CHAPTER 4:
LOGICAL DATABASE DESIGN AND THE RELATIONAL MODEL (*PART I*)

Shakeel Ahmad

Modern Database Management

13th Edition

Jeff Hoffer, Venkataraman Ramesh, Heikki Topi

We learned in Chap 2 Database **Analysis;**

Chap 4: **Transformation** from ERD to relational model

We are moving into Database **Design**

OBJECTIVES

Disambiguation:

- "Data model" means E-R diagram;
- "Relational model" means data in 2-dimensional tables (current chapter)
  - 1) short text statement, and
  - 2) graphical representation

Part 1:

- Define terms
- List five properties of relations
- State two properties of candidate keys
- *Define first, second, and third normal form – mostly in PART II*
- Describe problems from merging relations
- Transform E-R *and EER diagrams* to relations
- Create tables with entity and relational integrity constraints

Part 2:

- Use normalization to decompose anomalous relations to well-structured relations

DATABASE SCHEMA (CH 1)

- External Schema
  - User Views
  - Can be determined from business-function/data entity matrices [Fig 1-6; pasted as next slide]
  - DBA determines schema for different users
- Conceptual Schema
  - View of data architect or data administrator
  - E-R models – covered in Chapters 2 and 3
- Internal Schema
  - Logical structures–representation for a type of data mgmt tech; covered in **Chapter 4 (Relational model)**
  - Physical structures–how data are to be represented and stored in 2ndry storage; covered in Chapter 5

FIGURE 1-6 Example business function-to-data entity matrix

Business Functions \ Data Entity Types	Customer	Product	Raw Material	Order	Work Center	Work Order	Invoice	Equipment	Employee
Business Planning	X	X						X	X
Product Development		X	X		X			X	
Materials Management		X	X	X	X	X		X	
Order Fulfillment	X	X	X	X	X	X	X	X	X
Order Shipment	X	X		X	X		X		X
Sales Summarization	X	X		X			X		X
Production Operations		X	X	X	X	X		X	X
Finance and Accounting	X	X	X	X	X		X	X	X

X = data entity is used within business function

Can use this matrix to help with your DB project:
"Did I get the right entities for my biz functions?"

INTRODUCTION OF BASIC CONCEPTS AND PROCEDURES

INTRODUCTION

- Logical DB design: process of transforming the conceptual data model (i.e., ERD) into a logical data model (i.e., relational models in this chap)
  - Conceptual data modeling: getting the right requirements >>model the biz logic correctly ← ← biz rules
  - Logical DB design: getting the requirements right >>correctly **transform** the biz logic **for implementation**
- An E-R data model is not a relational data model ➔ need normalization [2nd half of Ch 4]
  - ERD is developed for the purpose of **understanding** biz rules, not structuring data for sound DB **processing**
    - The last part is the goal of logical DB design

COMPONENTS OF RELATIONAL MODEL

- Data structure: Direct view (DB terms)
  - Tables (relation), rows (record), columns (field/attribute) (Chaps 2, 3, & 4)
- Data manipulation (Chaps 5 & 6)
  - Powerful SQL operations for retrieving and modifying data
- Data integrity (Chaps 4 & 5)
  - Mechanisms for implementing business rules that maintain integrity of manipulated data

RELATION (P. 155); CORRESPONDENCE W ERD

- A relation is a named, two-dimensional table of data. (with specific requirements)
  - A table consists of rows (records) and columns (attributes or fields).
- Correspondence w E-R model: 
- Relations (tables) correspond with **entity types** and with many-to-many relationship types (_____ entity)

1. Columns correspond with attributes.
2. Rows correspond with **entity instances** and with many-to-many relationship instances.

