L15: Normalization

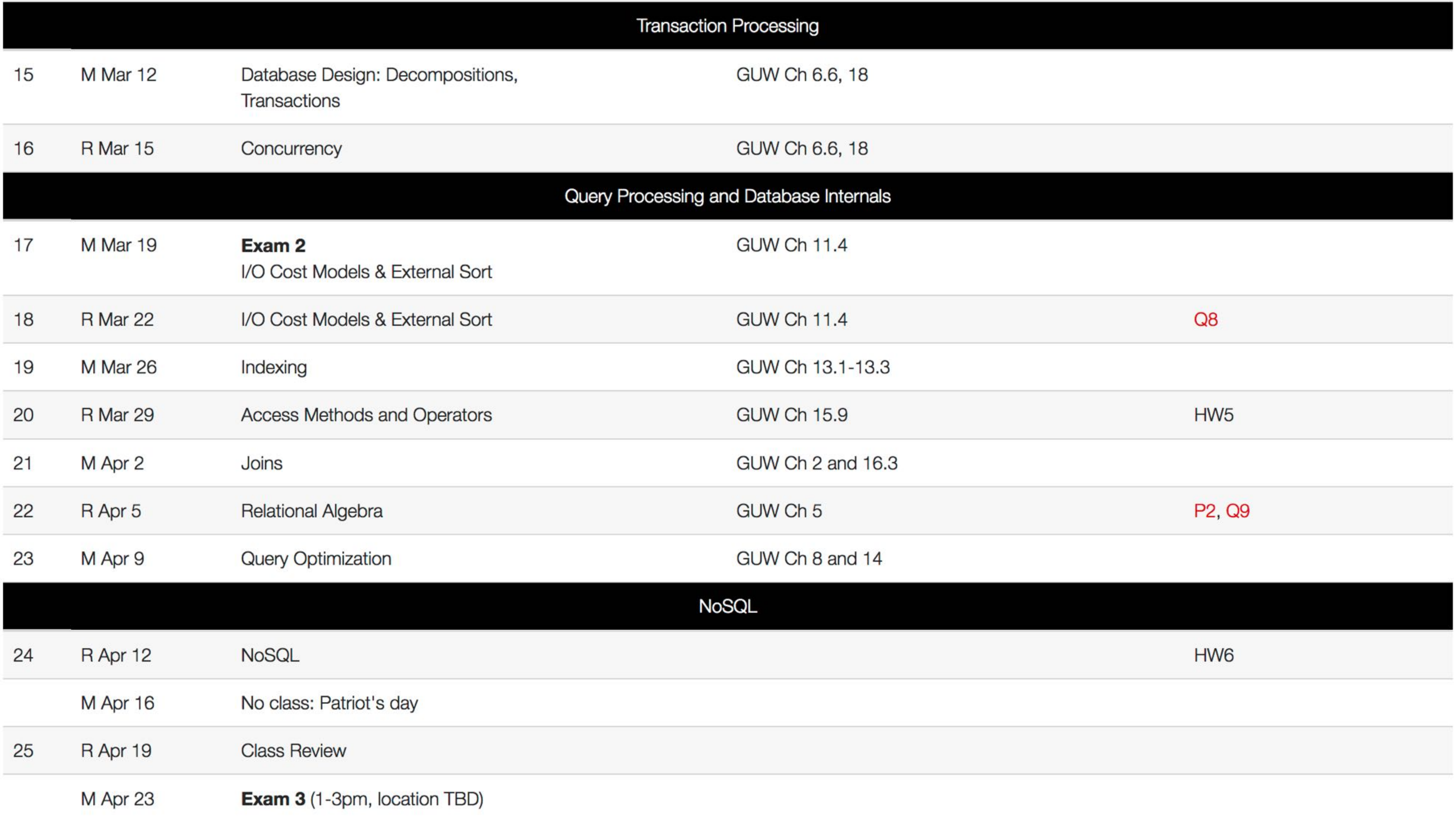
CS3200 Database design (sp18 s2)

