L13: Relational modeling 3

CS3200 Database design (fa18 s2)

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

Version 10/22/2018

Announcements!

- Various textbook excerpts
  - Enhanced ER are not part of this class (ch 3 in Hoffer, Ramesh, Topi)
- Slides we discuss, and others we don't: those provide detailed instructions for which we develop the intuition in class
- Outline
  - We continue with Relational Data modeling
  - Then start with normalization (there is an intuitive and a formal part)
    - We will use Jupyter exercises for the more formal part

Resources

RESOURCES

- Blackboard: only used for grades, HW, Project and exam submissions
- Piazza: access code posted on Blackboard
- Gradiance: the class token is posted on Blackboard. Some help: Setup Gradiance, A tour of Gradiance
- Lucidcharts: handy for drawing ER diagrams (Class template)
- Jupyter Activities: link to our Jupyter install instructions and Jupyter activities. Additional slides: Setup Jupyter (slides)
- We will use chapters from various textbooks. All textbook material will be available digitally, some of which on Blackboard:
  - 1. SQL
    - **SAMS**: Forta. SAMS Teach yourself SQL in 10min. 4th ed. [Safari books eBook (NEU free online access)], [EBSCOhost eBook (NEU free online access)], [EBSCOhost eBook (NEU free online access, but old edition)], [Amazon (30$)]
  - 2. Database design
    - **Watt**: Adrienne Watt, Database design. Online textbook. 2nd ed: Easy quick read with short descriptions of key concepts. However (!) this book contains a few inconsistencies or even mistakes. Thus, this is an optional read. Here is a list of problems: 1) ERDs do not contain FKs (FKs are a concept reserved for the relationsl model); 2) The concepts of cardinaltiy, connectivity, participation are all mixed up and different from most textbooks, including our slides.
    - **Hoffer, Ramesh, Topi**: Sect 2: Data modeling (ERDs)
    - **Gillenson**: Ch 7: Logical database design
    - **Powell**: Sect 4: Normalization
  - 3. Transactions
    - **Elsmari**: Ch 21: Transactions
    - **Silberschatz**: Ch 15 Ch16: Concurrency and Recovery

Updated Schedule

		Database Design and Normal Forms		
8	M Oct 1	Database Design: ER Diagrams		HW3
9	R Oct 4	**Exam 1** Database Design: ER Diagrams		Q5 (Oct 18)
	M Oct 8	No class: Columbus Day		
10	R Oct 11	Database Design: ER Diagrams	**Hoffer**: Sect 2	Q6 (Oct 18)
11	M Oct 15	Database Design: Relations	**Watt**: Ch 7-10	
12	R Oct 18	Database Design: Relations	**Gillenson**: Ch 7	Q7 (Oct 25), HW4
13	M Oct 22	Database Design: Normalization and Decompositions	**Watt**: Ch 11-12	
14	R Oct 25	Database Design: Normalization and Decompositions	**Powell**: Sect 4	Q8 (Nov 1), HW5
		Transaction Processing		
15	M Oct 29	Transactions	**Elmasri**: Ch 21	
16	R Nov 1	Concurrency		Q9, HW6
17	M Nov 5	**Exam 2** Concurrency		
18	R Nov 8	Recovery	**Silberschatz**: Ch 15, 16	Q10
	M Nov 12	No class: Veteran's Day		HW7 (due 11/13)
		NoSQL		
19	R Nov 15	NoSQL		Q11

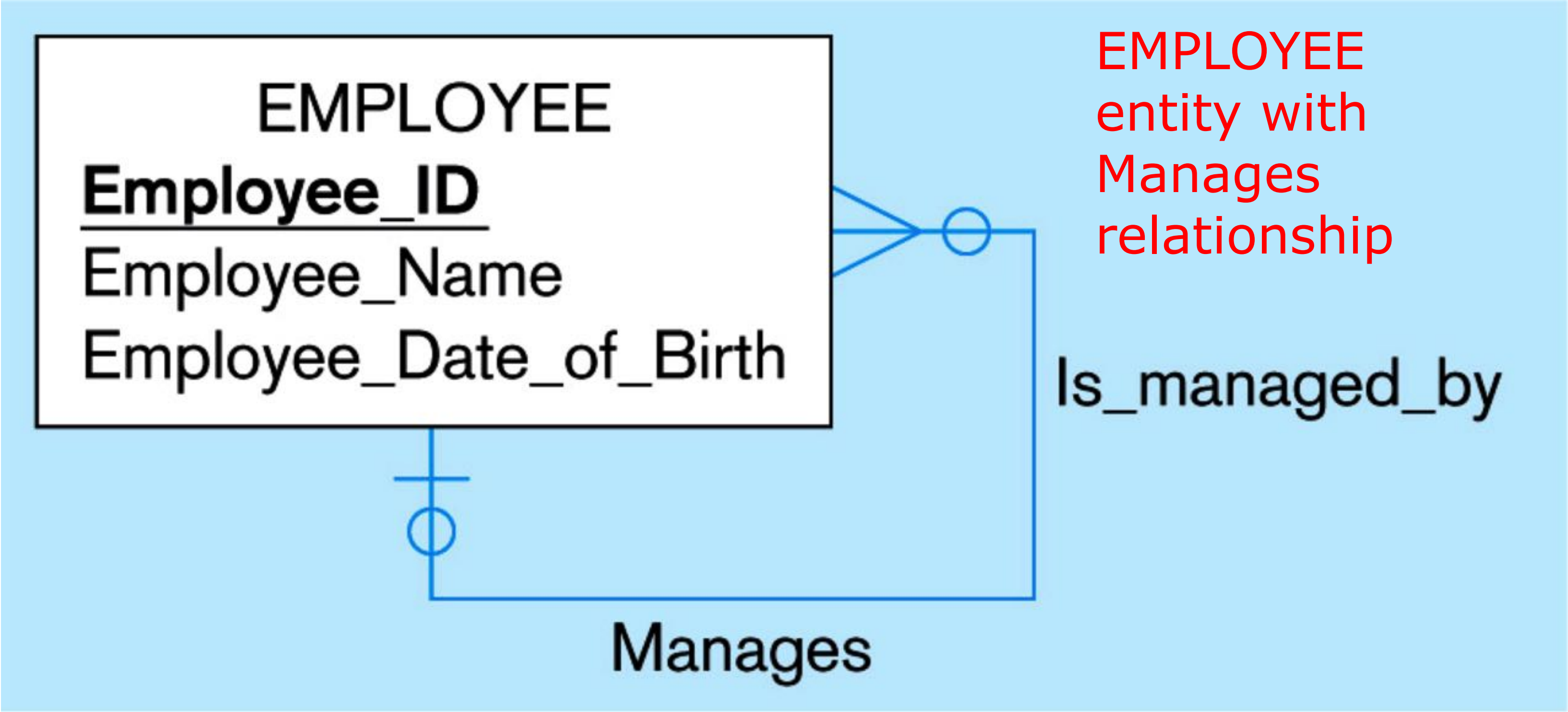
Relational Modeling: Unary Relationships

Mapping Unary Relationships

- 1) One-to-Many
  - Create a recursive foreign key in the same relation

- 2) Many-to-Many – Create two relations:
  - One for the entity type
  - One for an associative relation in which the primary key has two attributes, both taken from the primary key of the entity

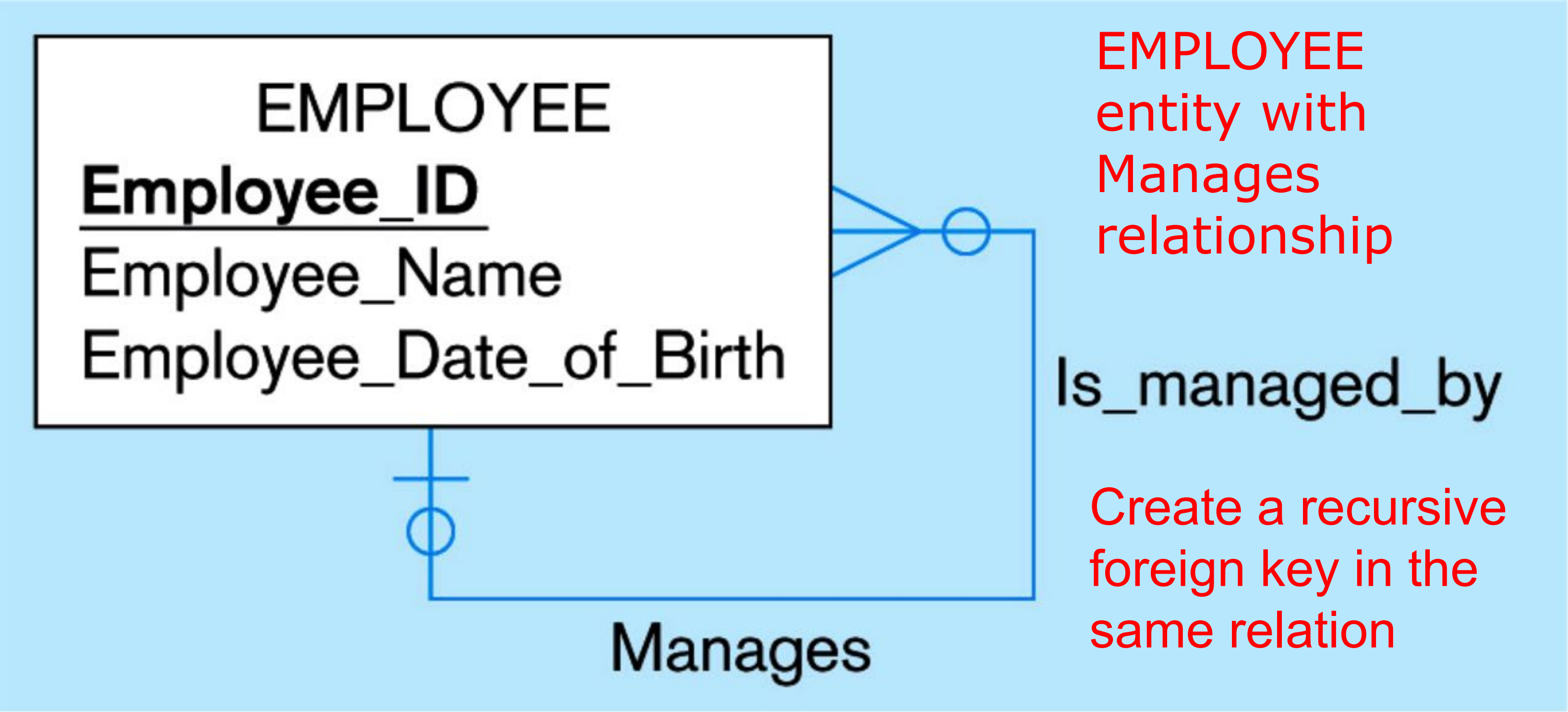
1) Mapping a Unary 1:N Relationship



EMPLOYEE entity with Manages relationship

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

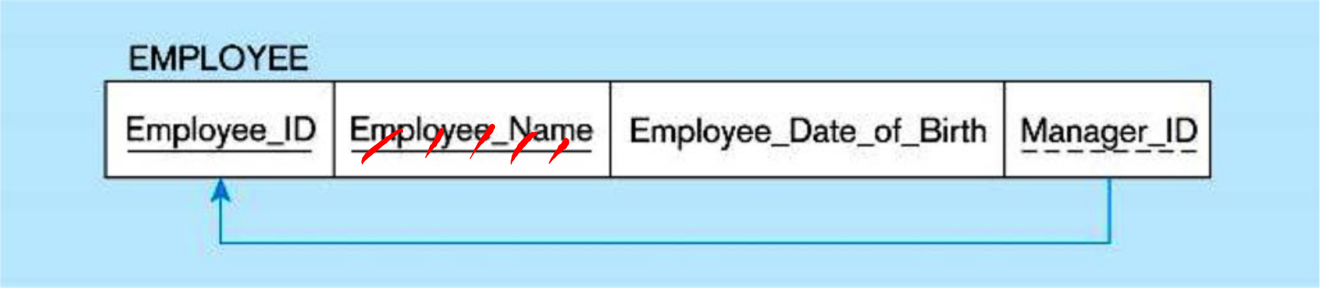
1) Mapping a Unary 1:N Relationship



EMPLOYEE entity with Manages relationship

Create a recursive foreign key in the same relation

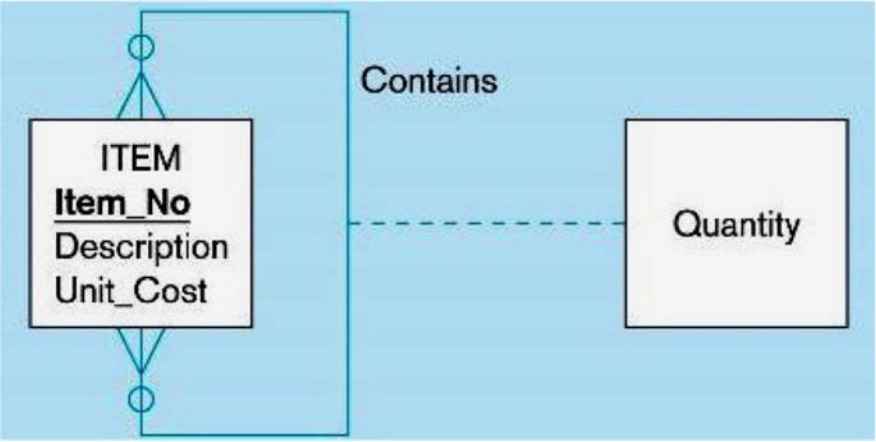
EMPLOYEE relation with recursive foreign key