TEXT DESCRIPTION OF A RELATION

- We can express the structure of a relation by a shorthand notation (“text description”):

  RELATION (attribute1, attribute2, attribute3, …)

  Example: PRODUCT(…) ← exercise

- NOTE: The word ***relation*** (in relational database) is NOT to be confused with the word ***relationship*** (in E-R model): …

Relation != Relationship

KEY FIELDS

Primary key = Identifier

- Keys are special fields that serve two main purposes:
  - ***Primary key*** is an attribute or combination of attributes that uniquely identifies a row in a relation.
    - Example: employee numbers, social security numbers, etc.
    - -- *guarantees that all rows are unique.*
  - ***Foreign keys*** are identifiers that enable a dependent relation (**on the many side** of a relationship) to refer to its parent relation (on the one side of the relationship).
- Keys can be ***simple*** (a single field) or ***composite*** (more than one field).
- Keys usually are used as indexes to speed up the response to user queries

A key can have >1 attribute

FOREIGN KEY

- Foreign key:
- represent the relationship between two tables or relations
- An attribute in a relation that serves as the primary key of another ("foreign") relation

P.156:

EMOPLYEE1(EmpID, Name, DeptName, Salary)

DEPARTMENT(DeptName, Location, Fax)

DeptName is a foreign key in EMPLOYEE1. It allows a user to associate any employee w the dept to which s/he is assigned.

PROPERTIES OF RELATION (P. 156)

- Requirements for a table to qualify as **a relation**:
  1. It must have a unique name.
  2. Every attribute value must be atomic
     1. (not multivalued, not composite).
  3. Every row must be unique (can't have two rows with exactly the same values for all their fields).
  4. Attributes (columns) in tables must have unique names.
  5. The order of the columns must be irrelevant.
  6. The order of the rows must be irrelevant.

"A relation is a table; not all tables are relations"

NOTE: all *relations* are in ***1st Normal form*** (Definition P.181)

Relation < == > First Normal Form

REMOVING MULTIVALUED ATTRIBUTES FIG 4-2

Emp_ID	Name	Dept_Name	Salary	Course_Title	Date_Completed
100	Margaret Simpson	Marketing	48,000	SPSS	6/19/200X
				Surveys	10/7/200X
140	Alan Beeton	Accounting	52,000	Tax Acc	12/8/200X
110	Chris Lucero	Info Systems	43,000	Visual Basic	1/12/200X
				C++	4/22/200X
190	Lorenzo Davis	Finance	55,000		
150	Susan Martin	Marketing	42,000	SPSS	6/16/200X
				Java	8/12/200X

- Identify the multivalued attributes for several records (entity instances)

SCHEMAS (PP. 157-158)

- The structure of a DB is described thru schemas
  - A DB can have any number of relations
- **Two common methods** for expressing schema:

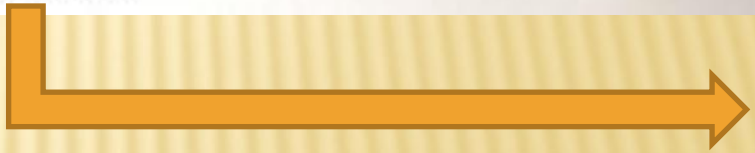
1. Short text statements (example: Slide 15)
   - RELATION (attribute1, attribute2, attribute3, …)
   - EMPLOYEE (Emp_ID, LastName, FirstName, …)
2. Graphical representation (example: Slide 16)
   - Each relation represented by a rectangle containing attributes
   - RELATION

Better means of expressing referential integrity constraint

Primary key Foreign key

SHORT TEXT STATEMENT OF PVFC

CUSTOMER(Customer_ID, Customer_Name,Customer_Address,
State,Postal_Code)
ORDER(Order_ID Order_Date,Customer_ID)
ORDER LINE((Order_ID, Product_ID Ordered_Quantity)
PRODUCT(Product_ID, Product_Description,Product_Finish,
Standard_Price, Product_Line_ID)



Graphical representation:

Lines/curves with arrowhead are used to indicate the referencing relationship (**referential integrity**) between foreign key and primary key.

Figure 4-3 Schema for four relations (Pine Valley Furniture Company)