https://course.ccs.neu.edu/cs3200sp18s2/

3/12/2018

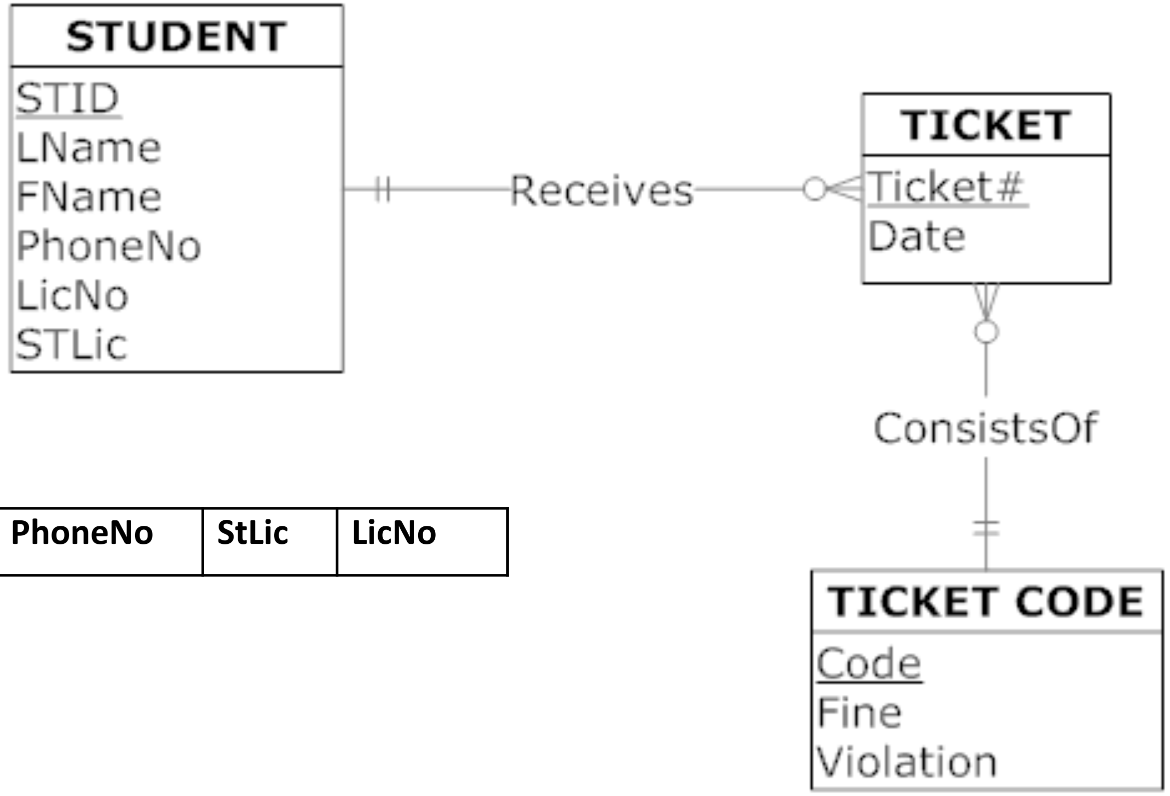
Announcements!

