L15: Normalization 3

CS3200 Database design (fa18 s2)

https://northeastern-datalab.github.io/cs3200/

Version 10/29/2018

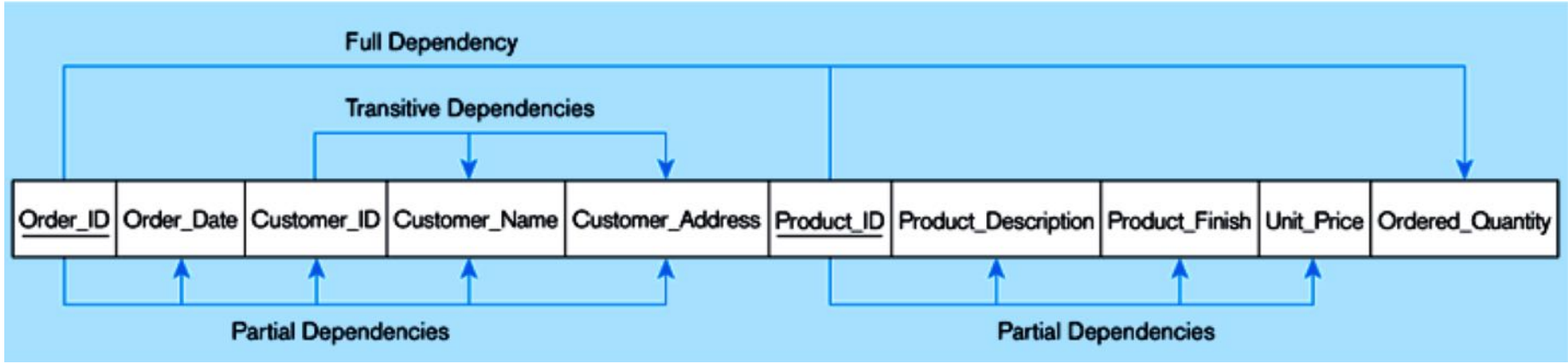
Announcements!

- Exam 2 in 1 week
- Jupyter exercises: do they work now?
- Stanford chapter on "Design theory" posted on BB
- Come to OHs
  - There will be changes this week (no THU, but TUE and FRI)
- Today
  - 3NF vs BCNF
  - Transactions

Boyce-Codd Normal Form (BCNF)

Quick recap FDs

- **Functional Dependency (FD)**: The value of one set of attributes (the **determinant**) uniquely determines the value of another set of attributes (the **dependents**)
- A **superkey (SK)** is as a set of attributes of a relation schema upon which all attributes of the schema are functionally dependent.
- A **(candidate) key (CK)** is a non-redundant (minimal) SK (sometimes called just "a key")
- **Prime attribute**: belonging to some candidate key
- **Partial FD**: FD in which more non-prime attributes are functionally dependent on part (but not all) of any CK
- **Transitive FD**: An FD between two (or more) nonkey attributes (important for distinction 3NF vs BCNF!)
- **3NF**: no partial nor transitive FD



Boyce-Codd Normal Form (BCNF)

- Boyce-Codd normal form (BCNF)
  - A relation is in BCNF, if and only if, every (non-trival) determinant is a superkey (SK).

- The difference between 3NF and BCNF is that for a FD A→B,
  - 3NF allows this dependency in a relation if B is a primary-key attribute and A is not a candidate key (CK),
  - whereas BCNF insists that for this dependency to remain in a relation, A must be a SK (contain a CK).

3NF to BCNF

STUDENT ADVISOR

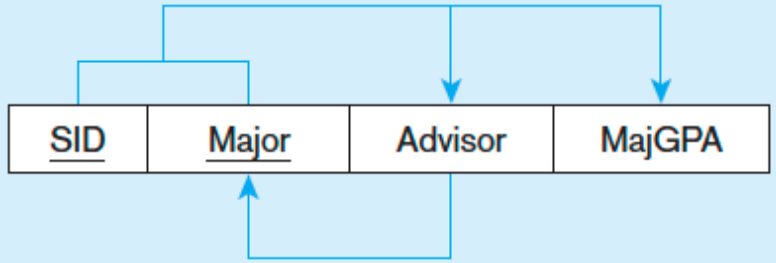
SID	Major	Advisor	MajGPA
123	Physics	Hawking	4.0
123	Music	Mahler	3.3
456	Literature	Michener	3.2
789	Music	Bach	3.7
678	Physics	Hawking	3.5

Source: Hoffer, Ramesh, Topi, Modern database management, 10th ed, Appendix B, 2010.