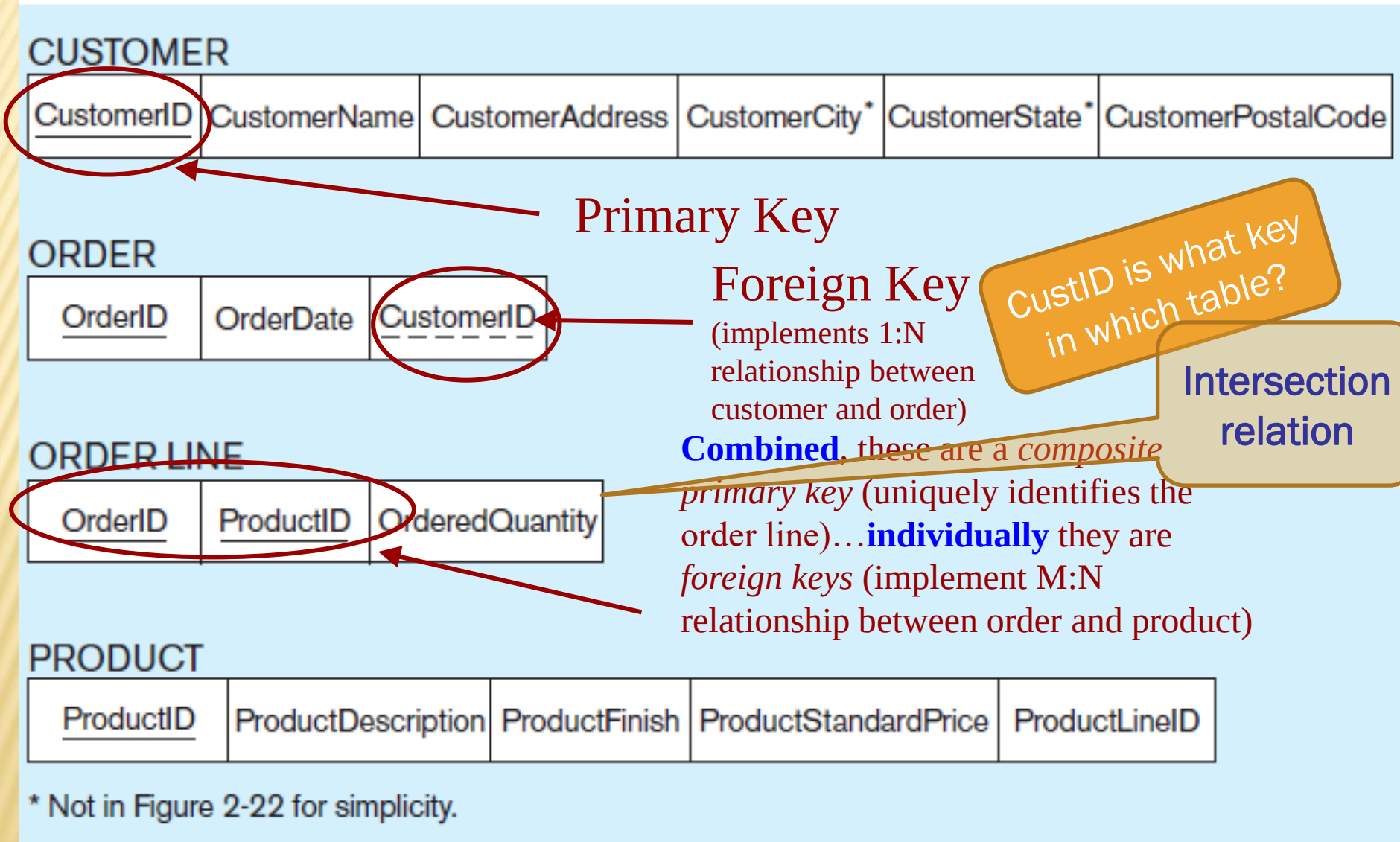
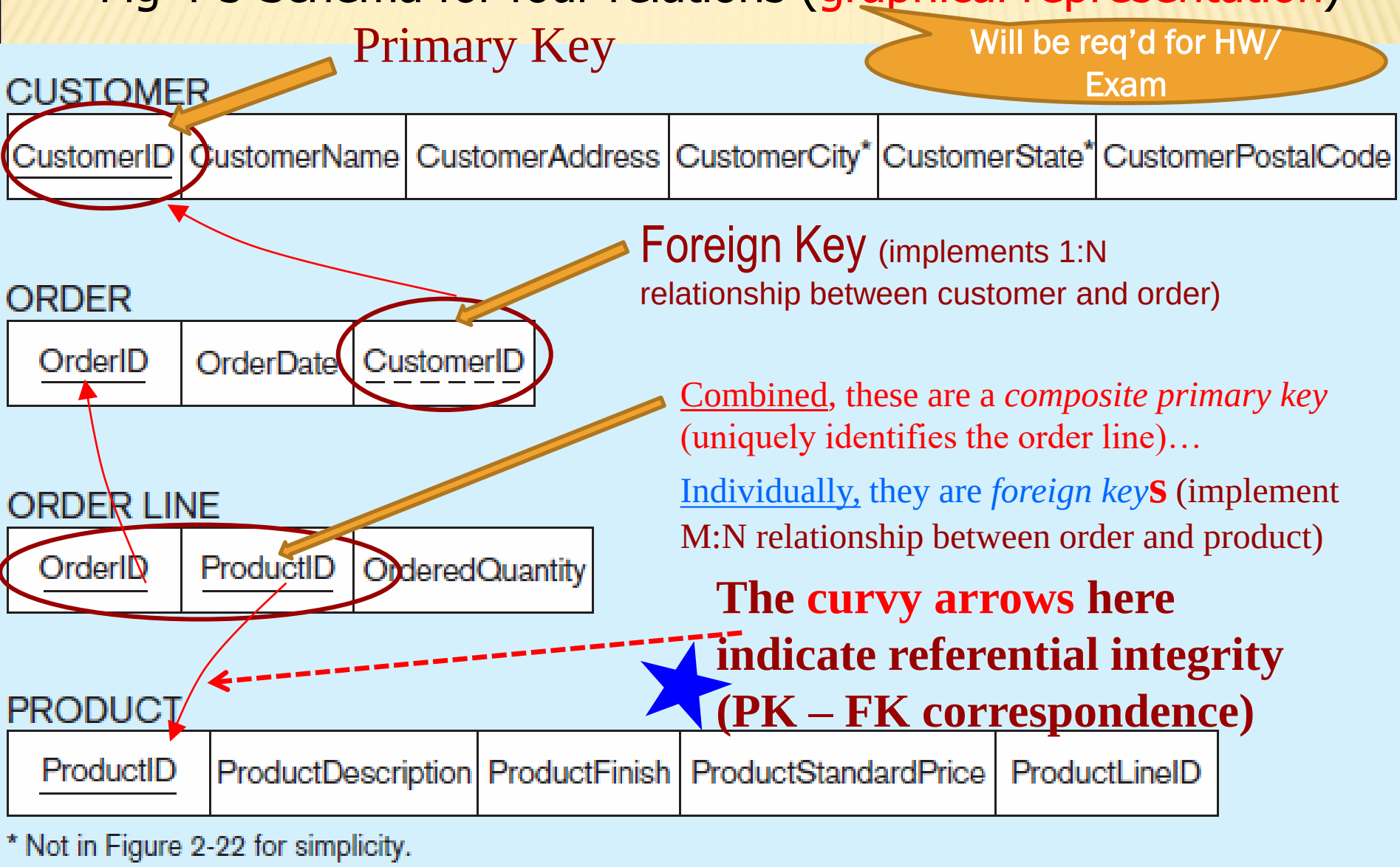