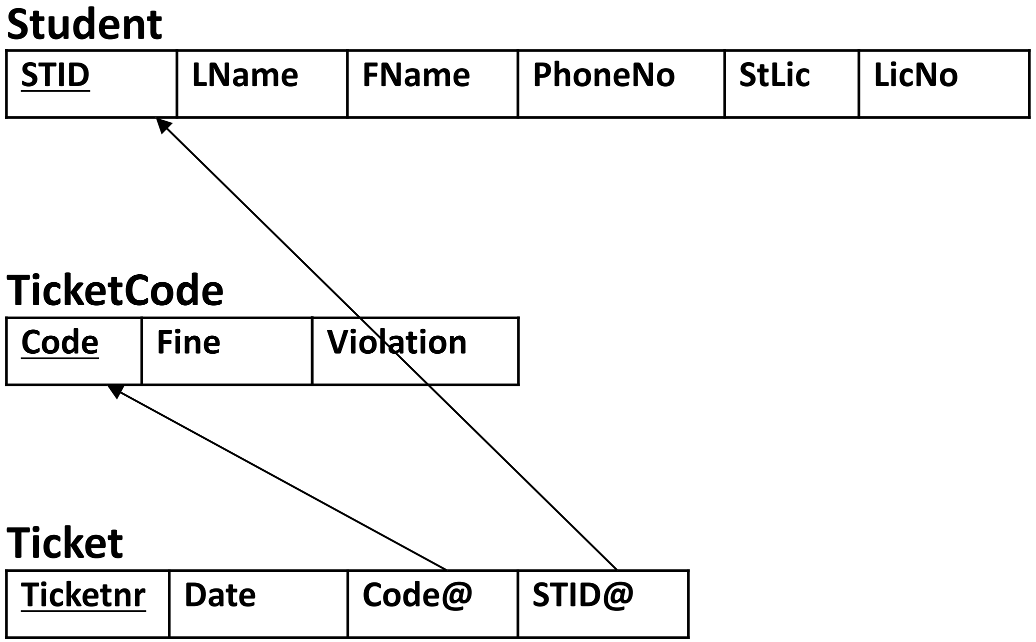
- Keep bringing your name plates ☺
- Verify your grades and feedback on BB. If something is unclear or confusing, or displays incorrectly, please let us know (e.g., via Piazza instructors only)
- P2 is posted and updated calendar
- Exam2 next week: content is everything seen until this Thursday: setup like for Exam1: laptop SQL + paper database design + paper transactions
- Outline today
  - HW4, Projects
  - Decompositions
  - Transactions!



Transaction Processing				
15	M Mar 12	Database Design: Decompositions, Transactions	GUW Ch 6.6, 18	
16	R Mar 15	Concurrency	GUW Ch 6.6, 18	
Query Processing and Database Internals				
17	M Mar 19	**Exam 2** I/O Cost Models & External Sort	GUW Ch 11.4	
18	R Mar 22	I/O Cost Models & External Sort	GUW Ch 11.4	Q8
19	M Mar 26	Indexing	GUW Ch 13.1-13.3	
20	R Mar 29	Access Methods and Operators	GUW Ch 15.9	HW5
21	M Apr 2	Joins	GUW Ch 2 and 16.3	
22	R Apr 5	Relational Algebra	GUW Ch 5	P2, Q9
23	M Apr 9	Query Optimization	GUW Ch 8 and 14	
NoSQL				
24	R Apr 12	NoSQL		HW6
	M Apr 16	No class: Patriot's day		
25	R Apr 19	Class Review		
	M Apr 23	**Exam 3** (1-3pm, location TBD)		