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

2) Mapping a Unary M:N Relationship

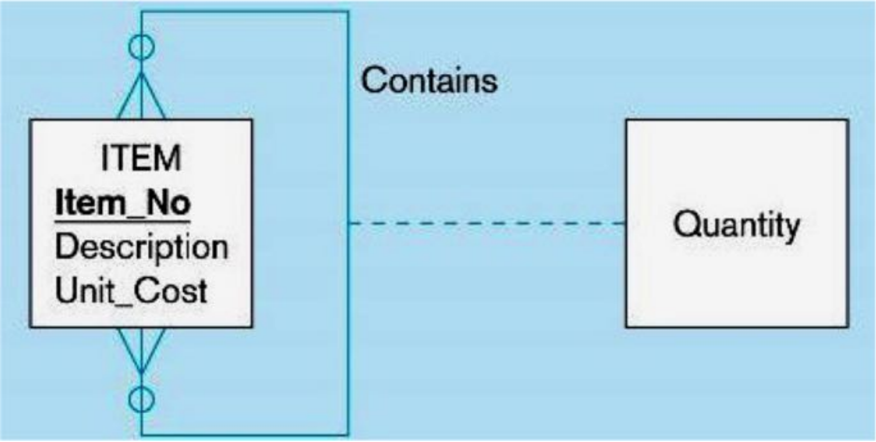
Bill-of-materials relationships (M:N)



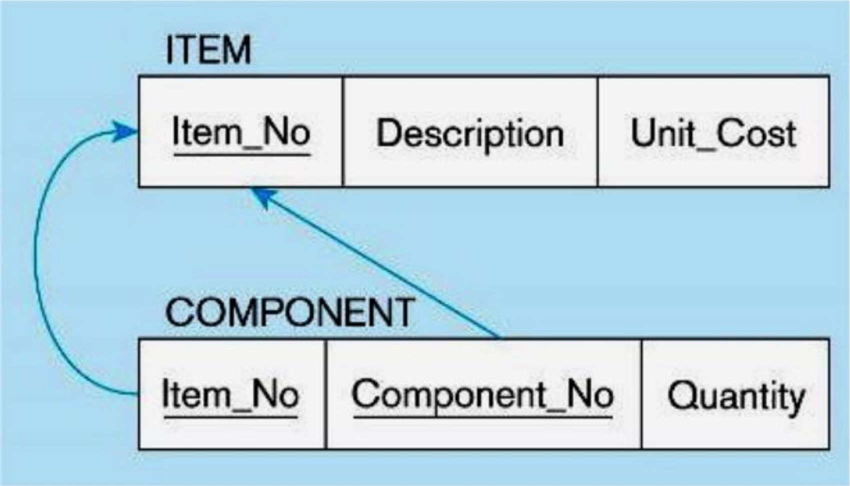
Source: Hoffer, Ramesh, Topi, "Modern database management," 10th ed, 2010.

2) Mapping a Unary M:N Relationship

Bill-of-materials relationships (M:N)



ITEM and COMPONENT relations



Create Two relations:

- One for the entity type
- One for an associative relation in which the primary key has two attributes, both taken from the primary key of the entity

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

Relational Modeling: Associative Entities

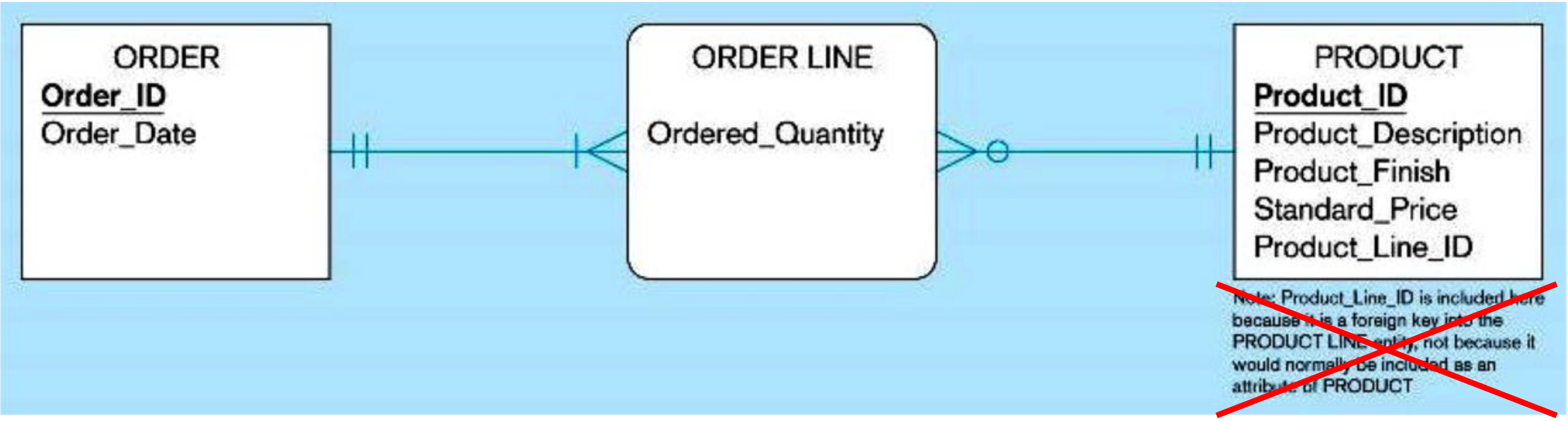
Mapping Associative Entities

- Rules for two scenarios:

- A) Identifier Not Assigned
  - Default primary key for the association relation is composed of the primary keys of the two entities (as in M:N relationship)

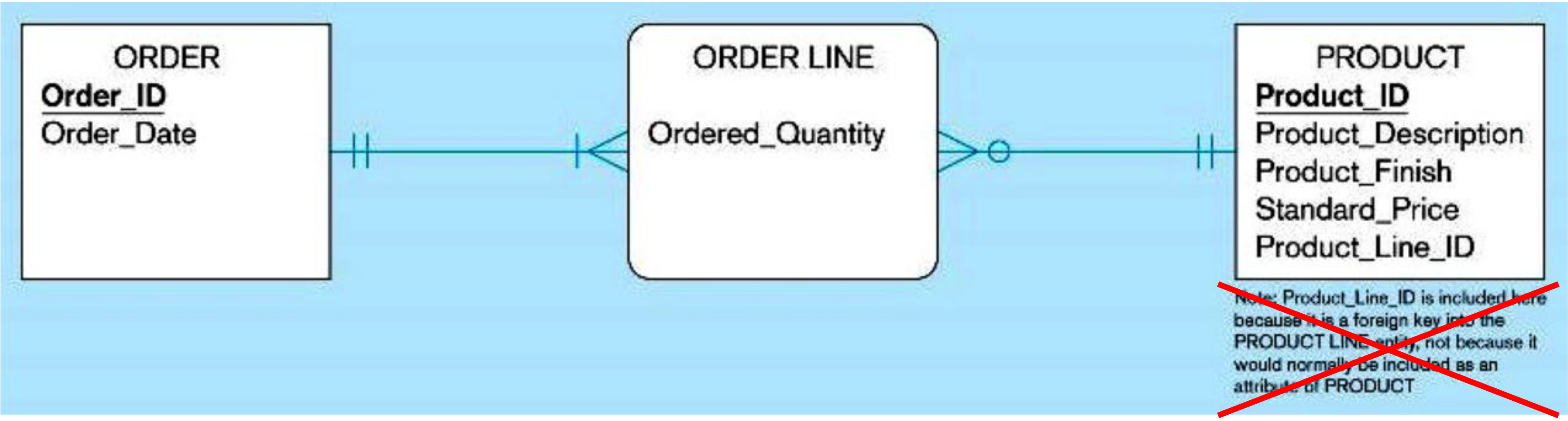
- B) Identifier Assigned
  - It is natural and familiar to end-users
  - Default identifier may not be unique

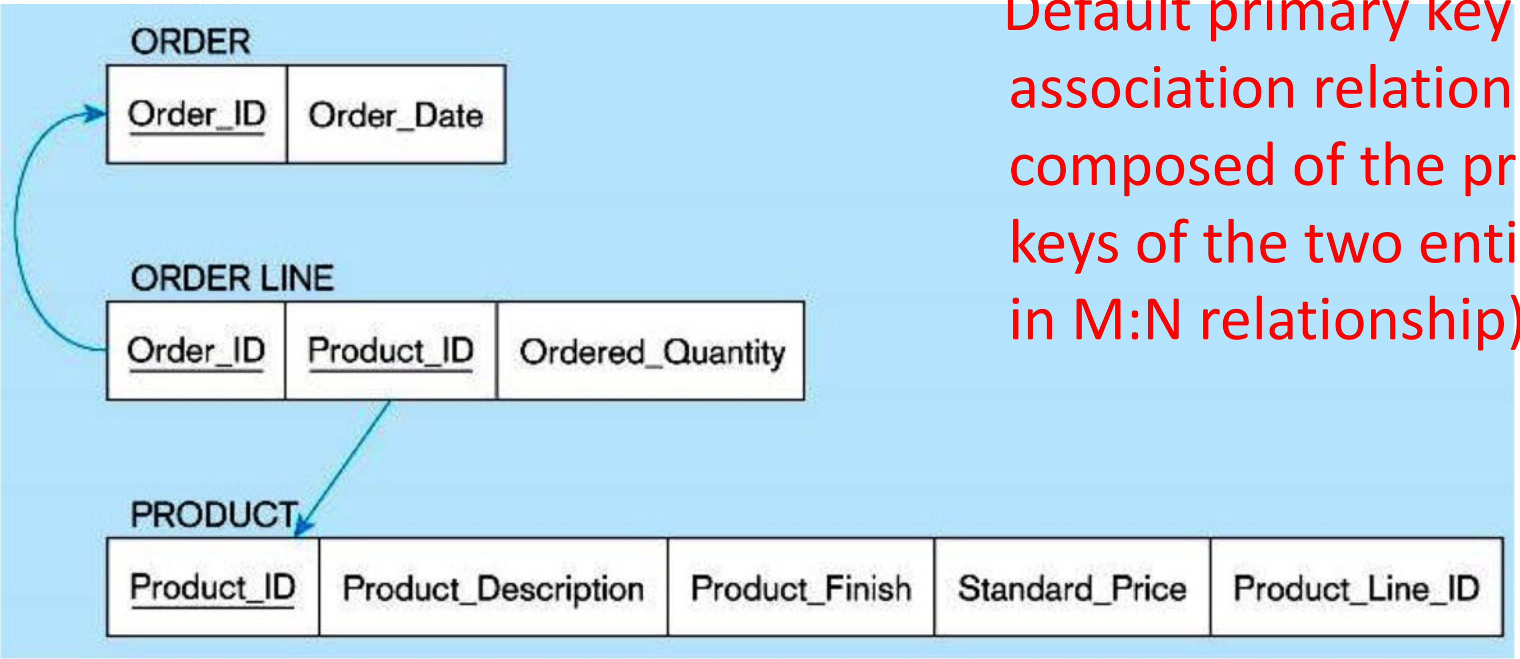
A) Associative Entity Relations (No Identifier)



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

A) Associative Entity Relations (No Identifier)





Default primary key for the association relation is composed of the primary keys of the two entities (as in M:N relationship)