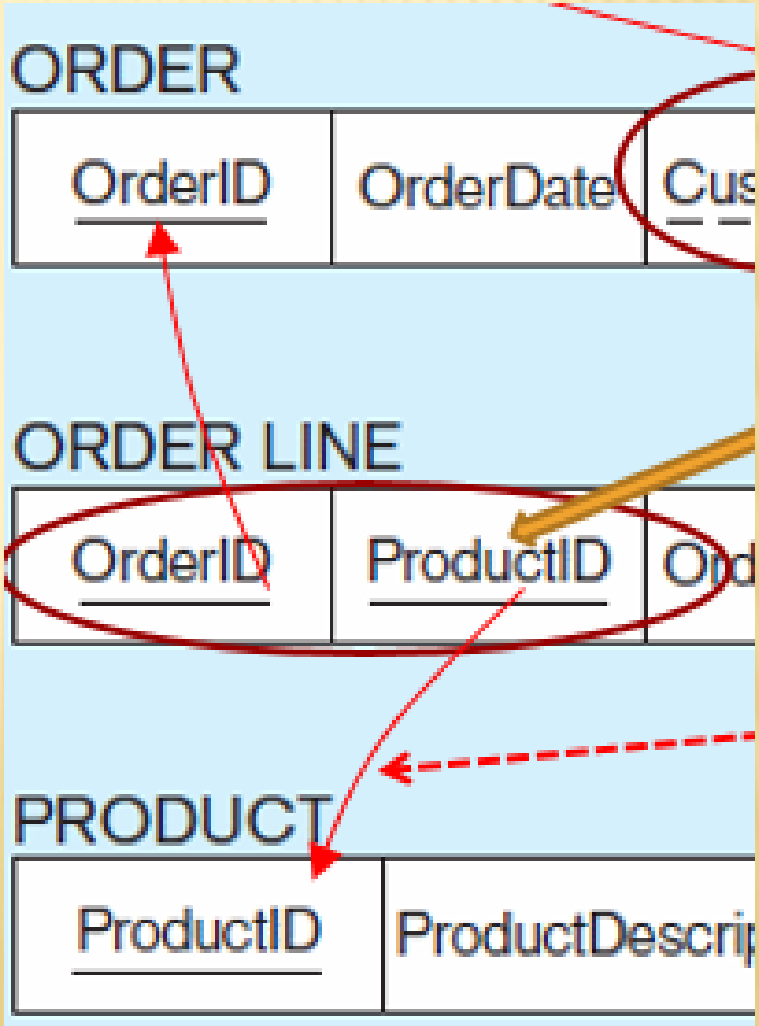
Fig 4-3 Schema for four relations (graphical representation)