Ryan's question: Parking Tickets: ER Diagram





Student

STID	LName	FName	PhoneNo	StLic	LicNo

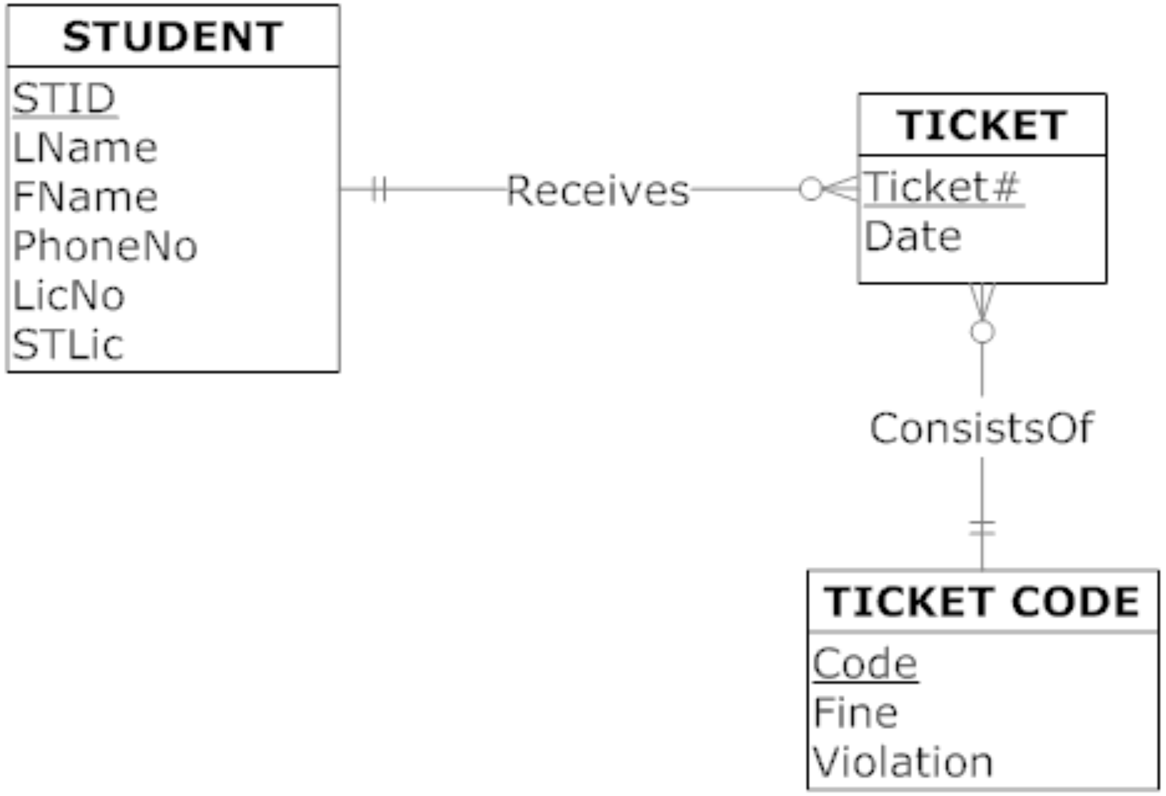
TicketCode

Code	Fine	Violation

Ticket

Ticketnr	Date	Code@	STID@

Ryan's question: Parking Tickets: ER Diagram



STUDENT

STID
LName
FName
PhoneNo
LicNo
STLic

Receives

TICKET

Ticket#
Date

ConsistsOf

TICKET CODE

Code
Fine
Violation

Decompositions

Recap: Decompose to remove redundancies

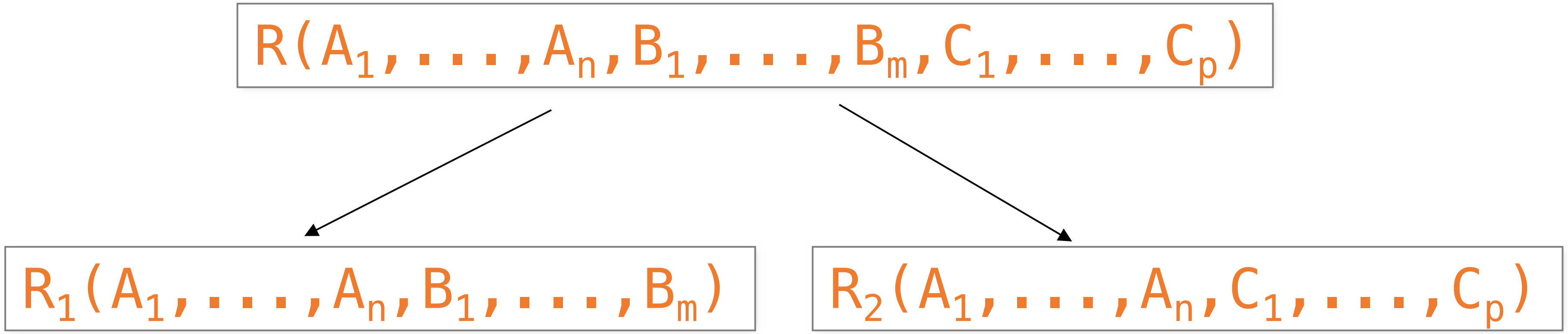
- We saw that redundancies in the data (“bad FDs”) can lead to data anomalies

