# BCNF Decomposition Properties

removes [certain types of] redundancy

is **lossless-join**

is not always dependency preserving

# BCNF – Lossless Join

## Example

**R** (A, B, C) and FD:  $A \rightarrow B$

superkey(s) of the relation?

$\{A, C\}^+, \{A, B, C\}^+ = \{A, B, C\}$

$A \rightarrow B$  violates BCNF (A is not a superkey)



so, the BCNF decomposition is :



**R<sub>1</sub>**(A, B) and **R<sub>2</sub>**(A, C)

we can reconstruct it!

# BCNF – **not** dependency preserving

## Example

$R(A, B, C)$ , FDs:  $A \rightarrow B$  and  $B, C \rightarrow A$

superkey(s) of the relation?

$\{A, C\}^+, \{B, C\}^+, \{A, B, C\}^+ = \{A, B, C\}$

*$B, C \rightarrow A$  is ok, but  $A \rightarrow B$  violates BCNF*



so, the BCNF decomposition is :

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

*$A \rightarrow B$  is preserved in  $R_1$*

*$B, C \rightarrow A$  is not preserved!*

# BCNF Decomposition Examples

**Books** (author, gender, booktitle, genre, price)

*author*  $\rightarrow$  *gender*

*booktitle*  $\rightarrow$  *genre, price*

candidate key(s)?

{author, booktitle} is the only one



Is it in BCNF?



**No**, because LHS of both FD are not a superkey!



# BCNF Decomposition Examples

**Books** (author, gender, booktitle, genre, price)

*author*  $\rightarrow$  *gender*

*booktitle*  $\rightarrow$  *genre, price*

Splitting using: *author*  $\rightarrow$  *gender*



**AuthorInfo** (author, gender)

*FD author*  $\rightarrow$  *gender* (in BCNF!)

**Book2** (author, booktitle, genre, price)

*FD booktitle*  $\rightarrow$  *genre, price*

is booktitle a superkey?



No! {booktitle, author} is.  
So not in BCNF!

# BCNF Decomposition Examples

**Books** (author, gender, booktitle, genre, price)

*author*  $\rightarrow$  *gender*

*booktitle*  $\rightarrow$  *genre, price*

**AuthorInfo** (author, gender)

Further splitting with *booktitle*  $\rightarrow$  *genre, price*



~~**Book2** (author, booktitle, genre, price)~~

**BookAuthor** (booktitle, author)      binary is in BCNF!

**BookInfo** (booktitle, genre, price)      in BCNF!

*FD booktitle*  $\rightarrow$  *genre, price*

is booktitle a superkey?



Yes!

# what if not dependency preserving?

in some cases BCNF decomposition is not dependency preserving

how to address this?



relax the normalization requirements

# Third Normal Form (3NF)

given a relation  $\mathbf{R} (A_1, \dots, A_n)$ ,

a set of FDs  $F$ , and  $X \subseteq \{A_1, \dots, A_n\}$

$\mathbf{R}$  is in 3NF if  $\forall X \rightarrow A$  one of the three holds:

- $A \in X$  (that is, it is a trivial FD)
- $X$  is a superkey
- $A$  is part of some key for  $\mathbf{R}$

is a relation in 3NF also in BCNF?



*No, but a relation in BCNF is always in 3NF!*

# Third Normal Form (3NF)

## Example

**R** (A, B, C), FDs  $C \rightarrow A$  and  $A, B \rightarrow C$   
is in 3NF but not in BCNF. Why?

superkeys?

{A, B}, {B, C}, and {A, B, C}



candidate keys?

{A, B} and {B, C}



**Compromise:** aim for BCNF but settle for 3NF  
lossless-join & dependency preserving possible

# 3NF Algorithm

(1) apply BCNF until all relations are in 3NF

(2) compute a minimal cover  $F'$  of  $F$

(3) for each non-preserved FD  $X \rightarrow A$  in  $F'$   
add a new relation **R** ( $X, A$ )

# 3NF algorithm example

Assume **R** (A, B, C, D)

$A \rightarrow D$

$A, B \rightarrow C$

$A, D \rightarrow C$

$B \rightarrow C$

$D \rightarrow A, B$

superkeys?



