

CS 6400 A

Database Systems Concepts and Design

Lecture 6
09/09/24

Logistics

Assignment 1 due Wednesday (9/11) at 11:59PM

Reminder about late policy:

- One automatic late day granted for this semester
- 10% deduction for each additional day (24 hrs or part of 24 hrs) after the due date
- Assignments can be submitted up to 72 hours past the deadline.

Agenda

1. Boyce-Codd Normal Form
2. Properties of Decomposition
3. 3NF
4. MVDs

Back to Conceptual Design

Now that we know how to find FDs, it's a straight-forward process:

1. Search for “bad” FDs
2. If there are any, then *keep decomposing the table into sub-tables* until no more bad FDs
3. When done, the database schema is *normalized*

Recall: there are several normal forms...

Normal Forms

1st Normal Form (1NF) = All tables are flat

2nd Normal Form = disused

Boyce-Codd Normal Form (BCNF)

3rd Normal Form (3NF)

DB designs based
on functional
dependencies,
intended to prevent
data anomalies

Our focus
in this
lecture

4th and 5th Normal Forms = see text books

1. BCNF

Boyce-Codd Normal Form (BCNF)

Main idea is that we define “good” and “bad” FDs as follows:

- $X \rightarrow A$ is a “*good FD*” if X is a (*super*)key
 - In other words, if A is the set of all attributes
- $X \rightarrow A$ is a “*bad FD*” otherwise

We will try to eliminate the “bad” FDs!

Boyce-Codd Normal Form (BCNF)

Why does this definition of “good” and “bad” FDs make sense?

If X is *not* a (super)key, it functionally determines *some* of the attributes; therefore, those other attributes can be duplicated

- Recall: this means there is redundancy
- And redundancy like this can lead to data anomalies!

EmpID	Name	Phone	Position
E0045	Smith	1234	Clerk
E3542	Mike	9876	Salesrep
E1111	Smith	9876	Salesrep
E9999	Mary	1234	Lawyer

Boyce-Codd Normal Form

BCNF is a simple condition for removing anomalies from relations:

A relation R is in BCNF if:

if $\{A_1, \dots, A_n\} \rightarrow B$ is a non-trivial FD in R

then $\{A_1, \dots, A_n\}$ is a superkey for R

Equivalently: \forall sets of attributes X, either $(X^+ = X)$ or $(X^+ = \text{all attributes})$

In other words: there are no “bad” FDs

Example

Name	SSN	PhoneNumber	City
Fred	123-45-6789	206-555-1234	Seattle
Fred	123-45-6789	206-555-6543	Seattle
Joe	987-65-4321	908-555-2121	Westfield
Joe	987-65-4321	908-555-1234	Westfield

SSN \rightarrow Name, City

This FD is bad because it is not a superkey

\Rightarrow Not in BCNF

What is the key?
{SSN, PhoneNumber}

Example

Name	<u>SSN</u>	City
Fred	123-45-6789	Seattle
Joe	987-65-4321	Madison

<u>SSN</u>	<u>PhoneNumber</u>
123-45-6789	206-555-1234
123-45-6789	206-555-6543
987-65-4321	908-555-2121
987-65-4321	908-555-1234

SSN \rightarrow Name, City

This FD is now good because it is the key

Let's check anomalies:

- Redundancy ?
- Update ?
- Delete ?

Now in BCNF!

Boyce-Codd Normal Form (BCNF)

Any **two-attribute** relation is in BCNF

- If there are no nontrivial FDs, BCNF holds
- If $A \rightarrow B$ holds, but not $B \rightarrow A$, the only nontrivial FD has A (i.e., the key) on the left
- Symmetric case when $B \rightarrow A$ holds, but not $A \rightarrow B$
- If both $A \rightarrow B$ and $B \rightarrow A$ hold, any nontrivial FD has A or B (both are keys) on the left

Employee(empId, ssn)

empld \rightarrow ssn
ssn \rightarrow empld

BCNF Decomposition Algorithm

BCNFDecomp(R):

- Find an FD $X \rightarrow Y$ that violates BCNF
(X and Y are sets of attributes)
- Compute the closure X^+
- let $Y = X^+ - X$, $Z = (X^+)^c$

Let Y be the attributes that X functionally determines (+ that are not in X)