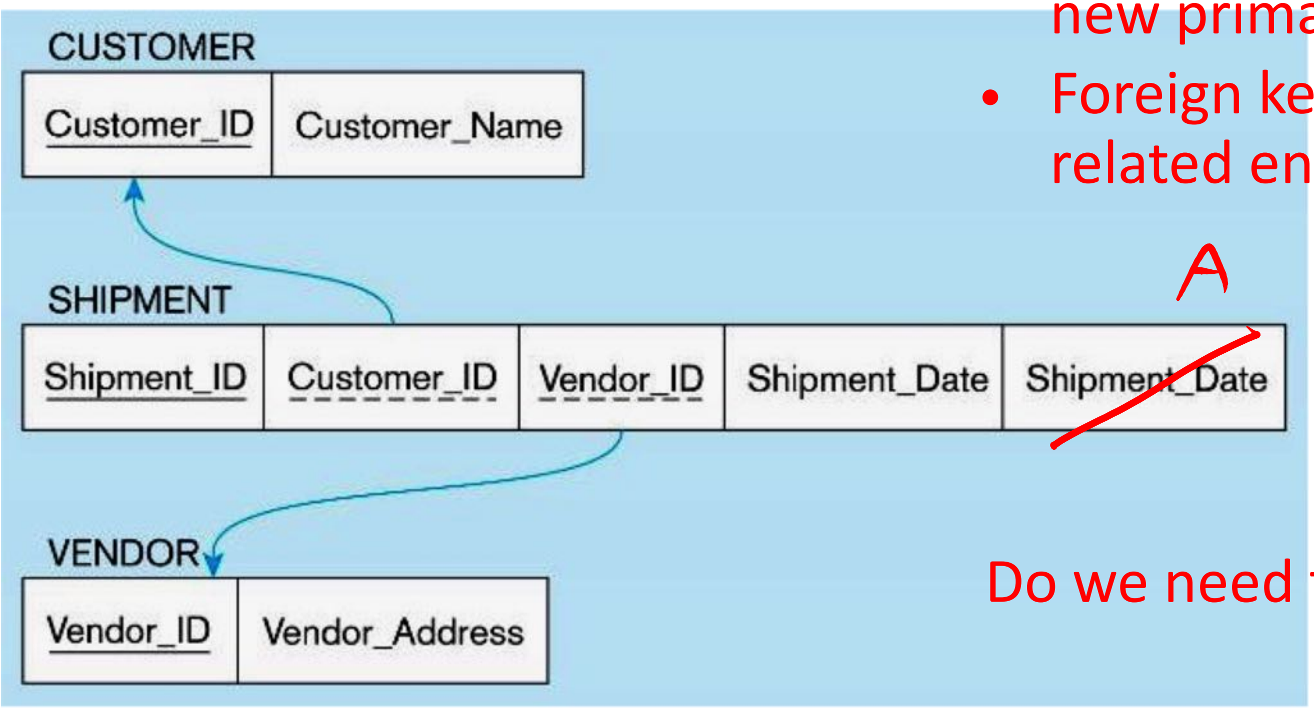
Source: Hoffer, Ramesh, Topi, "Modern database management," 10th ed, 2010.

B) Associative Entity Relations (With Identifier)

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

B) Associative Entity Relations (With Identifier)

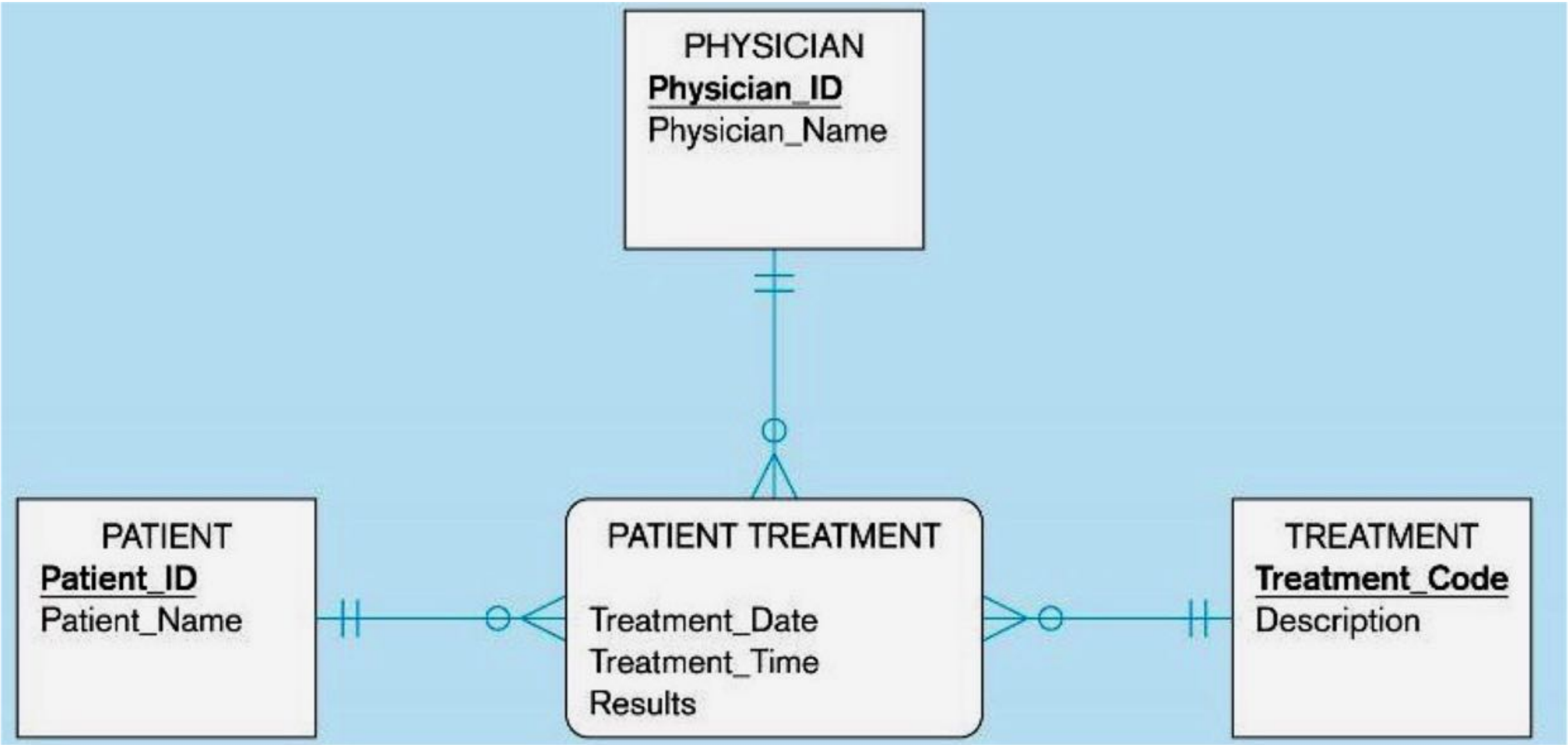
- Identifier attribute becomes new primary key in relation
- Foreign keys reference all related entities



Do we need the key?

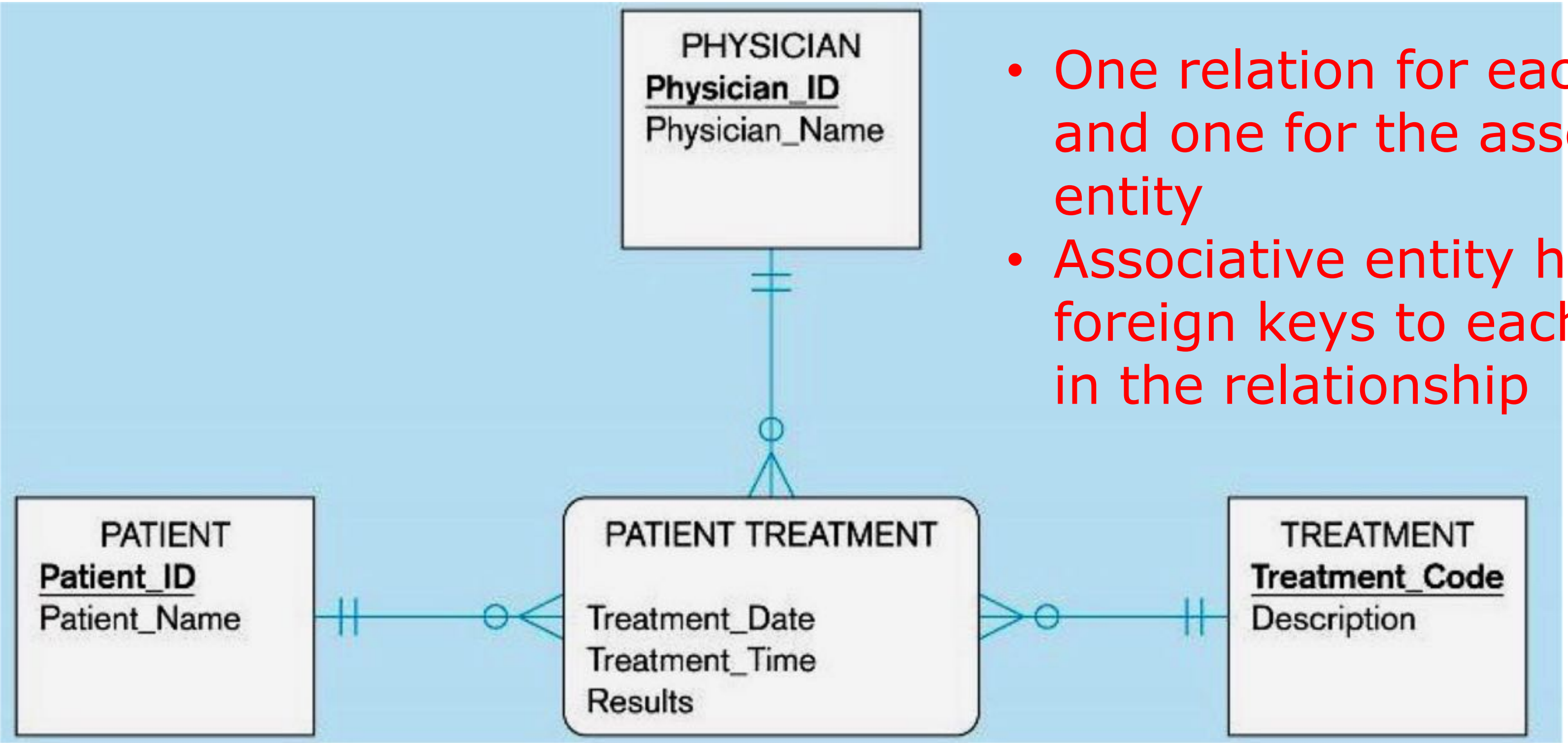
Source: Hoffer, Ramesh, Topi, "Modern database management," 10th ed, 2010.

Mapping ternary relationship w/ associative entity

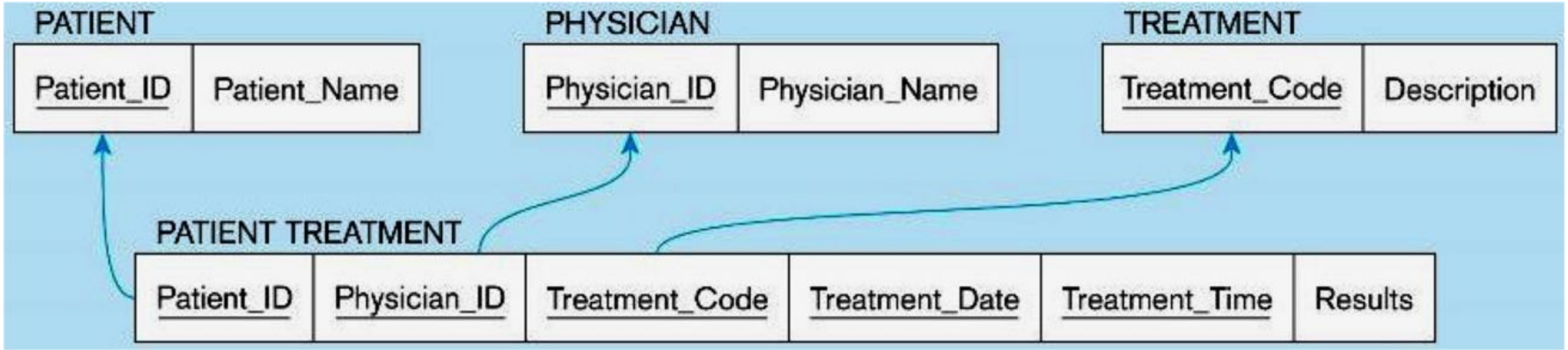


Source: Hoffer, Ramesh, Topi, "Modern database management," 10th ed, 2010.