3NF to BCNF

STUDENT ADVISOR

SID	Major	Advisor	MajGPA
123	Physics	Hawking	4.0
123	Music	Mahler	3.3
456	Literature	Michener	3.2
789	Music	Bach	3.7
678	Physics	Hawking	3.5

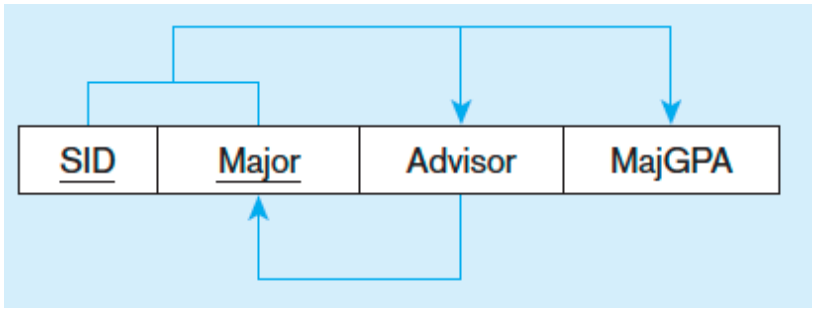


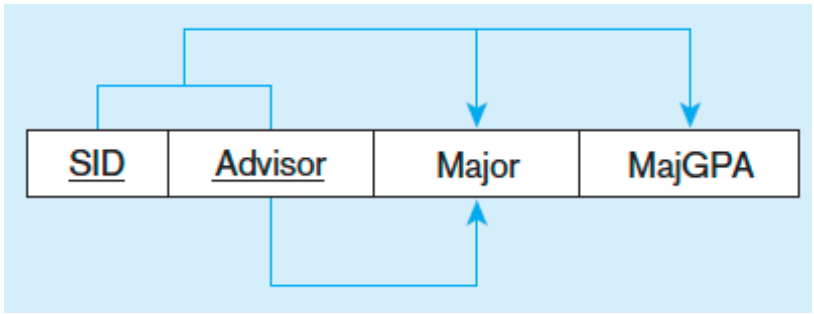
Source: Hoffer, Ramesh, Topi, Modern database management, 10th ed, Appendix B, 2010.

3NF to BCNF

STUDENT ADVISOR

SID	Major	Advisor	MajGPA
123	Physics	Hawking	4.0
123	Music	Mahler	3.3
456	Literature	Michener	3.2
789	Music	Bach	3.7
678	Physics	Hawking	3.5



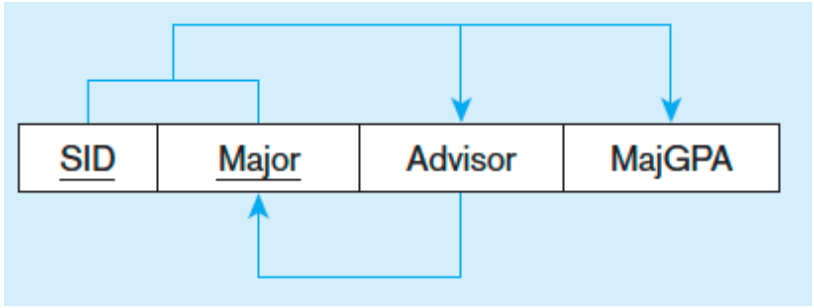


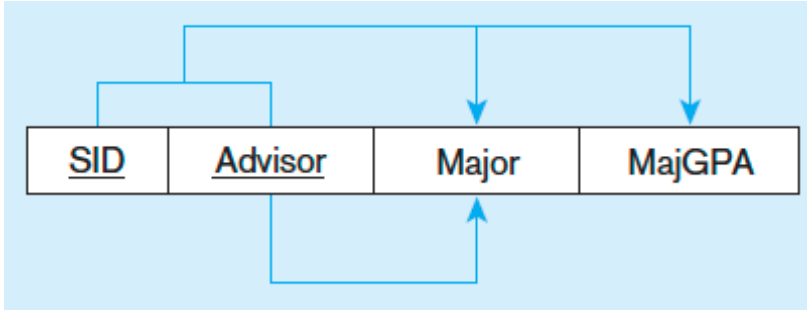
Source: Hoffer, Ramesh, Topi, Modern database management, 10th ed, Appendix B, 2010.