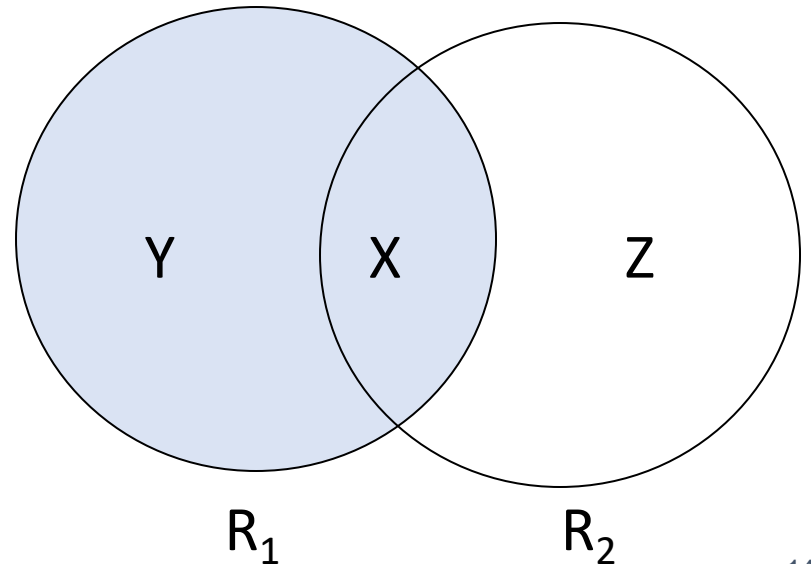
And let Z be the complement, the other attributes that it doesn't

BCNF Decomposition Algorithm

BCNFDecomp(R):

- Find an FD $X \rightarrow Y$ that violates BCNF
(X and Y are sets of attributes)
- Compute the closure X^+
- let $Y = X^+ - X$, $Z = (X^+)^c$
decompose R into $R_1(X \cup Y)$ and $R_2(X \cup Z)$

Split into one relation (table) with X plus the attributes that X determines (Y)...

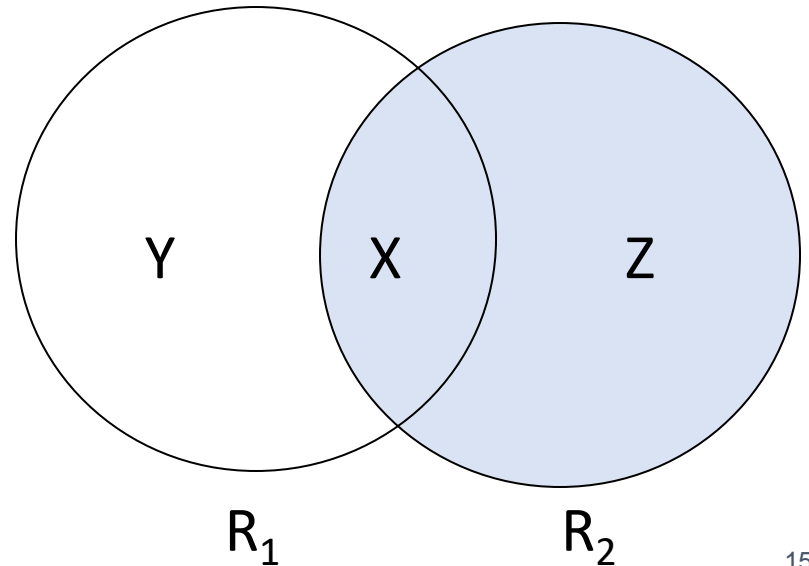


BCNF Decomposition Algorithm

BCNFDecomp(R):

- Find an FD $X \rightarrow Y$ that violates BCNF
(X and Y are sets of attributes)
- Compute the closure X^+
- let $Y = X^+ - X$, $Z = (X^+)^c$
decompose R into $R_1(X \cup Y)$ and $R_2(X \cup Z)$
- Recursively decompose R_1 and R_2

And one relation with X plus
the attributes it does not
determine (Z)



Note: Projection of FDs

- Given a relation R with a set of FD's S , what FD's hold for $R_1 = \pi_L(R)$?
- Compute all the FD's that
 - follow from S and
 - involve only attributes in R_1

Example

- Suppose $R(A, B, C, D)$ has FD's $A \rightarrow B, B \rightarrow C, C \rightarrow D$
- Then the FD's for $R_1(A, C, D)$ are $A \rightarrow C, C \rightarrow D$

Example: BCNF Decomposition

- In general, there can be multiple decompositions

`R(title, year, studioName, president, presAddr)`

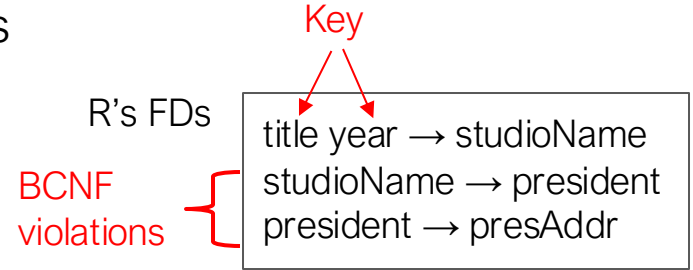
R's FDs

title year → studioName
studioName → president
president → presAddr

Example: BCNF Decomposition

- In general, there can be multiple decompositions

`R(title, year, studioName, president, presAddr)`



Example: BCNF Decomposition

- In general, there can be multiple decompositions

$R(\text{title}, \text{year}, \text{studioName}, \text{president}, \text{presAddr})$