Mapping ternary relationship w/ associative entity



- One relation for each entity and one for the associative entity
- Associative entity has foreign keys to each entity in the relationship



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

Relational Modeling: Weak entities

Mapping Weak Entities

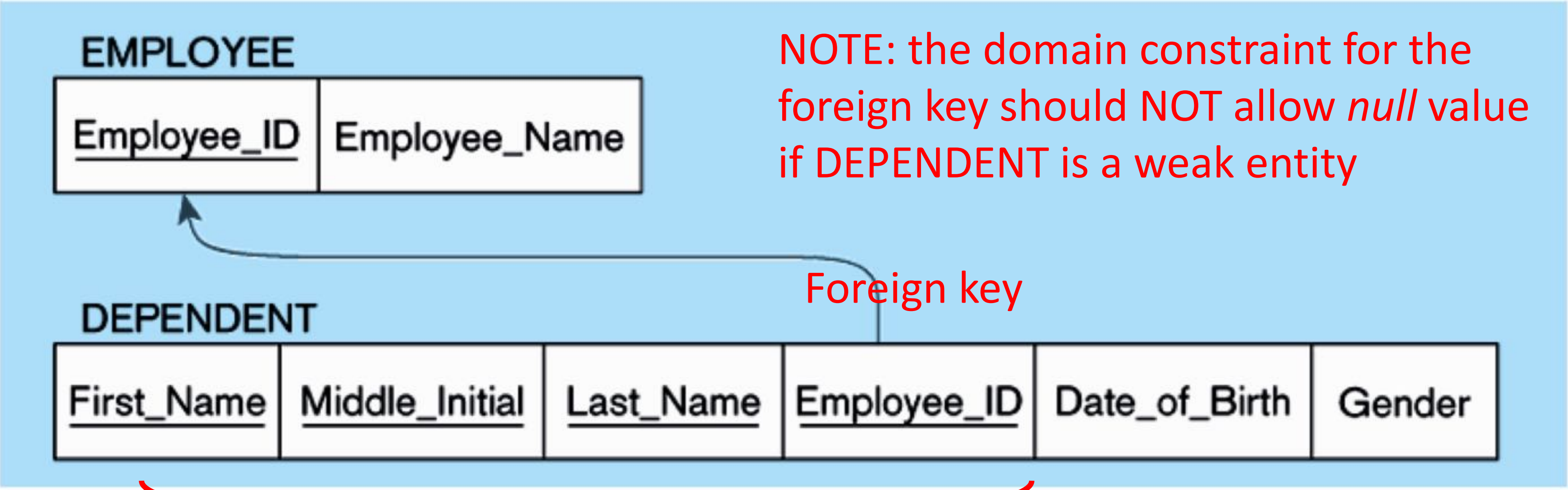
- Weak Entities become separate relations with a foreign key taken from the superior entity

- Primary key composed of:
  - Partial identifier of weak entity
  - Primary key of identifying relation (strong entity)

Example: Mapping A Weak Entity (Relations)

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

Example: Mapping A Weak Entity (Relations)



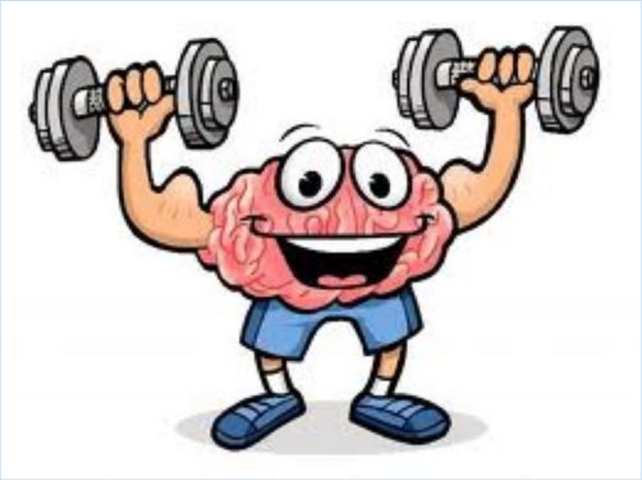
Composite primary key

or "Surrogate primary key" *(George Foreman …)*

DEPENDENT(Dependent#, EmployeeID, FirstName, MiddleInitial, LastName, DateOfBirth, Gender)

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

Practice



Exercise 1

- Create a relational schema to represent the following E-R Diagram:

Product

<u>Product_ID</u>
Product_Name
{Price History
(Effective_Date,
Price) }

Exercise 1

- Create a relational schema to represent the following E-R Diagram:

Product
Product_ID
Product_Name
{Price History
(Effective_Date,
Price) }

Product

Product_ID	Product_Name

Price_History

Product_ID	Effective_Date	Price

Exercise 2

- Create a relational schema to represent the following E-R Diagram:

Product
<u>Product_ID</u>
Product_Name
StandardPrice
{(customer,
Price)}

Product

<u>Product_ID</u>	Product_Name

Exercise 2

- Create a relational schema to represent the following E-R Diagram:

Product
Product_ID
Product_Name
StandardPrice
{(customer,
Price)}

Product

Product_ID	Product_Name	Standard_Price

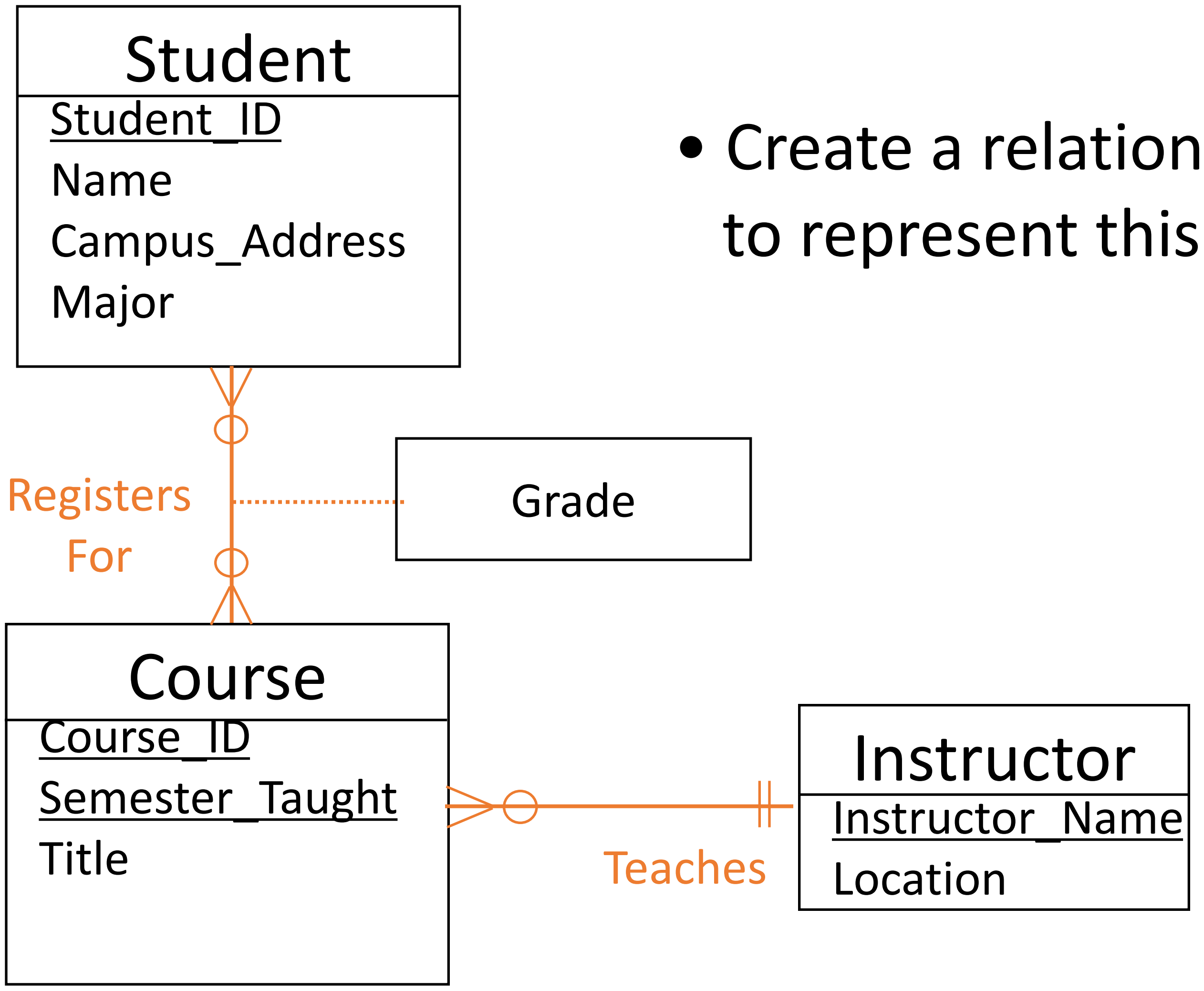
Customer

CustomerName	...

SpecialPrice

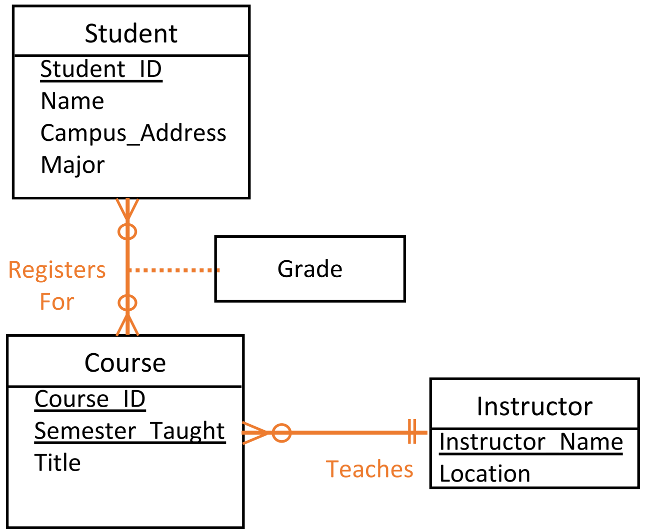
Product_ID	customer	Price

Exercise 3



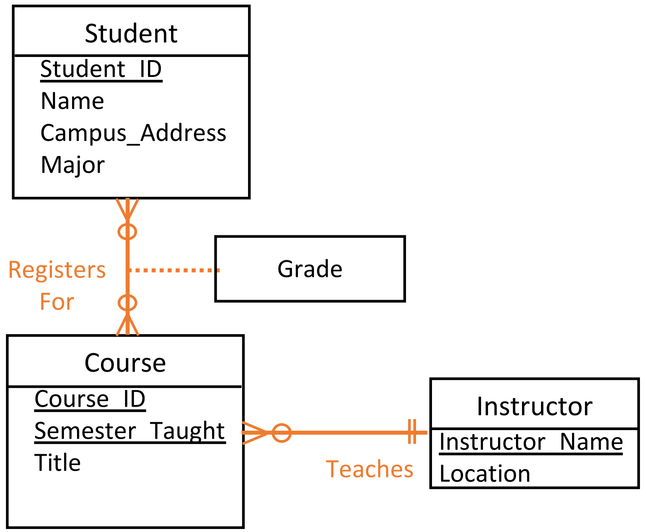
- Create a relational schema to represent this E-R Diagram:

Exercise 3



Student
Student_ID
Name
Campus_Address
Major
Registers
For
Grade
Course
Course_ID
Semester_Taught
Title
Teaches
Instructor
Instructor_Name
Location

Exercise 3: Solution



Student

Student_ID	Name	Campus_Address	Major

Course_Registration

@Student_ID	@Course_ID	@Semester_Taught	Grade

Course

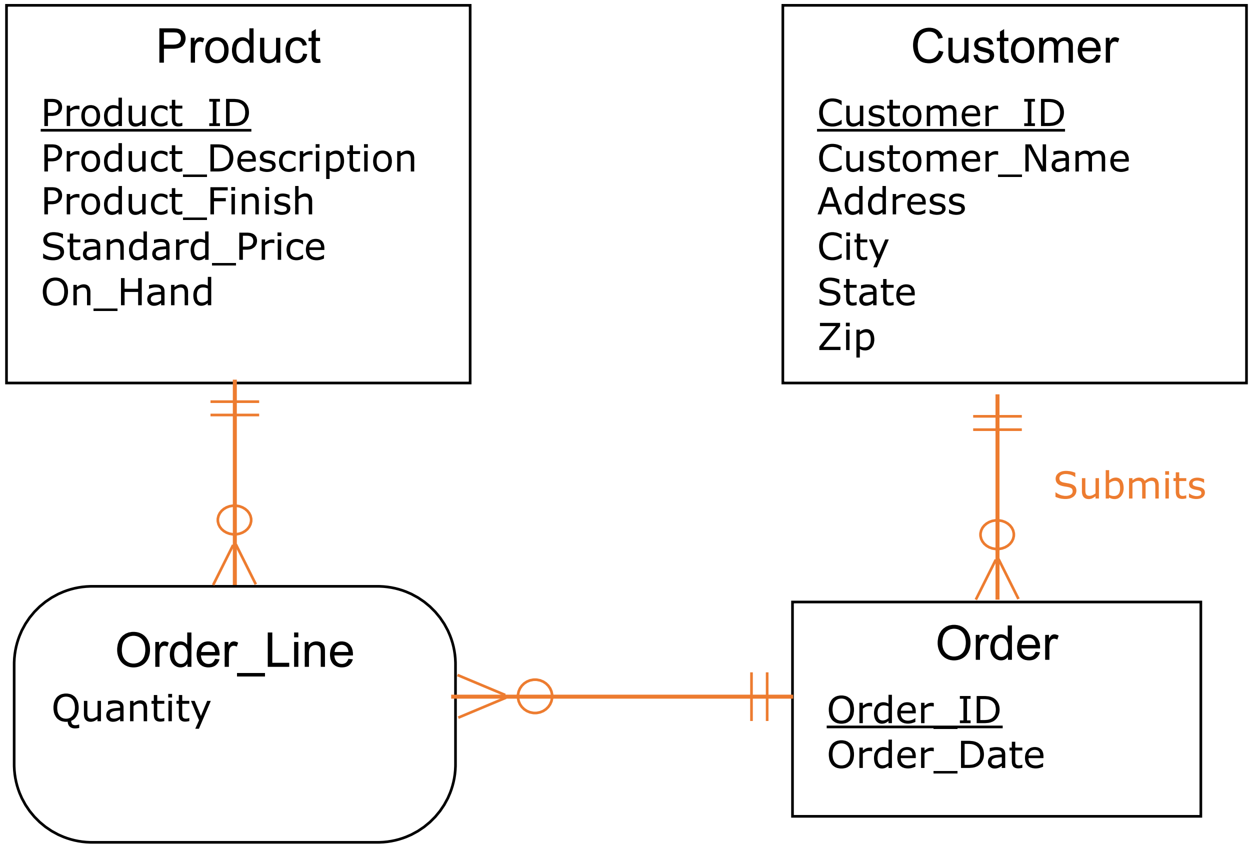
Course_ID	Semester_Taught	Title	@Instructor_Name