$\{A\}$   $\{D\}$   $\{A, B\}$   $\{A, D\}$ , ...

not  $\{B\}$

**Step 1:** find a BCNF decomposition

**R**<sub>1</sub> (B, C)

**R**<sub>2</sub> (A, B, D)

# 3NF algorithm example

Assume **R** (A, B, C, D)

$$A \rightarrow D$$

$$A, B \rightarrow C$$

$$A, D \rightarrow C$$

$$B \rightarrow C$$

$$D \rightarrow A, B$$

**Step 2:** find a minimal cover

$$A \rightarrow D$$

$$B \rightarrow C$$

$$D \rightarrow A$$

$$D \rightarrow B$$



# 3NF algorithm example

Assume **R** (A, B, C, D)

$A \rightarrow D$

$A, B \rightarrow C$

$A, D \rightarrow C$

$B \rightarrow C$

$D \rightarrow A, B$

**Step 3:** add a new relation for not preserved FDs

$A \rightarrow D$

$B \rightarrow C$

$D \rightarrow A$

$D \rightarrow B$

$R_1 (B, C)$

$R_2 (A, B, D)$

all FD are preserved!

both are in BCNF!

# Is Normalization Always Good?

**Example 1:** suppose A and B are always used together, but normalization puts them in different tables (e.g., hours\_worked and hourly\_rate)

decomposition might produce unacceptable performance loss

**Example 2:** data warehouses  
huge historical DBs, rarely updated after creation  
joins expensive or impractical  
[we want “flat” tables, a.k.a, denormalized]

# Example

$R(C, S, J, D, P, Q, V)$

$C \rightarrow S, J, D, P, Q, V$

$J, P \rightarrow C$

$S, D \rightarrow P$

$J \rightarrow S$

**Step 1:**

$R_1(S, D, P)$

$R_2(C, S, J, D, Q, V)$

superkeys?



$\{C\}, \{J, P\}, \{D, J\}, \dots$

not  $\{S, D\}$

# Example

$R (C, S, J, D, P, Q, V)$

$C \rightarrow S, J, D, P, Q, V$

$J, P \rightarrow C$

$S, D \rightarrow P$

$J \rightarrow S$

**Step 1b:**

$R_1 (S, D, P)$

$R_2, (J, S)$

$R_3 (C, J, D, Q, V)$

superkeys of  $R_2 (C, S, J, D, Q, V)$  ?  
 $\{C\}$ , ... not  $\{J\}$



# Example

$R (C, S, J, D, P, Q, V)$

$C \rightarrow S, J, D, P, Q, V$

$J, P \rightarrow C$

$S, D \rightarrow P$

$J \rightarrow S$

## Step 2: Minimal Cover

$C \rightarrow J, C \rightarrow D, C \rightarrow Q, C \rightarrow V$

$J, P \rightarrow C$

$S, D \rightarrow P$

$J \rightarrow S$

$R_1 (S, D, P)$

$R_2 (J, S)$

$R_3 (C, J, D, Q, V)$

$R_4 (J, P, C)$



are they all preserved?

**No!**

**Step 3:** need to add  $R_4 (J, P, C)$

# Example

$R (C, S, J, D, P, Q, V)$

$C \rightarrow S, J, D, P, Q, V$

$J, P \rightarrow C$

$S, D \rightarrow P$

$J \rightarrow S$

## Step 2: Minimal Cover

$C \rightarrow J, C \rightarrow D, C \rightarrow Q, C \rightarrow V$

$J, P \rightarrow C$

$S, D \rightarrow P$

$J \rightarrow S$

$R_1 (S, D, P)$

$R_2 (J, S)$

$R_3 (C, J, D, Q, V)$

$R_4 (J, P, C)$



are they all preserved?

**No!**

**Step 3:** need to add  $R_4 (J, P, C)$

*did we just introduce redundancy?*



# Lesson!

theory of normalization is a guide

cannot always give a “perfect” solution

redundancy

alternatives

query performance

# Summary

fix bad schemas (redundancy) by decomposition

lossless-join

dependency preserving

Desired normal forms

**BCNF:** only superkey FDs

**3NF:** superkey FDs + dependencies to prime attributes in RHS

**Next:** execution of queries