- 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



$R(A_1,\ldots,A_n,B_1,\ldots,B_m,C_1,\ldots,C_p)$

$R_1(A_1,\ldots,A_n,B_1,\ldots,B_m)$

$R_2(A_1,\ldots,A_n,C_1,\ldots,C_p)$

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

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

Theory of Decomposition

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

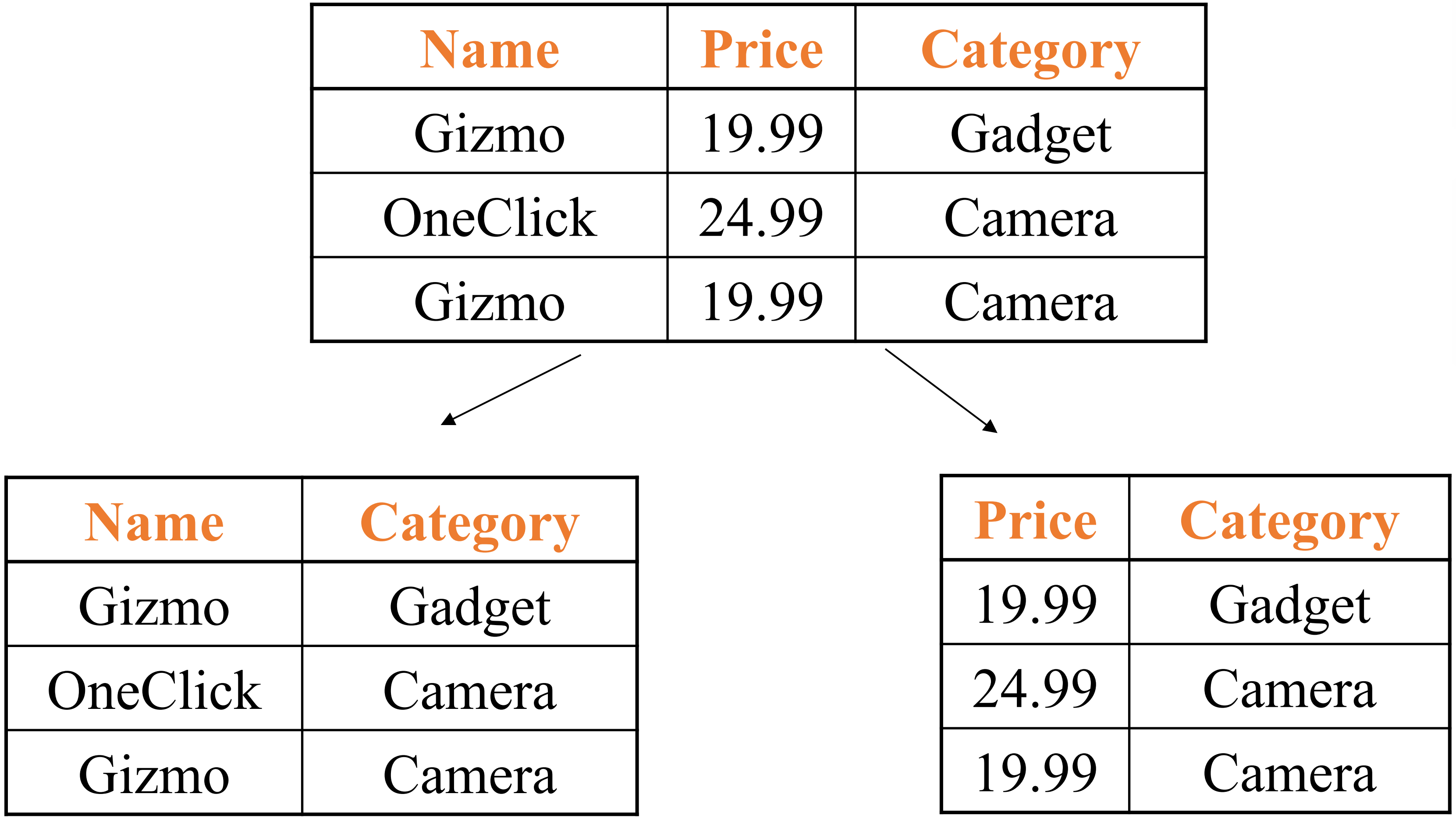
Name	Price
Gizmo	19.99
OneClick	24.99
~~Gizmo~~	~~19.99~~