$R1(\text{studioName}, \text{president}, \text{presAddr})$

$R2(\text{title}, \text{year}, \text{studioName})$

R's FDs

BCNF
violations

Key

$\text{title year} \rightarrow \text{studioName}$
 $\text{studioName} \rightarrow \text{president}$
 $\text{president} \rightarrow \text{presAddr}$

Example: BCNF Decomposition

- In general, there can be multiple decompositions

R(title, year, studioName, president, presAddr)



R1(studioName, president, presAddr)

R2(title, year, studioName)

R1's FDs

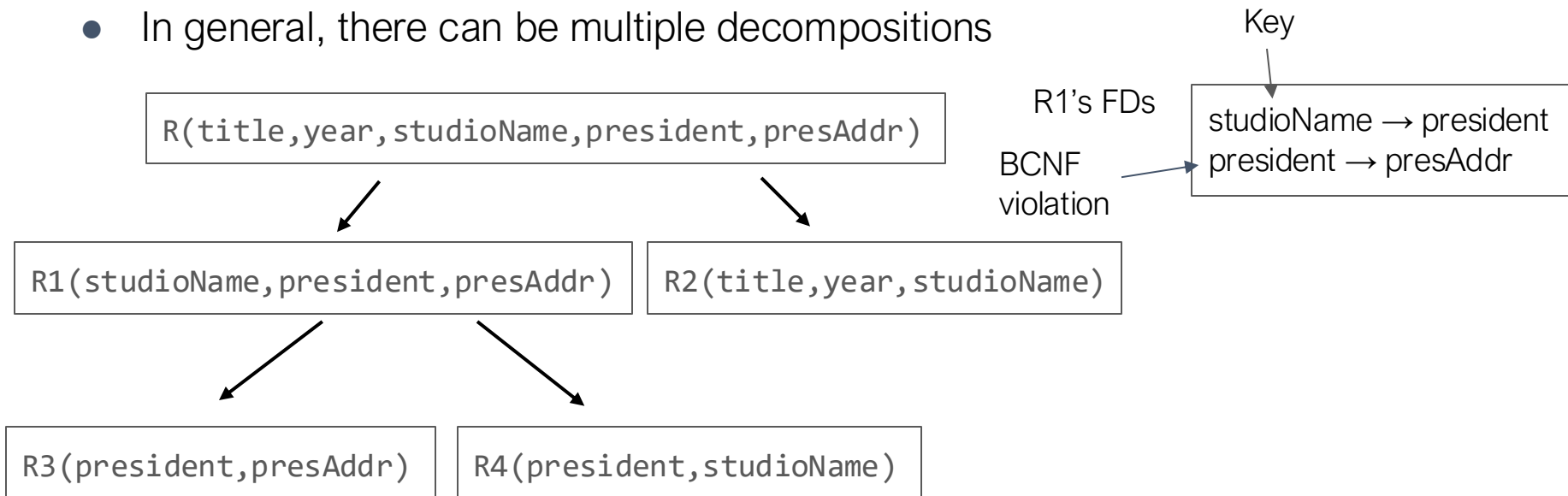
BCNF violation

Key

studioName → president
president → presAddr

Example: BCNF Decomposition

- In general, there can be multiple decompositions



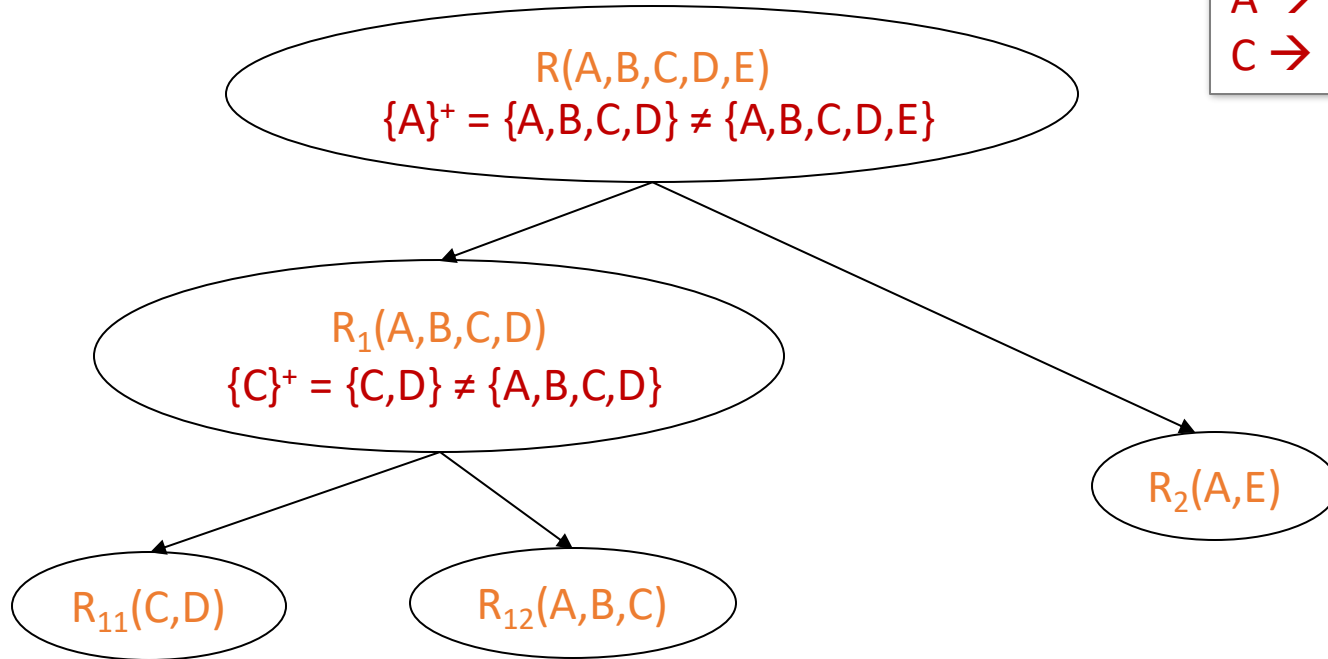
Q: Is this algorithm guaranteed to terminate successfully?

Exercise #1

Decompose into relations satisfying BCNF

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

$A \rightarrow BC$
 $C \rightarrow D$



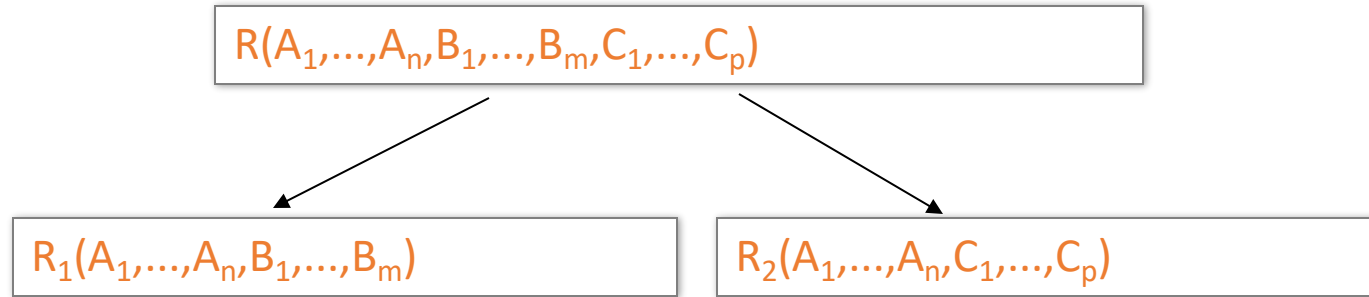
2. Properties of Decomposition

Recap: Decompose to remove redundancies

1. We saw that **redundancies** in the data (“bad FDs”) can lead to data anomalies
2. We developed mechanisms to **detect and remove redundancies by decomposing tables into BCNF**
 1. BCNF decomposition is *standard practice*- very powerful & widely used!
3. However, sometimes decompositions can lead to **more subtle unwanted effects...**

When does this happen?

Recovering information from a decomposition



R_1 = the *projection* of R on $A_1, \dots, A_n, B_1, \dots, B_m$


R_2 = the *projection* of R on $A_1, \dots, A_n, C_1, \dots, C_p$

Recovering information from a decomposition


Name	Price	Category
Gizmo	19.99	Gadget
OneClick	24.99	Camera
Gizmo	19.99	Camera

Sometimes a decomposition is “correct”

i.e. it is a Lossless decomposition



Name	Price
Gizmo	19.99
OneClick	24.99
Gizmo	19.99



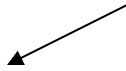
Name	Category
Gizmo	Gadget
OneClick	Camera
Gizmo	Camera

Recovering information from a decomposition

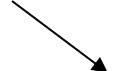
Name	Price	Category
Gizmo	19.99	Gadget
OneClick	24.99	Camera
Gizmo	19.99	Camera

However sometimes
it isn't

What's wrong
here?