Instructor

Instructor_Name	Location

don't forget:
"not null" constraint

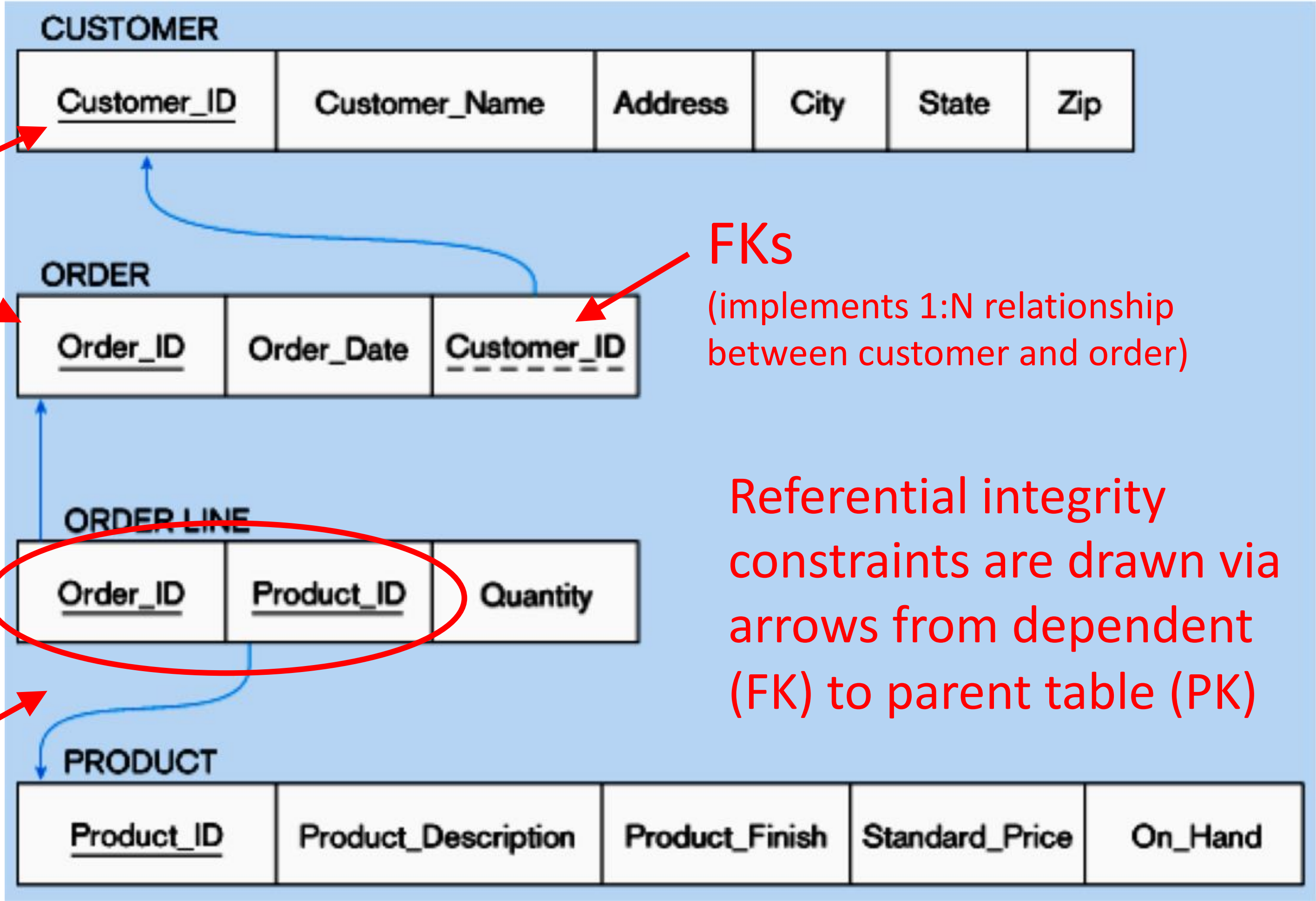
Example: Pine Valley Furniture Company



Example: Pine Valley Furniture Company

PKs

Combined, these are a ***composite primary key*** (uniquely identifies the order line)…individually they are ***foreign keys*** (together implement M:N relationship between order and product)



FKs

(implements 1:N relationship between customer and order)

Referential integrity constraints are drawn via arrows from dependent (FK) to parent table (PK)

Source: Compare with Fig 4-30: Hoffer, Ramesh, Topi, "Modern database management," 10th ed, 2010.

Overview
Database normalization & Design Theory

Normalization

- Understand the <u>normalization</u> process and why a normalized data model is desirable (no redundancy)
- Be able to explain <u>normal forms</u> and identify when a relational model is in any of them
- Be able to explain <u>anomalies</u> and how to avoid them
  - Insertion, deletion, and modification
- Actually apply normalization ☺

Normalization

- Organizing data to <u>minimize redundancy</u> (repeated data)

- This is good for two reasons
  - The database takes up less space
  - You have a lower chance of inconsistencies in your data

- If you want to make a change to a record, you only have to make it in one place
  - The relationships take care of the rest

- But you will usually need to link the separate tables together in order to retrieve information

First Normal Form (1NF)

- A database schema is in *First Normal Form* if all tables are flat (no "nested relations")

Student

Name	GPA	Course
Alice	3.8	Math DB OS
Bob	3.7	DB OS
Carol	3.9	Math OS

⇨ ?

First Normal Form (1NF)

- A database schema is in *First Normal Form* if all tables are flat (no "nested relations")

Student

Name	GPA	Course
Alice	3.8	Math DB OS
Bob	3.7	DB OS
Carol	3.9	Math OS

Student

Name	GPA	Course
Alice	3.8	Math
Alice	3.8	DB
Alice	3.8	OS
Bob	3.7	DB
Bob	3.7	OS
Carol	3.9	Math
Carol	3.9	OS

First Normal Form (1NF)

- A database schema is in *First Normal Form* if all tables are flat (no "nested relations")

Student

Name	GPA	Course
Alice	3.8	Math DB OS
Bob	3.7	DB OS
Carol	3.9	Math OS

May need to add keys

Student

<u>Name</u>	GPA
Alice	3.8
Bob	3.7
Carol	3.9

Takes

Student	Course
Alice	Math
Carol	Math
Alice	DB
Bob	DB
Alice	OS
Carol	OS

Course

<u>Course</u>
Math
DB
OS

Data Anomalies

- When a database is poorly designed we get anomalies (those are bad) resulting from redundancies:
  - Update anomalies: need to change in several places
  - Insert anomalies: need to repeat data for new inserts
  - Deletion anomalies: may lose data when we don't want

Relational Schema Design

Recall multivalued (set) attributes (persons with several phones):

Employee

Name	<u>SSN</u>	<u>PhoneNumber</u>	City
Fred	123-45-6789	412-555-1234	Boston
Fred	123-45-6789	412-555-6543	Boston
Joe	987-65-4321	908-555-2121	Cambridge

- One person may have multiple phones, but lives in only one city
- Primary key is thus (SSN, PhoneNumber)

Do you see any anomalies?

Relational Schema Design

Recall multivalued (set) attributes (persons with several phones):

Employee

Name	SSN	PhoneNumber	City
Fred	123-45-6789	412-555-1234	Boston
Fred	123-45-6789	412-555-6543	Boston
Joe	987-65-4321	908-555-2121	Cambridge

- One person may have multiple phones, but lives in only one city
- Primary key is thus (SSN, PhoneNumber)

Do you see any anomalies?

- **Update anomalies**: what if Fred moves to "New York"?
- **Insert anomalies**: what if Joe gets a second phone number
- **Deletion anomalies**: what if Joe deletes his phone number?
  (what if Joe had no phone #)

What do we do????

Relation Decomposition

Break the relation into two:

Employee

Name	SSN	PhoneNumber	City
Fred	123-45-6789	412-555-1234	Boston
Fred	123-45-6789	412-555-6543	Boston
Joe	987-65-4321	908-555-2121	Cambridge

Employee

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

Phone

SSN	PhoneNumber
123-45-6789	412-555-1234
123-45-6789	412-555-6543
987-65-4321	908-555-2121

Anomalies have gone:

- No more repeated data
- Easy to move Fred to "New York" (how ?)
- Easy to delete all Joe's phone numbers (how ?)

Good News / Bad News

- The good news: when you start with solid ER modeling and follow the steps described to create relations then your relations will usually be pretty well normalized

- The bad news: you often don't have the benefit of starting from a good ER model.

- The good news (part 2): the steps we will cover in class will help you convert poorly normalized tables into highly normalized tables

1. Normal forms and Functional Dependencies

Design Theory

- Design theory is about how to represent your data to avoid anomalies.

- It is a mostly mechanical process
  - Tools can carry out routine portions

- We have a notebook implementing all algorithms!
    - We'll play with it in the activities!

Data Normalization

- Data normalization is the process of decomposing relations with anomalies to produce smaller, well-structured relations

- Goals of normalization include:
  - Minimize data redundancy
  - Simplifying the enforcement of referential integrity constraints
  - Simplify data maintenance (inserts, updates, deletes)
  - Improve representation model to match "the real world"

Well-Structured Relations

- A <u>well-structured relation</u> contains minimal data redundancy and allows users to insert, delete, and update rows without causing data inconsistencies

- <u>Anomalies</u> are errors or inconsistencies that may result when a user attempts to update a table that contains redundant data.

- Three types of anomalies:
  - <u>Insertion Anomaly</u> – adding new rows forces user to create duplicate data
  - <u>Deletion Anomaly</u> – deleting rows may cause a loss of data that would be needed for other future rows
  - <u>Modification Anomaly</u> – changing data in a row forces changes to other rows because of duplication

- General rule of thumb: a table should not pertain to more than one entity type

Normal Forms

Normal Form: a state of a relation that results from applying simple rules regarding FDs to that relation

- 1st Normal Form (1NF) = All tables are flat
- 2nd Normal Form = not used anymore
  - no more "partial FDs" (those are part of the "bad" FDs)
- **3rd Normal Form (3NF)**
  - **no more transitive FDs (also "bad")**
- **Boyce-Codd Normal Form (BCNF)**
  - **every determinant is a candidate key**

DB designs based on FDs (*functional dependencies*), intended to prevent data ***anomalies***

Our focus next

- 4th: any multivalued dependencies have been removed (we will give intuition)
- 5th: any remaining anomalies have been removed (not covered)

1st Normal Form (1NF)

Student	Courses
Mary	{CS3200, CS4240}
Joe	{CS3200, CS4240}
…	…

Violates 1NF.

1st Normal Form (1NF)

Student	Courses
Mary	{CS3200, CS4240}
Joe	{CS3200, CS4240}
…	…

Violates 1NF.

Student	Courses
Mary	CS3200
Mary	CS4240
Joe	CS3200
Joe	CS4240

In 1st NF

1NF Constraint: Types must be atomic!

Constraints Prevent (some) Anomalies in the Data

A poorly designed database causes *anomalies*:

Student	Course	Room
Mary	CS3200	WVF20
Joe	CS3200	WVF20
Sam	CS3200	WVF20
..	..	..

If every course is in only one room, contains ***redundant*** information!

Constraints Prevent (some) Anomalies in the Data

A poorly designed database causes ***anomalies***:

Student	**Course**	**Room**
Mary	CS3200	WVF20
Joe	CS3200	B12
Sam	CS3200	WVF20
..	..	..

If we update the room number for one tuple, we get inconsistent data = an ***update* anomaly**

Constraints Prevent (some) Anomalies in the Data

A poorly designed database causes ***anomalies***:

Student	Course	Room
…	…	…

If everyone drops the class, we lose what room the class is in! = a ***delete*** **anomaly**

Constraints Prevent (some) Anomalies in the Data

A poorly designed database causes *anomalies*:

...	CS4240	B12

Student	Course	Room
Mary	CS3200	WVF20
Joe	CS3200	WVF20
Sam	CS3200	WVF20
..	..	..