3NF to BCNF

STUDENT ADVISOR

SID	Major	Advisor	MajGPA
123	Physics	Hawking	4.0
123	Music	Mahler	3.3
456	Literature	Michener	3.2
789	Music	Bach	3.7
678	Physics	Hawking	3.5



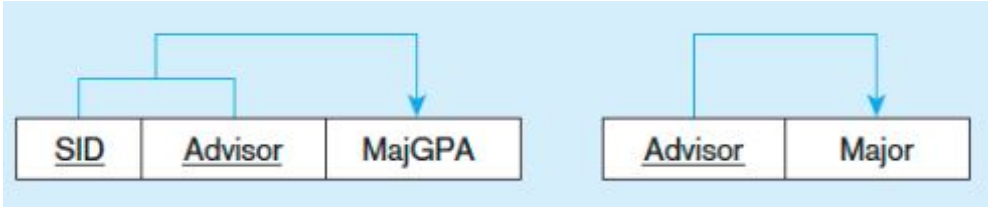


STUDENT

SID	Advisor	MajGPA
123	Hawking	4.0
123	Mahler	3.3
456	Michener	3.2
789	Bach	3.7
678	Hawking	3.5

ADVISOR

Advisor	Major
Hawking	Physics
Mahler	Music
Michener	Literature
Bach	Music



Source: Hoffer, Ramesh, Topi, Modern database management, 10th ed, Appendix B, 2010.

BCNF vs 3NF

- **BCNF**: For every nontrivial FD X→Y over relation *R*:
  - X is a superkey of R

- **3NF**: For every nontrivial FD X→Y over relation *R*, either:
  - X is a superkey of R
  - or Y is prime (i.e. it is part of some CK)

Recall: a FD X→Y is "trivial" iff Y⊆X

Recall: no subset of a CK is a CK

Back to Conceptual Design

- Now that we know how to find FDs, it's a straight-forward process:
  - Search for "bad" FDs
  - If there are any, then keep decomposing the table into sub-tables until no more bad FDs
  - When done, the database schema is normalized

Recall: there are several normal forms…

Boyce-Codd Normal Form (BCNF)

- Main idea is that we define “good” and “bad” FDs as follows:
  - X → B is a “good FD” if X is a (super)key
    - In other words, if B is the set of all attributes
  - X → B 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 $\{A_1, ..., A_n\} \rightarrow B$ is a *non-trivial* FD in R

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

Equivalently: ∀ sets of attributes X, either ($X^+ = X$) or (X^+ = 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	Boston
Joe	987-65-4321	908-555-1234	Boston

{SSN} → {Name,City}

This FD is *bad* because it is **not** a superkey

⟹ **Not** in BCNF

What is the key?
{SSN, PhoneNumber}

Example

Name	SSN	City
Fred	123-45-6789	Seattle
Joe	987-65-4321	Boston

SSN	PhoneNumber
123-45-6789	206-555-1234
123-45-6789	206-555-6543
987-65-4321	908-555-2121
987-65-4321	908-555-1234

Now in BCNF!

{SSN} → {Name,City}

This FD is now *good* because it is the key

Let's check anomalies:

- Redundancy ?
- Update ?
- Delete ?

BCNF Decomposition Algorithm

BCNFDecomp(R):

BCNF Decomposition Algorithm

BCNFDecomp(R):

Find *a set of attributes* X s.t.: $X^+ \neq X$ and $X^+ \neq$ [all attributes]

Find a set of attributes X which has non-trivial "bad" FDs, i.e. is not a superkey, using closures

BCNF Decomposition Algorithm

BCNFDecomp(R):

Find a *set of attributes* X s.t.: $X^+ \neq X$ and $X^+ \neq$ [all attributes]

if (not found) **then Return** R

If no "bad" FDs found, in BCNF!

BCNF Decomposition Algorithm

BCNFDecomp(R):

Find a *set of attributes* X s.t.: $X^+ \neq X$ and $X^+ \neq$ [all attributes]

if (not found) **then Return** R

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):

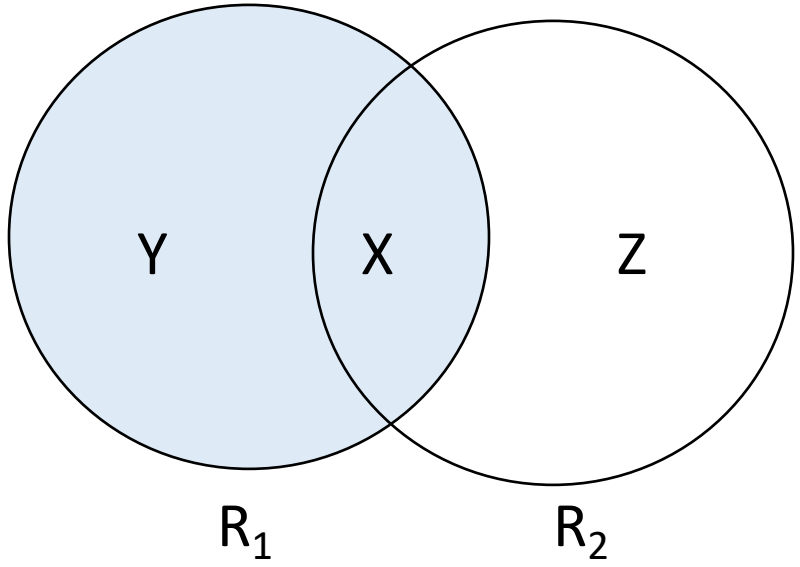
Find a *set of attributes* X s.t.: $X^+ \neq X$ and $X^+ \neq$ [all attributes]