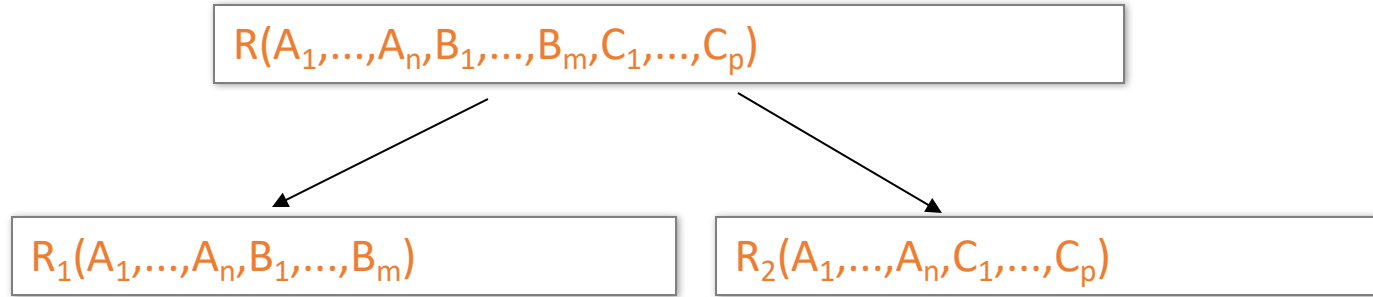


Name	Category
Gizmo	Gadget
OneClick	Camera
Gizmo	Camera



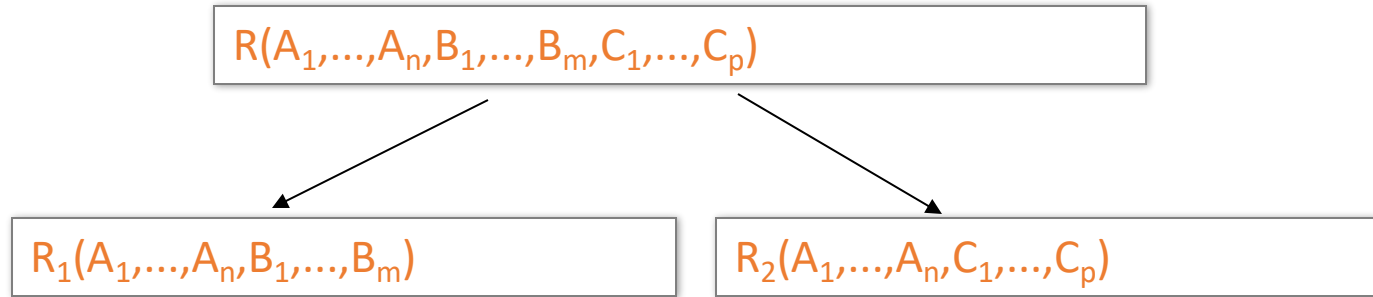
Price	Category
19.99	Gadget
24.99	Camera
19.99	Camera

Lossless Decompositions



A decomposition R to (R_1, R_2) is lossless if $R = R_1 \text{ Join } R_2$

Lossless Decompositions



If $A_1, \dots, A_n \rightarrow B_1, \dots, B_m$
Then the decomposition is lossless

Note: don't need
 $A_1, \dots, A_n \rightarrow C_1, \dots, C_p$

BCNF decomposition is always lossless. Why?

A Problem with BCNF

Unit	Company	Product
...

Unit \rightarrow Company
Company, Product \rightarrow Unit

↓

<u>Unit</u>	Company
...	...

↓

Unit	Product
...	...

Unit \rightarrow Company

We do a BCNF decomposition
on a “bad” FD:
 $\{Unit\}^+ = \{Unit, Company\}$

We lose the FD Company, Product \rightarrow Unit!!

So Why is that a Problem?

<u>Unit</u>	Company
Galaga99	UW
Bingo	UW

<u>Unit</u>	Product
Galaga99	Databases
Bingo	Databases

Unit → Company

<u>Unit</u>	Company	Product
Galaga99	UW	Databases
Bingo	UW	Databases

No problem so far.
All local FD's are satisfied.

Let's put all the data back into a single table again:

Violates the FD **Company,Product → Unit!!**

The problem with BCNF

- We started with a table R and FDs F
- We decomposed R into BCNF tables R_1, R_2, \dots with their own FDs F_1, F_2, \dots
- We insert some tuples into each of the relations—which satisfy their local FDs but when reconstruct it violates some FD **across** tables!