* Not in Figure 2-22 for simplicity.

The composite key of associative entity/intersection relation

1. Combined, these are a *composite primary key* (uniquely identifies the order line)…
2. Individually, they are *foreign key***s** (implement M:N relationship between order and product)



Read end of P. 157-158

INTEGRITY CONSTRAINTS

INTEGRITY CONSTRAINTS

- Domain Constraints
  - Allowable values for an attribute (See Table 4-1)
- Entity Integrity
  - **No primary key attribute may be null.** All primary key fields **MUST contain data values.**
- Referential Integrity
  - Rules that maintain **consistency between the rows of two related tables.**

TABLE 4-1 Domain Definitions for INVOICE Attributes

Attribute	Domain Name	Description	Domain
CustomerID	Customer IDs	Set of all possible customer IDs	character: size 5
CustomerName	Customer Names	Set of all possible customer names	character: size 25
CustomerAddress	Customer Addresses	Set of all possible customer addresses	character: size 30
CustomerCity	Cities	Set of all possible cities	character: size 20
CustomerState	States	Set of all possible states	character: size 2
CustomerPostalCode	Postal Codes	Set of all possible postal zip codes	character: size 10
OrderID	Order IDs	Set of all possible order IDs	character: size 5
OrderDate	Order Dates	Set of all possible order dates	date: format mm/dd/yy
ProductID	Product IDs	Set of all possible product IDs	character: size 5
ProductDescription	Product Descriptions	Set of all possible product descriptions	character: size 25
ProductFinish	Product Finishes	Set of all possible product finishes	character: size 15
ProductStandardPrice	Unit Prices	Set of all possible unit prices	monetary: 6 digits
ProductLineID	Product Line IDs	Set of all possible product line IDs	integer: 3 digits
OrderedQuantity	Quantities	Set of all possible ordered quantities	integer: 3 digits

Domain definitions enforce domain integrity constraints.

INTEGRITY CONSTRAINTS – REFERENTIAL INTEGRITY (P.160)

* Online discussion!

- Rule states that any foreign key value (in the relation of the many side) MUST match a primary key value (in the relation of the one side). (Or the foreign key can be null)
  - For example: Delete Rules
    - Restrict–don't allow delete of "parent" side if related rows exist in "child" / "dependent" side
    - Cascade–automatically delete "dependent" side rows that correspond with the "parent" side row to be deleted
    - Set-to-Null–set the foreign key in the dependent side to null if deleting from the parent side → not allowed for weak entities