if (not found) then **Return** R

let $Y = X^+ - X$, $Z = (X^+)^C$

decompose R into $\mathbf{R_1(X \cup Y)}$ and $\mathbf{R_2(X \cup Z)}$

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



BCNF Decomposition Algorithm

BCNFDecomp(R):

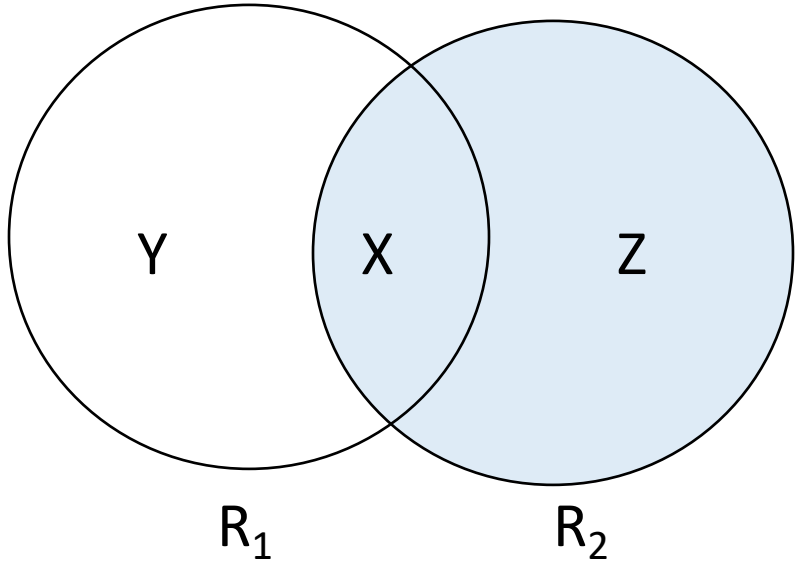
Find a *set of attributes* X s.t.: $X^+ \neq X$ and $X^+ \neq$ [all attributes]

if (not found) then Return R

let $Y = X^+ - X$, $Z = (X^+)^C$

decompose R into $\mathbf{R_1(X \cup Y)}$ and $\mathbf{R_2(X \cup Z)}$

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



BCNF Decomposition Algorithm

BCNFDecomp(R):

Find a *set of attributes* X s.t.: $X^+ \neq X$ and $X^+ \neq$ [all attributes]

if (not found) **then Return** R

let $Y = X^+ - X$, $Z = (X^+)^C$

decompose R into $R_1(X \cup Y)$ and $R_2(X \cup Z)$

Return BCNFDecomp(R_1), BCNFDecomp(R_2)

Proceed recursively until no more "bad" FDs!

Example for BCNF

BCNFDecomp(R):

Find a *set of attributes* X s.t.: $X^+ \neq X$ and $X^+ \neq$ [all attributes]