Practical Problem: To enforce FD, must reconstruct R —on each insert!

Desirable properties of decomposition

In general, we want the decomposition to have the following properties

(1) **Elimination of anomalies**

(2) **Recoverability of information:** can we recover the original relation by joining?

(3) **Preservation of dependencies:** if we check the projected FD's in the decomposed relations, does the reconstructed original relation satisfy the original FD's?

- BCNF gives (1) and (2), but not necessarily (3)
- 3NF gives (2) and (3), but not necessarily (1)
- In fact, there is no way to get all three at once!

3. 3NF

Third normal form (3NF)

A relation R is in 3NF if:

For every non-trivial FD $A_1, \dots, A_n \rightarrow B$, either

- $\{A_1, \dots, A_n\}$ is a superkey for R
- B is a prime attribute (i.e., B is part of some candidate key of R)

Example:

- The keys are AB and AC
- $B \rightarrow C$ is a BCNF violation, but not a 3NF violation because C is prime (part of the key AC)

R(A,B,C)

AC \rightarrow B
B \rightarrow C

3NF Decomposition Algorithm

3NFDecomp(R, F):

- Find minimal basis for F, say G
- For each FD $X \rightarrow A$ in G, if there is no relation that contains XA, create a new relation (X, A)
- Eliminate any relation that is a proper subset of another relation.
- If none of the resulting schemas are superkeys, add one more relation whose schema is a key for R

Keys:

ABE,ACE

Minimal basis:

$AB \rightarrow C$
 $C \rightarrow B$
 $A \rightarrow D$

R(A,B,C,D,E)

$R_1(A,B,C)$

$R_2(B,C)$

$R_3(A,D)$

$R_4(A,B,E)$

Minimal basis generation

Given a set of FD's F , any set of FD's equivalent to F is a **basis** for F

Input: $F = \{A \rightarrow AB, AB \rightarrow C\}$

1. Split FD's so that they have singleton right sides

$G = \{A \rightarrow B, A \rightarrow A, AB \rightarrow C\}$

2. Remove trivial FDs

$G = \{A \rightarrow B, AB \rightarrow C\}$

3. Minimize the left sides of each FD

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

4. Remove redundant FDs

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

*For each FD $X \rightarrow A$ in F :
For each attribute B in X :
If $(X - \{B\})^+$ contains A ,
remove B from X .*

Exercise #2

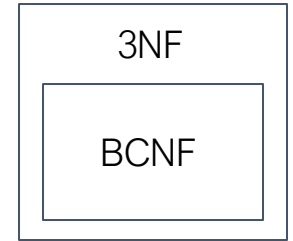
- What are the 3NF violations of the FDs?
- Decompose into relations satisfying 3NF

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

$AB \rightarrow C$
 $C \rightarrow D$
 $D \rightarrow A$

BCNF vs 3NF

- Given a non-trivial FD $X \rightarrow B$ (X is a set of attributes)
 - BCNF: X must be a superkey
 - 3NF: X must be a superkey **or B is prime**
- Use 3NF over BCNF if you need dependency preservation
- However, 3NF may not remove all redundancies and anomalies



3NF relation:

A	B	C
1	2	3
3	2	3
2	3	1

F: $B \rightarrow C, AC \rightarrow B$

Can have redundancy and update anomalies

Can have deletion anomalies

4. MVDS

MVDs: Movie Theatre Example

Movie_theater	film_name	snack
Rains 216	Star Trek: The Wrath of Kahn	Kale Chips
Rains 216	Star Trek: The Wrath of Kahn	Burrito
Rains 216	Lord of the Rings: Concatenated & Extended Edition	Kale Chips
Rains 216	Lord of the Rings: Concatenated & Extended Edition	Burrito
Rains 218	Star Wars: The Boba Fett Prequel	Ramen
Rains 218	Star Wars: The Boba Fett Prequel	Plain Pasta

Are there any functional dependencies that might hold here?

No...

And yet it seems like there is some pattern / dependency...

MVDs: Movie Theatre Example

Movie_theater	film_name	snack
Rains 216	Star Trek: The Wrath of Kahn	Kale Chips
Rains 216	Star Trek: The Wrath of Kahn	Burrito
Rains 216	Lord of the Rings: Concatenated & Extended Edition	Kale Chips
Rains 216	Lord of the Rings: Concatenated & Extended Edition	Burrito
Rains 218	Star Wars: The Boba Fett Prequel	Ramen
Rains 218	Star Wars: The Boba Fett Prequel	Plain Pasta

For a given movie theatre...