Name	Category
Gizmo	Gadget
OneClick	Camera
Gizmo	Camera

Sometimes a decomposition is "correct"

I.e. it is a <u>**Lossless decomposition**</u>

Lossy Decomposition



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

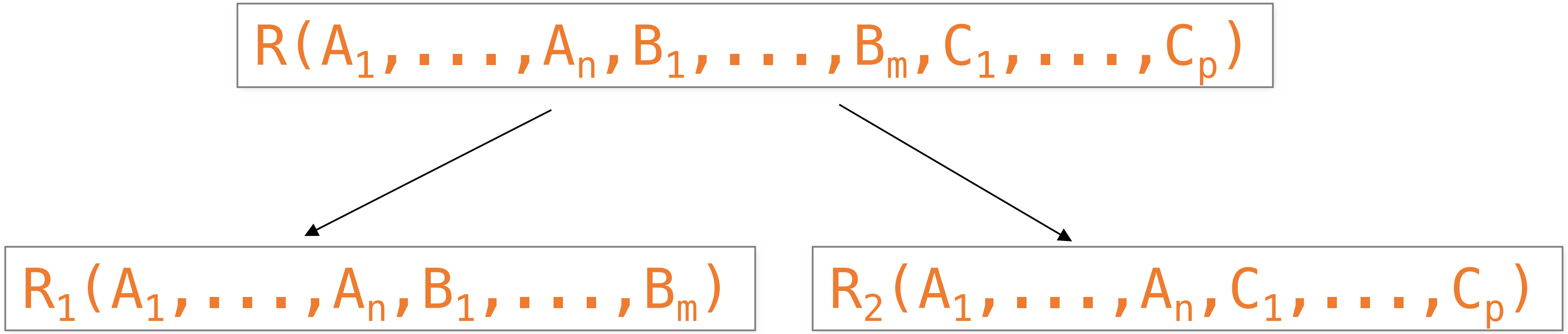
Name	Category
Gizmo	Gadget
OneClick	Camera
Gizmo	Camera

Price	Category
19.99	Gadget
24.99	Camera
19.99	Camera

However sometimes it isn’t

What’s wrong here?

Lossless Decompositions



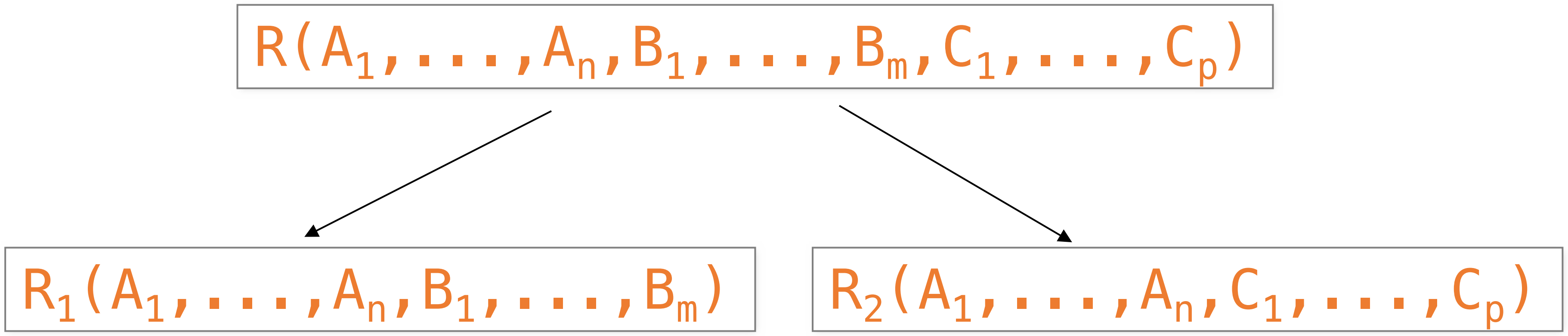
$R(A_1,\ldots,A_n,B_1,\ldots,B_m,C_1,\ldots,C_p)$