Similarly, we can't reserve a room without students = an ***insert* anomaly**

Constraints Prevent (some) Anomalies in the Data

Student	Course
Mary	CS3200
Joe	CS3200
Sam	CS3200
..	..

Course	Room
CS3200	WVF20
CS4240	B12

Is this form better?

- Redundancy?
- Update anomaly?
- Delete anomaly?
- Insert anomaly?

Next: develop theory to understand why this design may be better **and** how to find this *decomposition*…

StaffBranch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

StaffBranch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

Staff

staffNo	sName	position	salary	branchNo
SL21	John White	Manager	30000	B005
SG37	Ann Beech	Assistant	12000	B003
SG14	David Ford	Supervisor	18000	B003
SA9	Mary Howe	Assistant	9000	B007
SG5	Susan Brand	Manager	24000	B003
SL41	Julie Lee	Assistant	9000	B005

Branch

branchNo	bAddress
B005	22 Deer Rd, London
B007	16 Argyll St, Aberdeen
B003	163 Main St, Glasgow

Is This Table Well Structured?

EMPLOYEE2

Emp_ID	Name	Dept_Name	Salary	Course_Title	Date_Completed
100	Margaret Simpson	Marketing	48,000	SPSS	6/19/200X
100	Margaret Simpson	Marketing	48,000	Surveys	10/7/200X
140	Alan Beeton	Accounting	52,000	Tax Acc	12/8/200X
110	Chris Lucero	Info Systems	43,000	SPSS	1/12/200X
110	Chris Lucero	Info Systems	43,000	C++	4/22/200X
190	Lorenzo Davis	Finance	55,000		
150	Susan Martin	Marketing	42,000	SPSS	6/19/200X
150	Susan Martin	Marketing	42,000	Java	8/12/200X

- Does it contain anomalies?

Is This Table Well Structured?

EMPLOYEE2

Emp_ID	Name	Dept_Name	Salary	Course_Title	Date_Completed
100	Margaret Simpson	Marketing	48,000	SPSS	6/19/200X
100	Margaret Simpson	Marketing	48,000	Surveys	10/7/200X
140	Alan Beeton	Accounting	52,000	Tax Acc	12/8/200X
110	Chris Lucero	Info Systems	43,000	SPSS	1/12/200X
110	Chris Lucero	Info Systems	43,000	C++	4/22/200X
190	Lorenzo Davis	Finance	55,000		
150	Susan Martin	Marketing	42,000	SPSS	6/19/200X
150	Susan Martin	Marketing	42,000	Java	8/12/200X

- Does it contain anomalies?
  - Insertion: if an employee takes a new class we need to add duplicate data (Name, Dept_Name, Salary)
  - Deletion: If we remove employee 140, we lose information about the existence of a Tax Acc class
  - Modification: Giving a salary increase to employee 100 forces us to update multiple records

- Why do these anomalies exist?

Is This Table Well Structured?

EMPLOYEE2

Emp_ID	Name	Dept_Name	Salary	Course_Title	Date_Completed
100	Margaret Simpson	Marketing	48,000	SPSS	6/19/200X
100	Margaret Simpson	Marketing	48,000	Surveys	10/7/200X
140	Alan Beeton	Accounting	52,000	Tax Acc	12/8/200X
110	Chris Lucero	Info Systems	43,000	SPSS	1/12/200X
110	Chris Lucero	Info Systems	43,000	C++	4/22/200X
190	Lorenzo Davis	Finance	55,000		
150	Susan Martin	Marketing	42,000	SPSS	6/19/200X
150	Susan Martin	Marketing	42,000	Java	8/12/200X

- Does it contain anomalies?
  - Insertion: if an employee takes a new class we need to add duplicate data (Name, Dept_Name, Salary)
  - Deletion: If we remove employee 140, we lose information about the existence of a Tax Acc class
  - Modification: Giving a salary increase to employee 100 forces us to update multiple records

- Why do these anomalies exist?
  - Because there are <u>two themes (entity types) in one relation</u>. This results in duplication, and an unnecessary dependency between the entities

Normalizing Previous Employee/Class Table

Employee

Emp_ID	Name	Dept_Name	Salary
100	Margaret Simpson	Marketing	48000
140	Alan Beeton	Accounting	52000
110	Chris Lucero	Info Sys	43000
190	Lorenzo Davis	Finance	55000
150	Susan Martin	Marketing	42000

This seems more complicated

Why might this approach be superior to the previous one?

Course_Completion

Emp_ID	Course_ID	Date_Completed
100	1	6/19/2005
100	2	10/7/2004
140	3	12/8/2004
110	1	1/12/2004
110	4	4/22/2003
150	1	6/19/2005
150	5	8/12/2002

Course

Course_ID	Course_Title
1	SPSS
2	Surveys
3	Tax Acc
4	C++
5	Java

Functional Dependencies ("FDs")

Definition:

If two tuples agree on the attributes

$A_1, A_2, \ldots, A_n$

then they must also agree on the attributes

$B_1, B_2, \ldots, B_m$

Formally:

$A_1, A_2, \ldots, A_n \rightarrow B_1, B_2, \ldots, B_m$

Functional Dependencies ("FDs")

Def: Let A,B be *sets* of attributes
We write $A \rightarrow B$ or say A *functionally determines* B if, for any tuples t_1 and t_2:

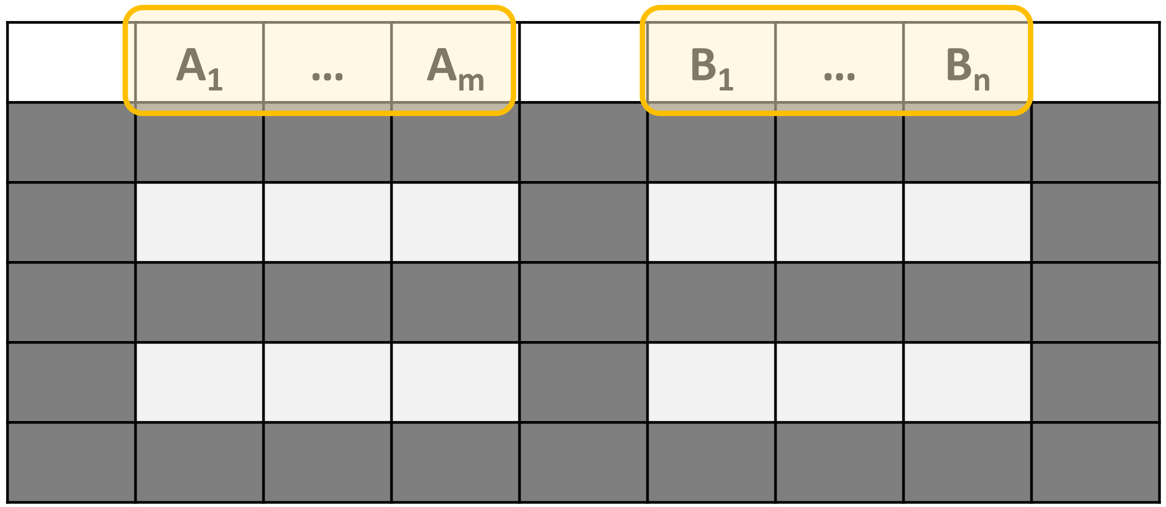
$$t_1[A] = t_2[A] \text{ implies } t_1[B] = t_2[B]$$

and we call $A \rightarrow B$ a <u>functional dependency</u>

A (<u>determinant</u>) $\rightarrow$ B (<u>dependent</u>)

$A \rightarrow B$ means that
"whenever two tuples agree on A then they agree on B."

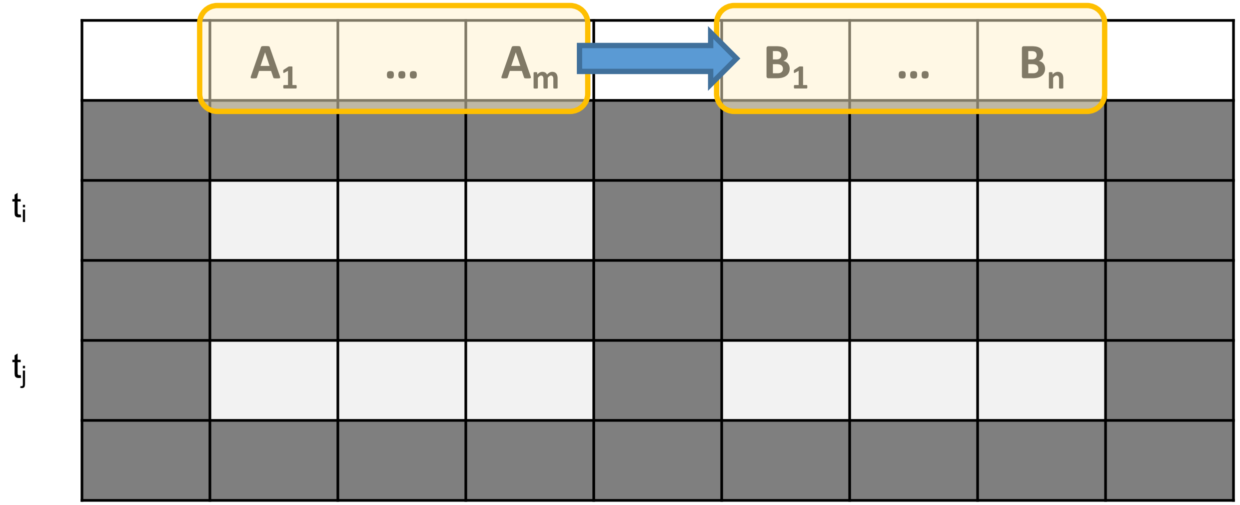
A Picture Of FDs



Defn (again):

Given attribute sets $A=\{A_1,\ldots,A_m\}$ and $B = \{B_1,\ldots B_n\}$ in R,

A Picture Of FDs

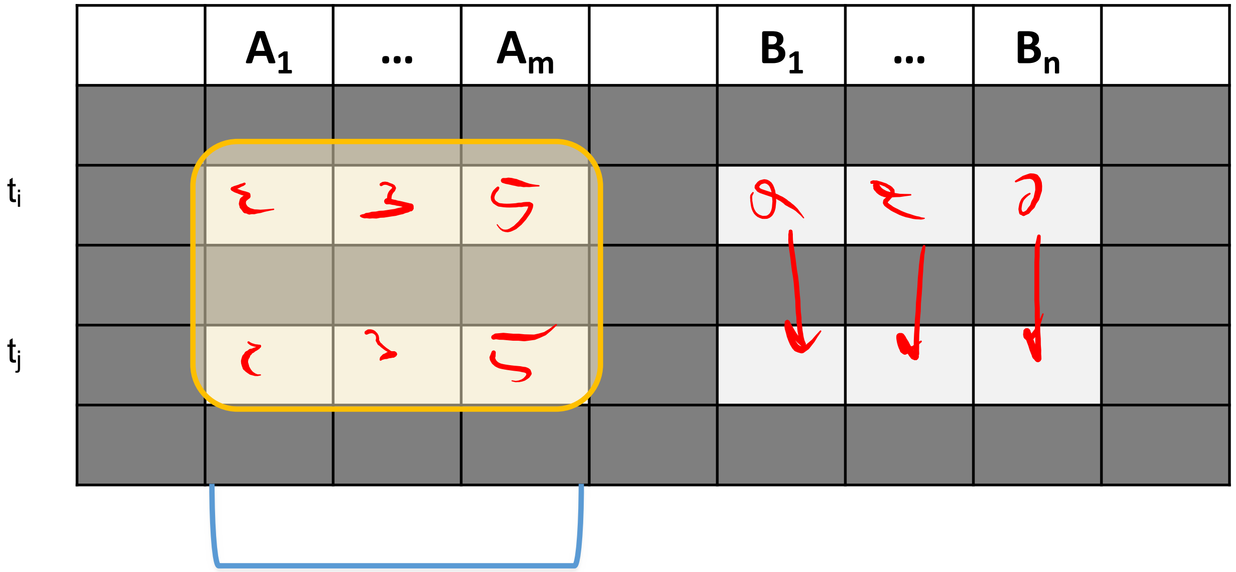


Defn (again):

Given attribute sets $A=\{A_1,\ldots,A_m\}$ and $B = \{B_1,\ldots B_n\}$ in R,

The *functional dependency* $A \rightarrow B$ on R holds if for *any* t_i,t_j in R:

A Picture Of FDs



If t_i,t_j agree here..