MVDs: Movie Theatre Example

Movie_theater	film_name	snack
Rains 216	Star Trek: The Wrath of Kahn	Kale Chips
Rains 216	Star Trek: The Wrath of Kahn	Burrito
Rains 216	Lord of the Rings: Concatenated & Extended Edition	Kale Chips
Rains 216	Lord of the Rings: Concatenated & Extended Edition	Burrito
Rains 218	Star Wars: The Boba Fett Prequel	Ramen
Rains 218	Star Wars: The Boba Fett Prequel	Plain Pasta

For a given movie theatre...

Any movie / snack combination is possible!

MVDs: Movie Theatre Example

	Movie_theater (A)	film_name (B)	Snack (C)
t_1	Rains 216	Star Trek: The Wrath of Kahn	Kale Chips
	Rains 216	Star Trek: The Wrath of Kahn	Burrito
	Rains 216	Lord of the Rings: Concatenated & Extended Edition	Kale Chips
t_2	Rains 216	Lord of the Rings: Concatenated & Extended Edition	Burrito
	Rains 218	Star Wars: The Boba Fett Prequel	Ramen
	Rains 218	Star Wars: The Boba Fett Prequel	Plain Pasta

More formally, we write $\{A\} \rightarrow \{B\}$ if for any tuples t_1, t_2 s.t. $t_1[A] = t_2[A]$

MVDs: Movie Theatre Example

	Movie_theater (A)	film_name (B)	Snack (C)
t_1	Rains 216	Star Trek: The Wrath of Kahn	Kale Chips
t_3	Rains 216	Star Trek: The Wrath of Kahn	Burrito
	Rains 216	Lord of the Rings: Concatenated & Extended Edition	Kale Chips
t_2	Rains 216	Lord of the Rings: Concatenated & Extended Edition	Burrito
	Rains 218	Star Wars: The Boba Fett Prequel	Ramen
	Rains 218	Star Wars: The Boba Fett Prequel	Plain Pasta

More formally, we write $\{A\} \rightarrow \{B\}$ if for any tuples t_1, t_2 s.t. $t_1[A] = t_2[A]$ there is a tuple t_3 s.t.

- $t_3[A] = t_1[A]$

MVDs: Movie Theatre Example

	Movie_theater (A)	film_name (B)	Snack (C)
t_1	Rains 216	Star Trek: The Wrath of Kahn	Kale Chips
t_3	Rains 216	Star Trek: The Wrath of Kahn	Burrito
	Rains 216	Lord of the Rings: Concatenated & Extended Edition	Kale Chips
t_2	Rains 216	Lord of the Rings: Concatenated & Extended Edition	Burrito
	Rains 218	Star Wars: The Boba Fett Prequel	Ramen
	Rains 218	Star Wars: The Boba Fett Prequel	Plain Pasta

More formally, we write $\{A\} \rightarrow \{B\}$ if for any tuples t_1, t_2 s.t. $t_1[A] = t_2[A]$ there is a tuple t_3 s.t.

- $t_3[A] = t_1[A]$
- $t_3[B] = t_1[B]$

MVDs: Movie Theatre Example

	Movie_theater (A)	film_name (B)	Snack (C)
t_1	Rains 216	Star Trek: The Wrath of Kahn	Kale Chips
t_3	Rains 216	Star Trek: The Wrath of Kahn	Burrito
	Rains 216	Lord of the Rings: Concatenated & Extended Edition	Kale Chips
t_2	Rains 216	Lord of the Rings: Concatenated & Extended Edition	Burrito
	Rains 218	Star Wars: The Boba Fett Prequel	Ramen
	Rains 218	Star Wars: The Boba Fett Prequel	Plain Pasta

More formally, we write $\{A\} \rightarrow \{B\}$ if for any tuples t_1, t_2 s.t. $t_1[A] = t_2[A]$ there is a tuple t_3 s.t.

- $t_3[A] = t_1[A]$
- $t_3[B] = t_1[B]$
- and $t_3[R \setminus B] = t_2[R \setminus B]$

Where $R \setminus B$ is “R minus B”
i.e. the attributes of R not in B

MVDs: Movie Theatre Example

	Movie_theater (A)	film_name (B)	Snack (C)
t_2	Rains 216	Star Trek: The Wrath of Kahn	Kale Chips
	Rains 216	Star Trek: The Wrath of Kahn	Burrito
t_3	Rains 216	Lord of the Rings: Concatenated & Extended Edition	Kale Chips
t_1	Rains 216	Lord of the Rings: Concatenated & Extended Edition	Burrito
	Rains 218	Star Wars: The Boba Fett Prequel	Ramen
	Rains 218	Star Wars: The Boba Fett Prequel	Plain Pasta

Note this also works!