<u>if</u> (not found) <u>then</u> **Return** R

<u>let</u> $Y = X^+ - X$, $Z = (X^+)^C$

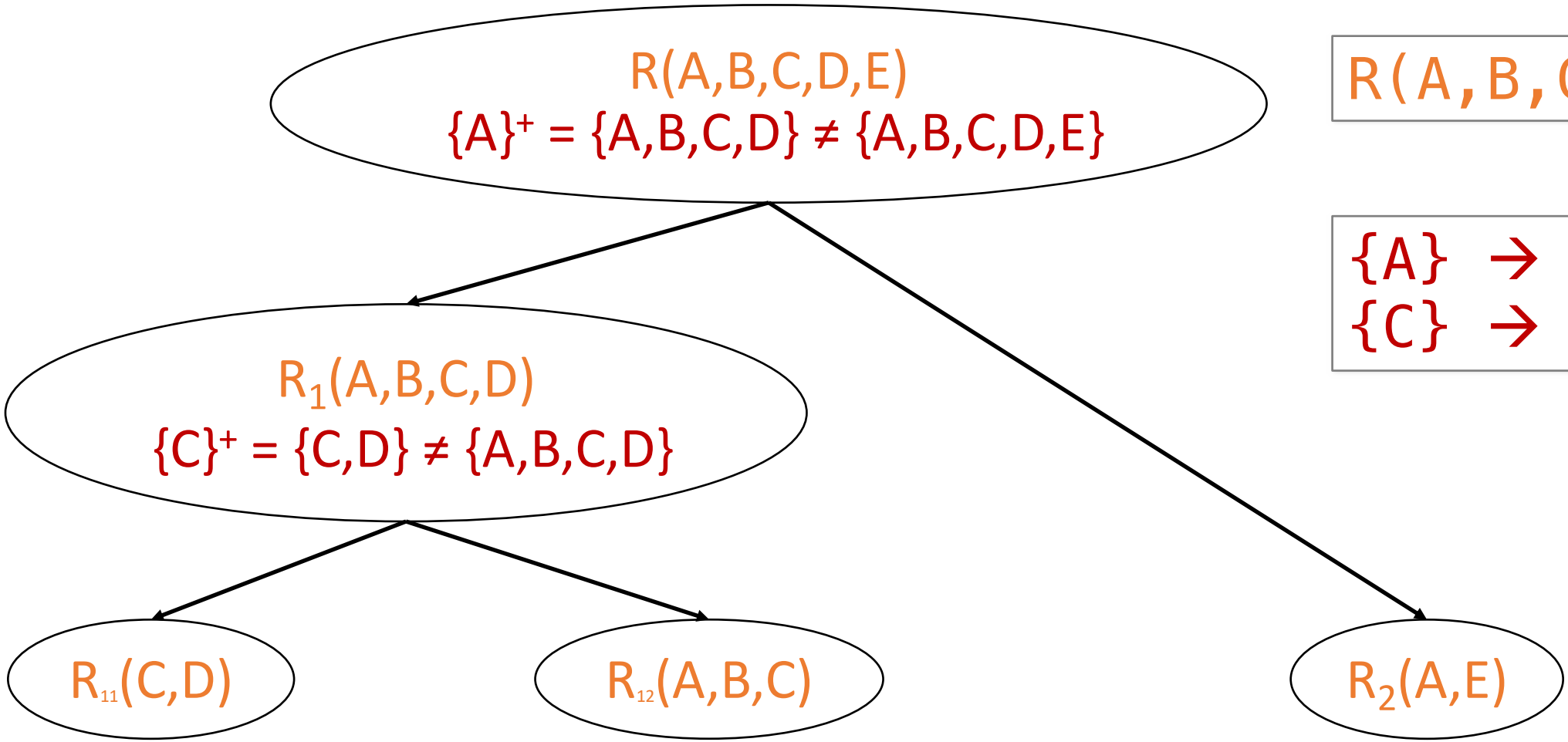
decompose R into $\mathbf{R_1(X \cup Y)}$ and $\mathbf{R_2(X \cup Z)}$

Return BCNFDecomp(R_1), BCNFDecomp(R_2)

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

{A} → {B,C}
{C} → {D}

Example



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

{A} → {B,C}
{C} → {D}

Practice (at home)

- Activity-22.ipynb

Decompositions

Recap: Decompose to remove redundancies

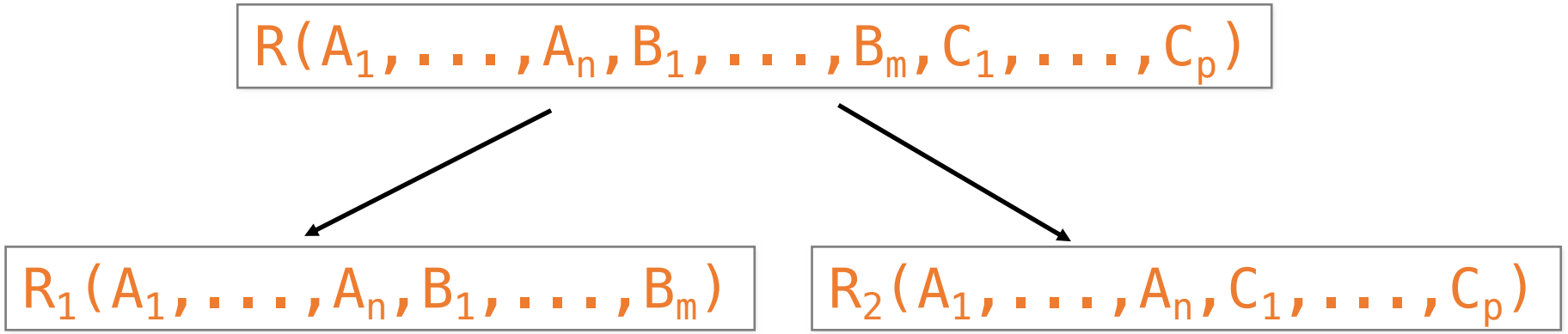
- We saw that redundancies in the data ("bad FDs") can lead to data anomalies

- We developed mechanisms to detect and remove redundancies by decomposing tables into 3NF or BCNF
  - BCNF decomposition is standard practice: very powerful & widely used!