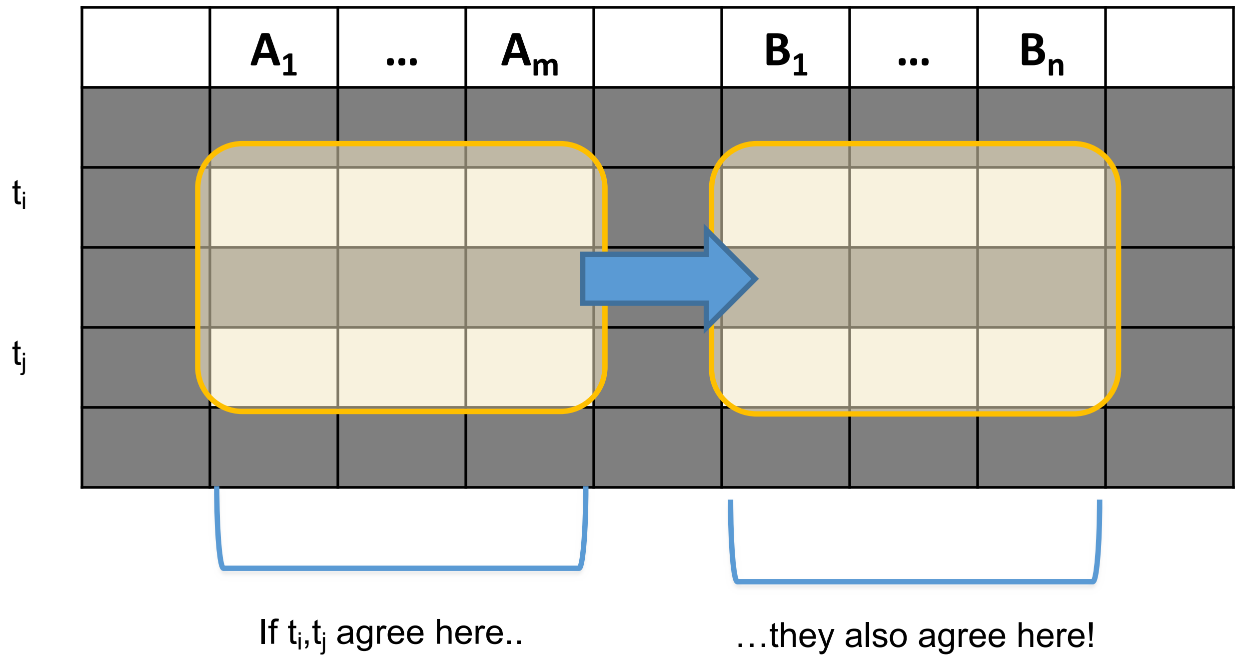
Defn (again):

Given attribute sets **A={A_1,…,A_m}** and **B = {B_1,…B_n}** in **R**,

The *functional dependency* **A→ B** on **R** holds if for *any* t_i,t_j in R:

if $t_i[A_1] = t_j[A_1]$ AND $t_i[A_2]=t_j[A_2]$ AND … AND $t_i[A_m] = t_j[A_m]$

A Picture Of FDs



<u>Defn (again):</u>
Given attribute sets $A=\{A_1,\ldots,A_m\}$ and $B = \{B_1,\ldots B_n\}$ in R,

The *functional dependency* $A \rightarrow B$ on R holds if for *any* t_i,t_j in R:

<u>if</u> $t_i[A_1] = t_j[A_1]$ AND $t_i[A_2]=t_j[A_2]$ AND … AND $t_i[A_m] = t_j[A_m]$

<u>**then**</u> $t_i[B_1] = t_j[B_1]$ AND $t_i[B_2]=t_j[B_2]$ AND … AND $t_i[B_n] = t_j[B_n]$

FDs for Relational Schema Design

- High-level idea: why do we care about FDs?
  - Start with some relational schema
  - Find out its functional dependencies (FDs)
  - Use these to design a better schema
    - One which minimizes the possibility of anomalies

Functional Dependencies as Constraints

A **functional dependency** is a form of **constraint**

- *Holds* on some instances (but not others) – can check whether there are violations
- Part of the schema, helps define a *valid* instance

Student	**Course**	**Room**
Mary	CS3200	WVF20
Joe	CS3200	WVF20
Sam	CS3200	WVF20
..	..	..

Note: The FD {Course} → {Room} ***holds on this instance***

*Recall: an **instance** of a schema is a multiset of tuples conforming to that schema, **i.e. a table***

Functional Dependencies as Constraints

Note that:

- You can check if an FD is **violated** by examining a single instance;
- However, you **cannot prove** that an FD is part of the schema by examining a single instance.
  - *This would require checking every valid instance*

Student	Course	Room
Mary	CS3200	WVF20
Joe	CS3200	WVF20
Sam	CS3200	WVF20
..	..	..

However, cannot *prove* that the FD {Course} → {Room} is ***part of the schema***

More Examples

An FD is a constraint which holds, or does not hold on an instance:

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

More Examples

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

{Position} → {Phone}

More Examples

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

but *not* {Phone} → {Position}

Practice

A	B	C	D	E
1	2	4	3	6
3	2	5	1	8
1	4	4	5	7
~~1~~	~~2~~	~~4~~	~~3~~	~~6~~
~~3~~	~~2~~	~~5~~	~~1~~	~~8~~

Find at least *three* FDs which are violated on this instance:

{ } → { }
{ } → { }
{ } → { }

2. Finding FDs

What we will learn about next

- “Good” vs. “Bad” FDs: Intuition
- Finding FDs
- Closures
- PRACTICE: Compute the closures

1NF

- First normal form: A relation that has a primary key and in which there are no repeating groups
  - No multivalued attributes
  - Every attribute value is atomic (single fact in each table cell)

- All relations are in 1NF

- Normalization steps (from tabular view of data):
  - Goal: create a relation from the tabular view
  - Action: remove repeating groups
  - Action: select the primary key

Example: Convert To 1NF

Order_ID	Order_ Date	Customer_ ID	Customer_ Name	Customer_ Address	Product_ID	Product_ Description	Product_ Finish	Unit_ Price	Ordered_ Quantity
1006	10/24/2004	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
					5	Writer's Desk	Cherry	325.00	2
					4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2004	6	Furniture Gallery	Boulder, CO	11	4–Dr Dresser	Oak	500.00	4
					4	Entertainment Center	Natural Maple	650.00	3

- Normalization steps (from tabular view of data):
  - Goal: create a relation from the tabular view
  - Action: remove repeating groups
  - Action: select the primary key

Action: Remove Repeating Groups

Order_ID	Order_Date	Customer_ID	Customer_Name	Customer_Address	Product_ID	Product_Description	Product_Finish	Unit_Price	Ordered_Quantity
1006	10/24/2004	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
1006	10/24/2004	2	Value Furniture	Plano, TX	5	Writer's Desk	Cherry	325.00	2
1006	10/24/2004	2	Value Furniture	Plano, TX	4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2004	6	Furniture Gallery	Boulder, CO	11	4–Dr Dresser	Oak	500.00	4
1007	10/25/2004	6	Furniture Gallery	Boulder, CO	4	Entertainment Center	Natural Maple	650.00	3

- Is the data view a relation now?
  - Answer: yes
- Is it well-structured?
  - Answer: no

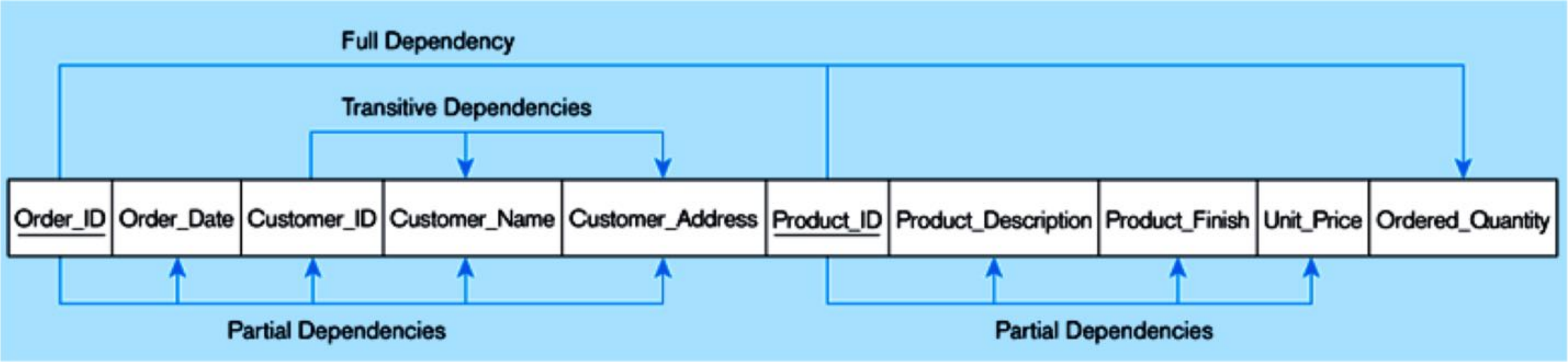
What are the anomalies in this table?

Order_ID	Order_ Date	Customer_ ID	Customer_ Name	Customer_ Address	Product_ID	Product_ Description	Product_ Finish	Unit_ Price	Ordered_ Quantity
1006	10/24/2004	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
1006	10/24/2004	2	Value Furniture	Plano, TX	5	Writer's Desk	Cherry	325.00	2
1006	10/24/2004	2	Value Furniture	Plano, TX	4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2004	6	Furniture Gallery	Boulder, CO	11	4–Dr Dresser	Oak	500.00	4
1007	10/25/2004	6	Furniture Gallery	Boulder, CO	4	Entertainment Center	Natural Maple	650.00	3

- Insertion: If new product is ordered for order 1007 of existing customer, customer data must be re-entered, causing duplication
- Deletion: If we delete the Dining Table from Order 1006, we lose information concerning this item's finish and price
- Update: Changing the price of product ID 4 requires update in several records
- Why do these anomalies exist? Because there are multiple themes (entity types) in one relation. -> duplication, and unnecessary dependency between entities

Action: Select A Primary Key

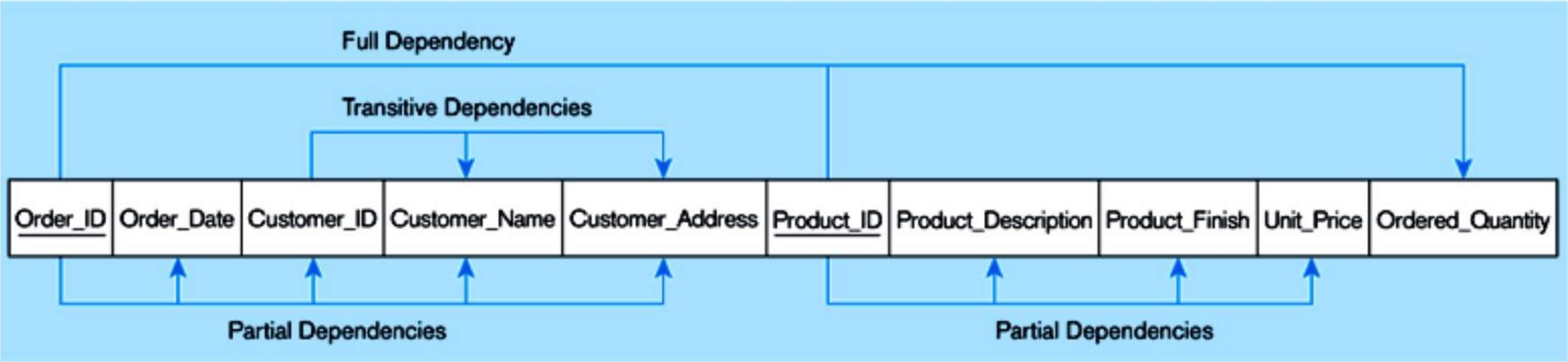
- Identify FDs and CKs (candidate keys = minimal superkeys)
- Four determinants and functional dependencies
  - Order_ID → Order_Date, Customer_ID, Customer_Name, Customer_Address
  - Customer_ID → Customer_Name, Customer_Address
  - Product_ID → Product_Description, Product_Finish, Unit_Price
  - Order_ID, Product_ID → Ordered_Quantity
- Select a PK from CKs
  - (Order_ID, Product_ID)