$R_1(A_1,\ldots,A_n,B_1,\ldots,B_m)$

$R_2(A_1,\ldots,A_n,C_1,\ldots,C_p)$

A decomposition R to (R1, R2) is **lossless** if R = R1 Join R2

Lossless Decompositions



$R(A_1,....,A_n,B_1,....,B_m,C_1,....,C_p)$

$R_1(A_1,....,A_n,B_1,....,B_m)$

$R_2(A_1,....,A_n,C_1,....,C_p)$

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?

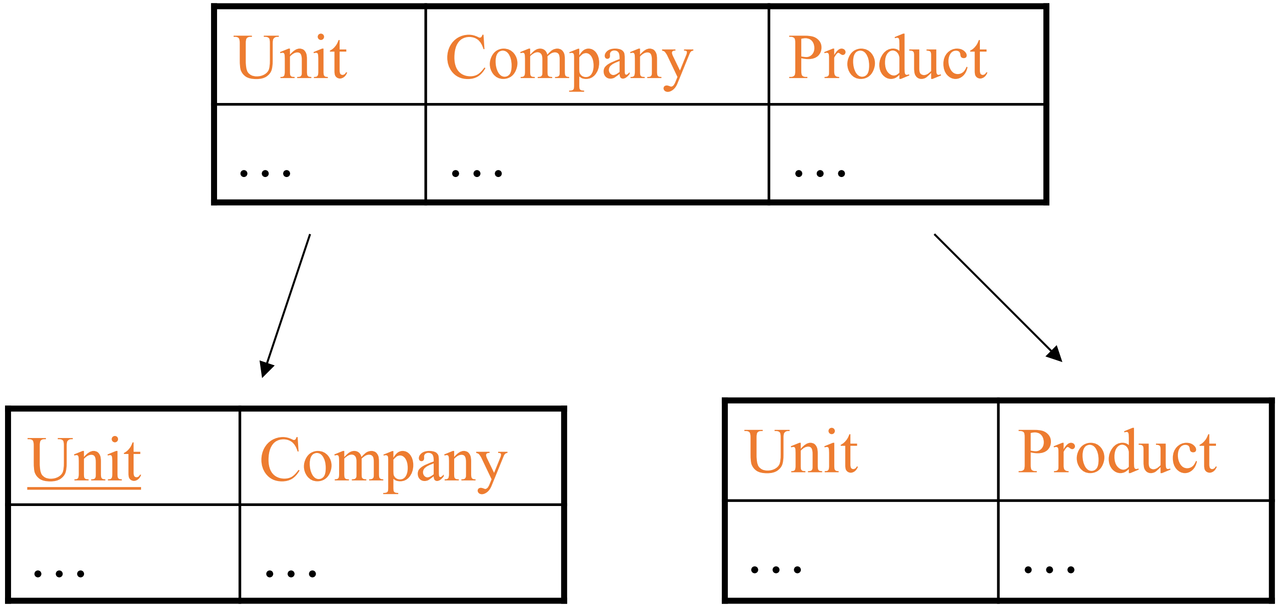
A problem with BCNF

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	Product
…	…	…

Unit	Company
…	…

Unit	Product
…	…

{Unit} → {Company}

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

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

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

{Unit} → {Company}

Unit	Company	Product
Galaga99	NEU	Databases
Bingo	NEU	Databases

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…

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 →→ 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).