- However, sometimes decompositions can lead to more subtle unwanted effects…

When does this happen?

Decompositions in General

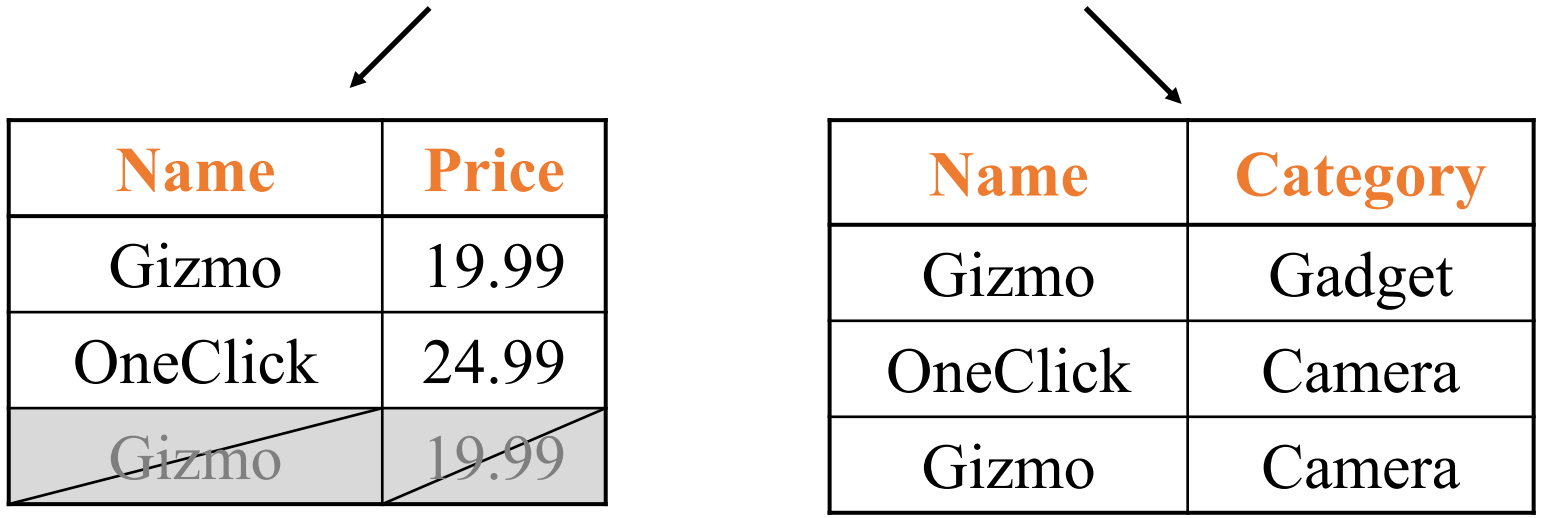


R_1 = the *projection* of R on A_1, ..., A_n, B_1, ..., B_m

R_2 = the *projection* of R on A_1, ..., A_n, C_1, ..., C_p

Lossless Decomposition

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



Name	Price
Gizmo	19.99
OneClick	24.99
Gizmo	19.99

Name	Category
Gizmo	Gadget
OneClick	Camera
Gizmo	Camera

Sometimes a decomposition is "correct"