Next Step: Convert To 2NF

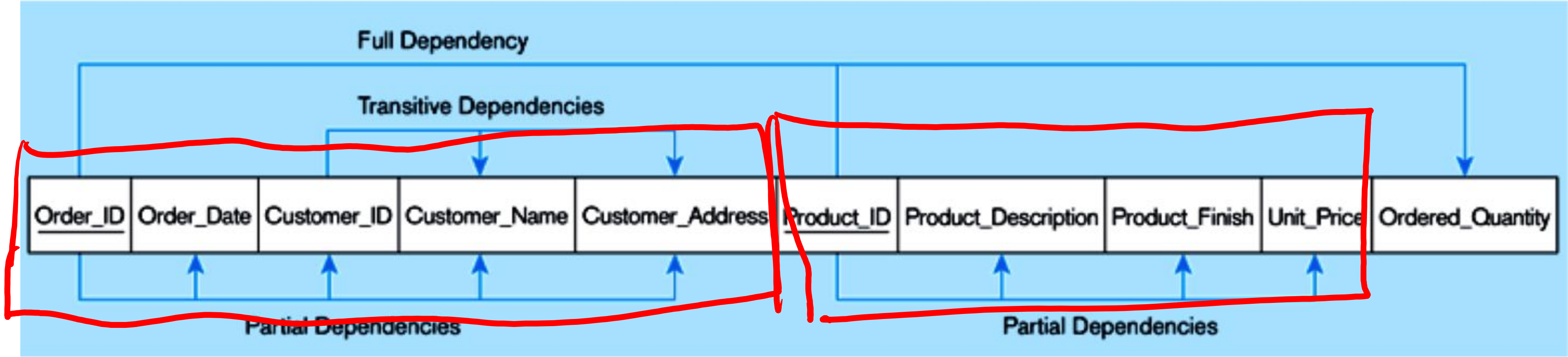
- 2NF: A relation in 2NF in which every non-key attribute is fully functionally dependent on the primary key

- <u>Partial FD</u>: A FD in which one or more nonkey attributes are functionally dependent on part (but not all) of the PK



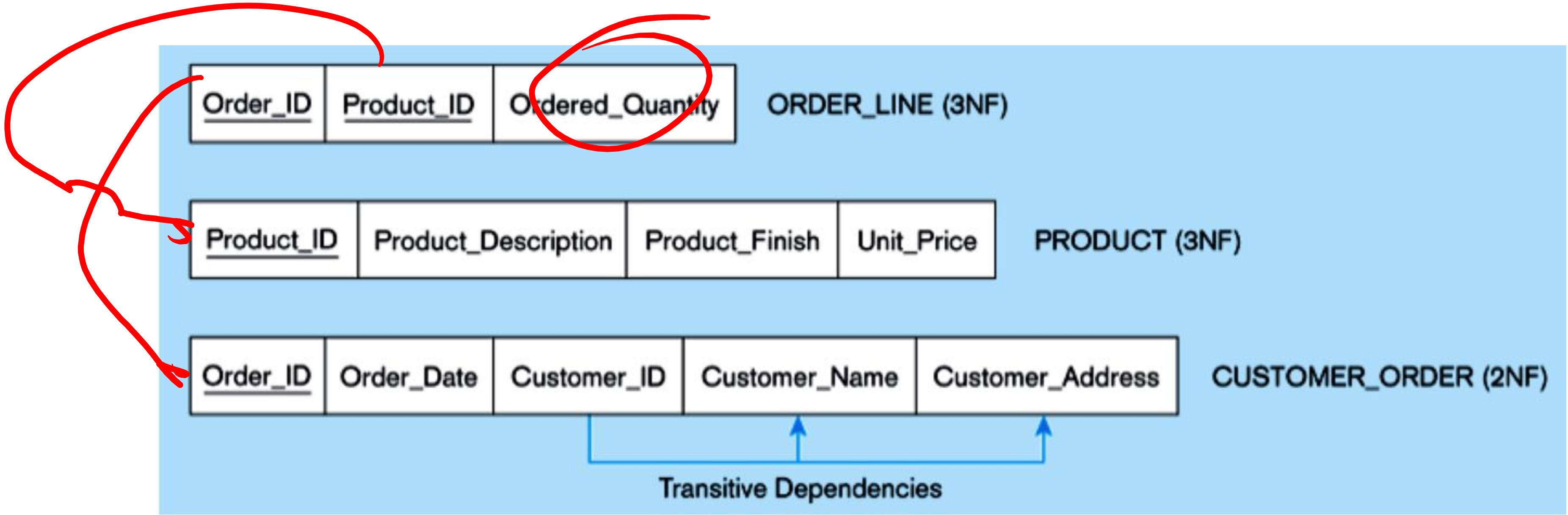
Getting A Relation To 2NF

- Create a new relation for each primary key attribute that is a determinant in a partial dependency
  - That attribute is the primary key in the new relation
- Move the nonkey attributes that are dependent on this primary key attribute(s) from the old relation to the new relation
- Exercise: Convert 1NF relation to 2NF



A 1NF Relation Is In 2NF if

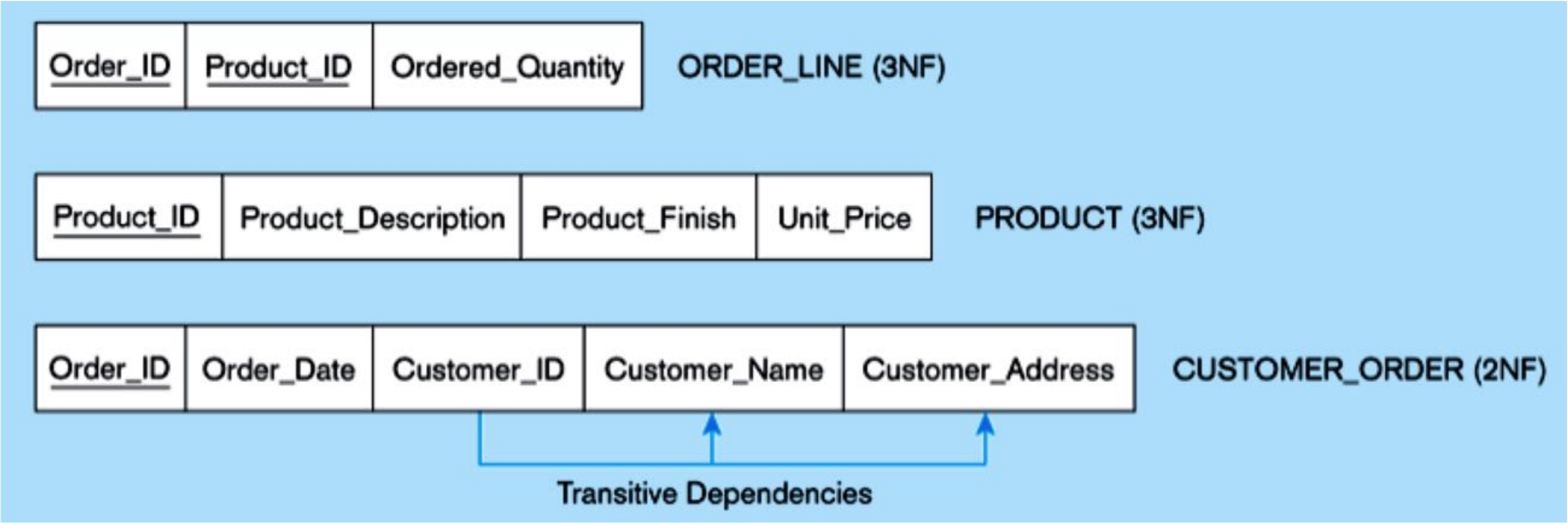
- The PK consists of only one attribute. There cannot be a partial dependency in such a relation
- (or) no nonkey attributes exist in the relation (thus all attributes in the relation are components of the PK). There are no FDs in such a relation
- (or) every nonkey attribute is functionally dependent on the full set of PK attributes.



3NF

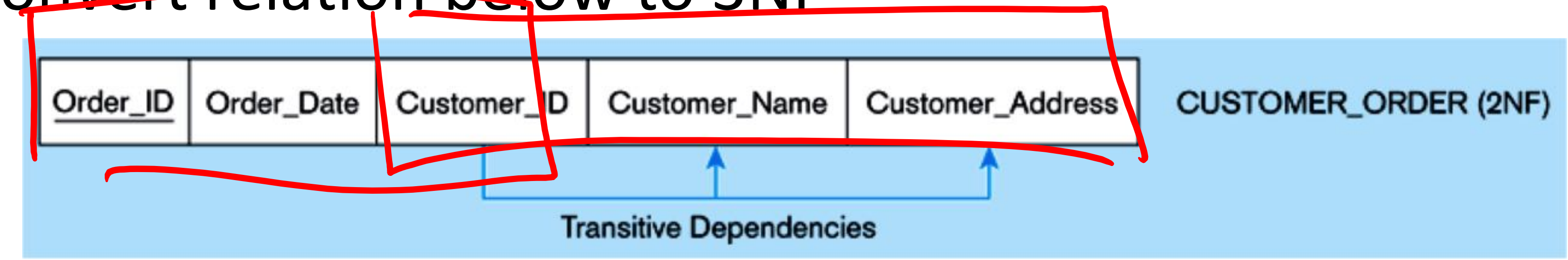
- 3NF: A relation that is in 2NF and has no transitive dependencies present
- <u>Transitive dependency</u>: An FD between two (or more) nonkey attributes
  - FD between the PK and one or more nonkey attributes that are dependent on the PK via another nonkey attribute
- Transitive dependency example:

<u>Transitivity:</u>
$a<b$ & $b<c \Rightarrow a<c$



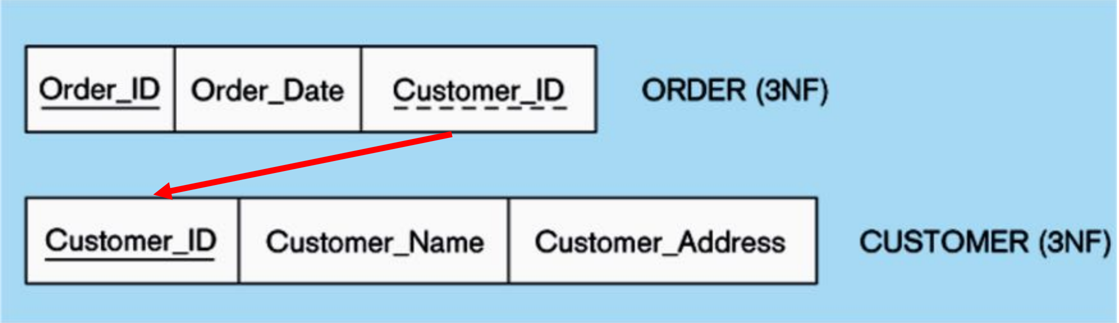
Removing Transitive Dependencies

- For each nonkey attribute(s) that is a determinant in a relation, create a new relation.
  - That attribute becomes the PK of the new relation
- Move all of the attributes that are functionally dependent on the attribute from the old to the new relation
- Leave the attribute (which serves as a PK in the new relation in the old relation to serve as a FK that allows us to associate the two relations
- Exercise: Convert relation below to 3NF

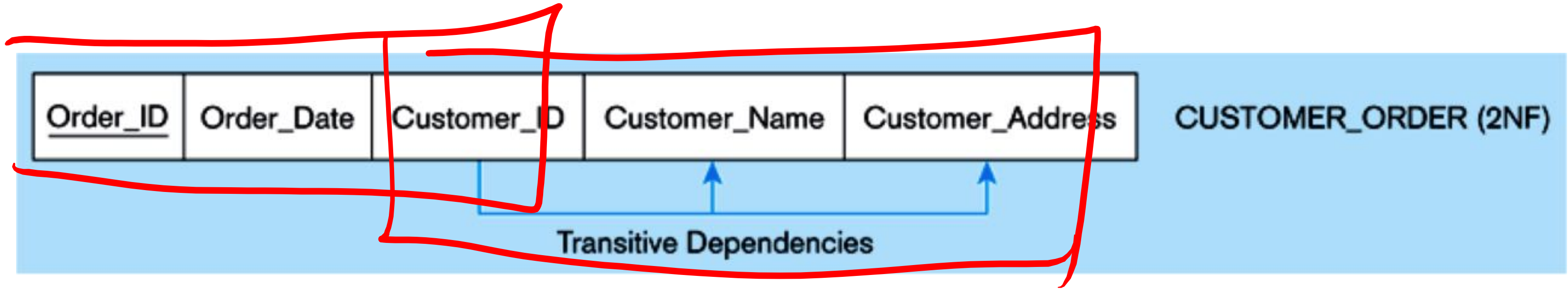


Third Normal Form

- Example converted to 3NF:



- Original example in 2NF:



Full Example: From 1NF to 3NF

Before (3NF):

Order_ID	Order_Date	Customer_ID	Customer_Name	Customer_Address	Product_ID	Product_Description	Product_Finish	Unit_Price	Ordered_Quantity

After (3NF):

Order_ID	Product_ID	Ordered_Quantity

ORDER_LINE (3NF)

Product_ID	Product_Description	Product_Finish	Unit_Price

PRODUCT (3NF)

Order_ID	Order_Date	Customer_ID

ORDER (3NF)

Customer_ID	Customer_Name	Customer_Address

CUSTOMER (3NF)

Normalization Summary

- Data normalization is the process of decomposing relations with anomalies to produce smaller, well-structured relations

- Goals of normalization include:
  - Minimize data redundancy
  - Simplifying the enforcement of referential integrity constraints
  - Simplify data maintenance (inserts, updates, deletes)
  - Improve representation model to match "the real world"