An MVD holds over a relation or an instance, so defn. must hold for every applicable pair...

MVDs: Movie Theatre Example

	Movie_theater (A)	film_name (B)	Snack (C)
t_2	Rains 216	Star Trek: The Wrath of Kahn	Kale Chips
	Rains 216	Star Trek: The Wrath of Kahn	Burrito
t_3	Rains 216	Lord of the Rings: Concatenated & Extended Edition	Kale Chips
t_1	Rains 216	Lord of the Rings: Concatenated & Extended Edition	Burrito
	Rains 218	Star Wars: The Boba Fett Prequel	Ramen
	Rains 218	Star Wars: The Boba Fett Prequel	Plain Pasta

This expresses a sort of dependency (= data redundancy) that we can't express with FDs

*Actually, it expresses conditional independence (between film and snack given movie theatre)!

Multi-Value Dependencies (MVDs)

A multi-value dependency (MVD) is another type of dependency that could hold in our data, *which is not captured by FDs*

- Every FD is an MVD

Definition:

- Given a relation R , attribute set A , and two sets of attributes $\mathbf{X}, \mathbf{Y} \subseteq A$
- The multi-value dependency (MVD) $\mathbf{X} \twoheadrightarrow \mathbf{Y}$ holds on R if for any tuples $\mathbf{t}_1, \mathbf{t}_2 \in R$ s.t. $\mathbf{t}_1[\mathbf{X}] = \mathbf{t}_2[\mathbf{X}]$, there exists a tuple \mathbf{t}_3 s.t.:
 - $t_1[X] = t_2[X] = t_3[X]$
 - $t_1[Y] = t_3[Y]$
 - $t_2[A \setminus Y] = t_3[A \setminus Y]$

$A \setminus B$ means “elements of set A not in set B ”

Multi-Value Dependencies (MVDs)

One less formal, literal way to phrase the definition of an MVD:

The MVD $\mathbf{X} \twoheadrightarrow \mathbf{Y}$ holds on R if for any pair of tuples with the same X values, the tuples with the same X values, but the other permutations of Y and A\Y values, is also in R

Ex: $X = \{x\}$, $Y = \{y\}$:

x	y	z
1	0	1
1	1	0



For $X \twoheadrightarrow Y$ to hold must have...

x	y	z
1	0	1
1	1	0
1	0	0
1	1	1

Multi-Value Dependencies (MVDs)

Another way to understand MVDs, in terms of conditional independence:

The MVD $\mathbf{X} \twoheadrightarrow \mathbf{Y}$ holds on R if given X, Y is conditionally independent of $A \setminus Y$ and vice versa...

Here, given $x = 1$, we know for ex. that:
 $y = 0 \rightarrow z = 1$

I.e. z is conditionally *dependent* on y given x

x	y	z
1	0	1
1	1	0

Here, this is not the case!

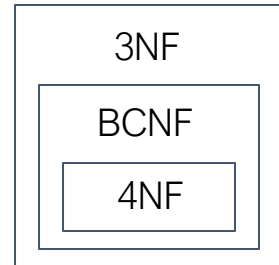
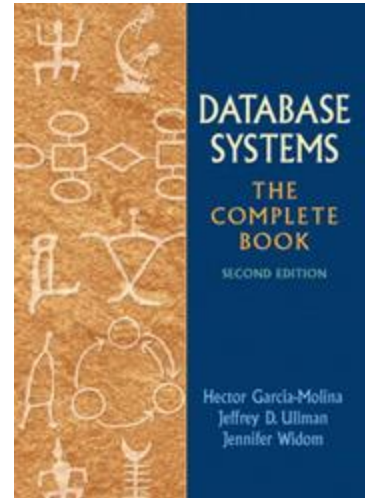
I.e. z is conditionally *independent* of y given x

x	y	z
1	0	1
1	1	0
1	0	0
1	1	1

Further Readings (Chapter 3.6)

4NF: Remove MVD redundancies

Property	3NF	BCNF	4NF
Lossless join	Yes	Yes	Yes
Eliminates FD redundancies	No	Yes	Yes
Eliminates MVD redundancies	No	No	Yes
Preserves FD's	Yes	No	No
Preserves MVD's	No	No	No



Summary

Good schema design is important

- Avoid redundancy and anomalies
- Functional dependencies

Normal forms describe how to **remove** this redundancy by **decomposing** relations

- BCNF gives elimination of anomalies and lossless join
- 3NF gives lossless join and dependency preservation

BCNF is intuitive and most widely used in practice