I.e. it is a Lossless decomposition

Lossy Decomposition

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

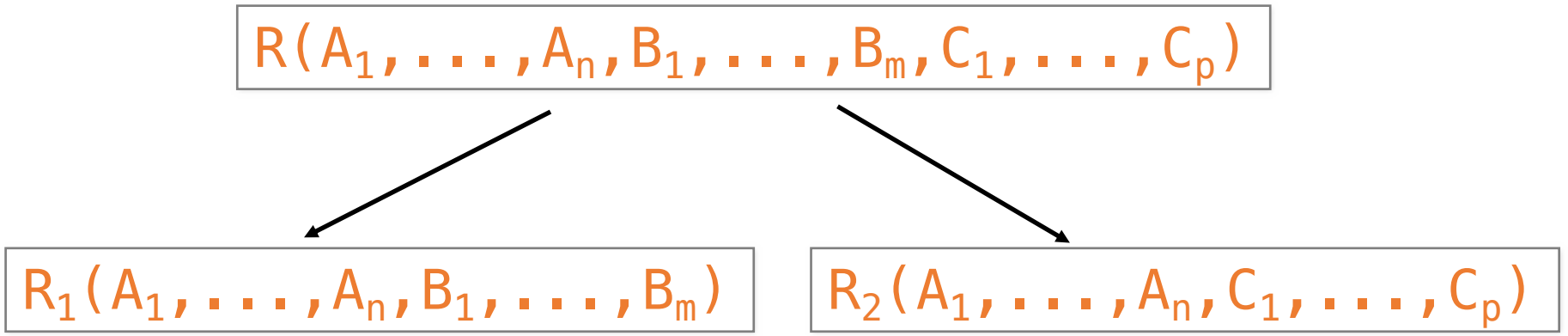
Name	Category
Gizmo	Gadget
OneClick	Camera
Gizmo	Camera

Price	Category
19.99	Gadget
24.99	Camera
19.99	Camera

However sometimes it isn't

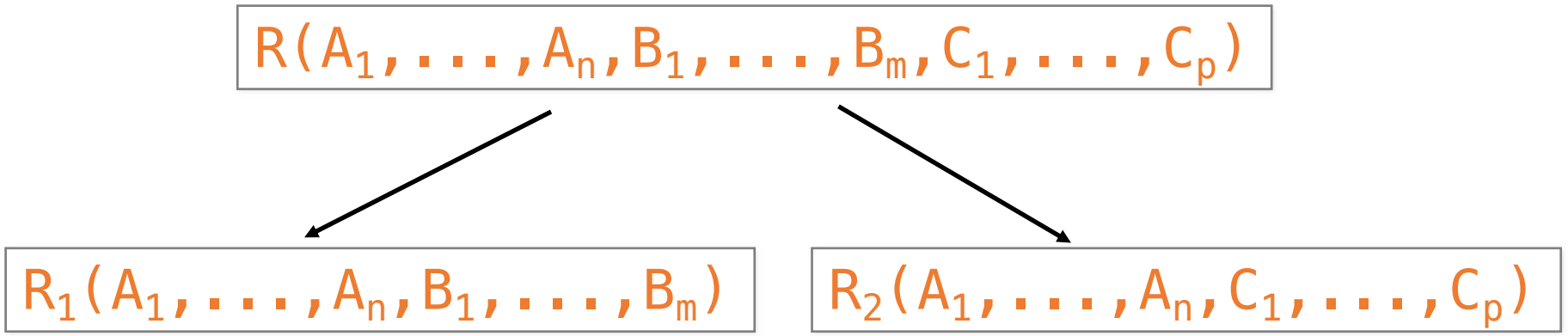
What's wrong here?

Lossless Decompositions



A decomposition R to (R1, R2) is **lossless** if $R = R1 \bowtie R2$

Lossless Decompositions



If $\{A_1, \ldots, A_n\} \rightarrow \{B_1, \ldots, B_m\}$
Then the decomposition is lossless

Note: don't need
$\{A_1, \ldots, A_n\} \rightarrow \{C_1, \ldots, C_p\}$

BCNF decomposition is always lossless. Why?

A familiar example

Item

PName	Price	Category	Manufacturer	StockPrice	Country
Gizmo	$19.99	Gadgets	GizmoWorks	25	USA
Powergizmo	$29.99	Gadgets	GizmoWorks	25	USA
SingleTouch	$149.99	Photography	Canon	65	Japan
MultiTouch	$203.99	Household	Hitachi	15	Japan

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

A familiar example

Item

PName	Price	Category	Manufacturer	StockPrice	Country
Gizmo	$19.99	Gadgets	GizmoWorks	25	USA
Powergizmo	$29.99	Gadgets	GizmoWorks	25	USA
SingleTouch	$149.99	Photography	Canon	65	Japan
MultiTouch	$203.99	Household	Hitachi	15	Japan

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

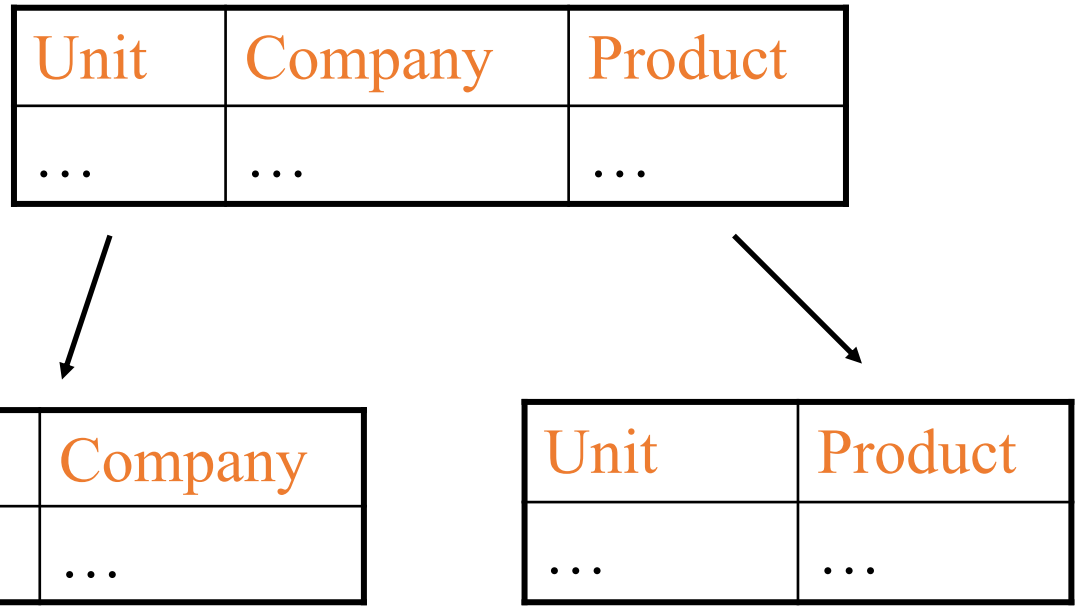
A problem with BCNF

Problem: To enforce a FD, must reconstruct original relation—*on each insert!*

Note: This is historically inaccurate, but it makes it easier to explain

A problem with BCNF

{Unit} → {Company}
{Company,Product} → {Unit}



{Unit} → {Company}

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

We lose the FD {Company,Product} → {Unit}!!

So Why is that a Problem?

{Unit} → {Company}
{Company,Product} → {Unit}

Unit	Company
Galaga99	NEU
Bingo	NEU

Unit	Product
Galaga99	Databases
Bingo	Databases

{Unit} → {Company}

Unit	Company	Product
Galaga99	NEU	Databases
Bingo	NEU	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

- We started with a table R and FDs F
- We decomposed R into BCNF tables R1, R2, …
  with their own FDs F1, F2, …
- 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!*

Possible Solutions

- Various ways to handle so that decompositions are all lossless / no FDs lost
  - For example 3NF: stop short of full BCNF decompositions.

- Usually a tradeoff between redundancy / data anomalies and FD preservation…

BCNF still most common- with additional steps to keep track of lost FDs…

Summary

- A decomposition of relation R into relation R1 and R2 is lossless if
  - R1 ⋈ R2 = R
- A decomposition is dependency-preserving if
  - all FDs (functional dependencies) from R are preserved in either R1 or R2 (or both or derivable from a combination of the FDs in R1 and R2).
  - a decomposition of relation R is dependency preserving if the Functional dependency of R can be obtained by taking the union of the functional dependency of all the decomposed relation.
    - The **dependency preservation decomposition** is another property of decomposed relational database schema D in which each functional dependency X -> Y specified in F either appeared directly in one of the relation schemas R_i in the decomposed D or could be inferred from the dependencies that appear in some Ri.
- A decomposition of Relation R into R1 and R2 is lossless if and only if at least one of following dependencies hold:
  - 1. R1 ∩ R2 -> R1
  - 2. R1 ∩ R2 -> R2

4NF and higher

3NF Motivation

A relation R is in 3rd normal form if :
Whenever there is a nontrivial dep. $A_1, A_2, ..., A_n \rightarrow B$ for R,
then $\{A_1, A_2, ..., A_n\}$ is a super-key for R,
or B is part of a key.

Tradeoffs:

- BCNF: no anomalies, but may lose some FDs
- 3NF: keeps all FDs, but may have some anomalies

Motivation of 4NF and higher

Assume for each course, we can independently choose a lecturer and a book. What is the problem?

Classes

Course	Lecturer	Book
cse444	Alexandra	Complete book
cse444	Wolfgang	Complete book
cse444	Alexandra	Cow book

cse444	Wolfgang	Cow book

Multi-valued dependency (MVD) Course $\rightarrow\!\!\rightarrow$ Lecturer:
In every legal instance, each Course value is associated with a set of Lecturer values and this set is independent of the values in the other attributes (here Book).