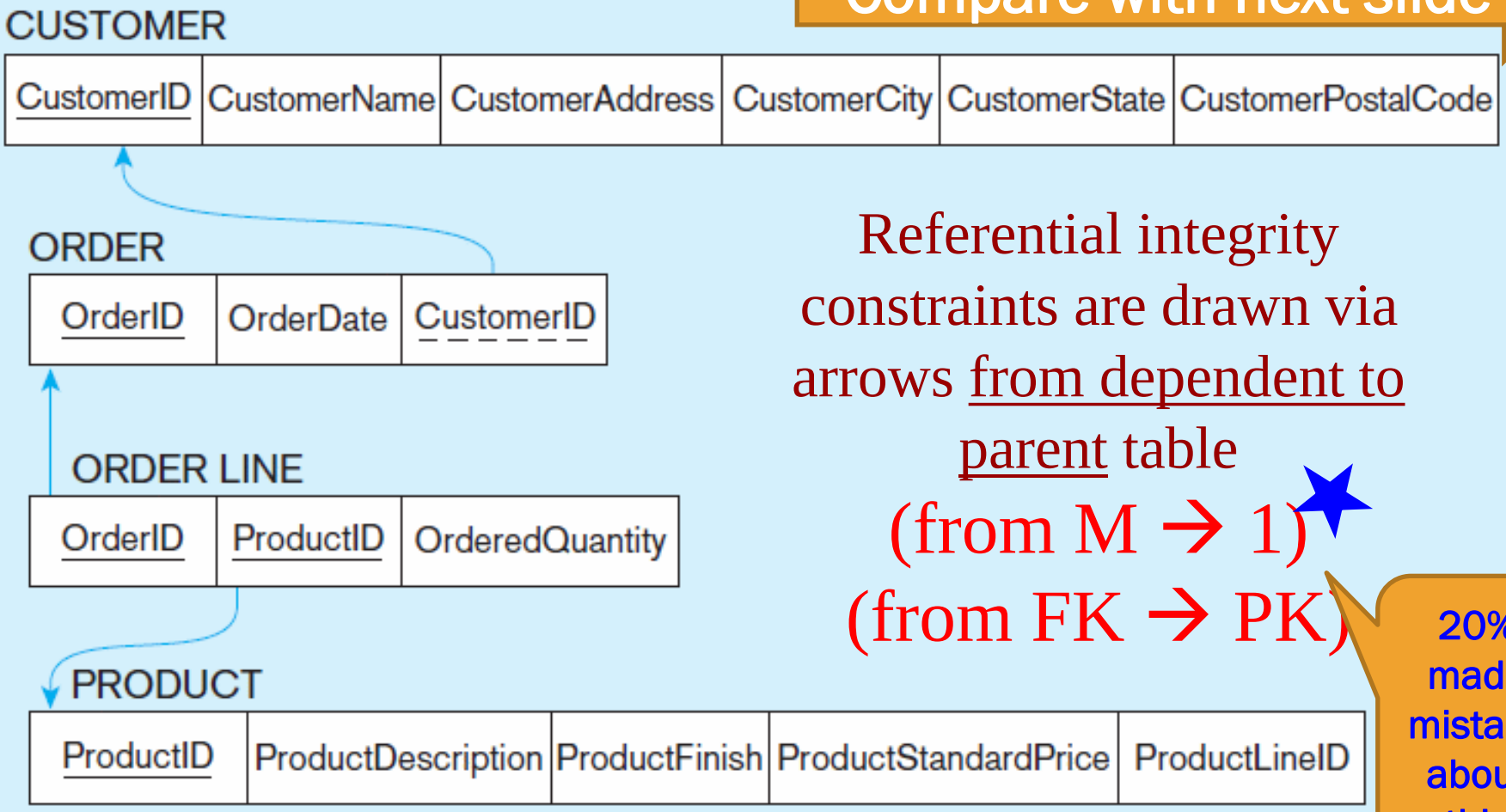
Example using IS 312 DB

EVERY relation must have a ref integrity arrow connected with at least one other relation in the schema

Fig 4-5 Referential integrity constraints

Read end of 160-161

Compare with next slide



Referential integrity constraints are drawn via arrows from dependent to parent table

(from M → 1)

(from FK → PK)

20% made mistake about this

What if there's a relation that does NOT have ref integrity arrow connected with at least one other relation?

Figure 1-3(b), P.12

Compare w previous slide

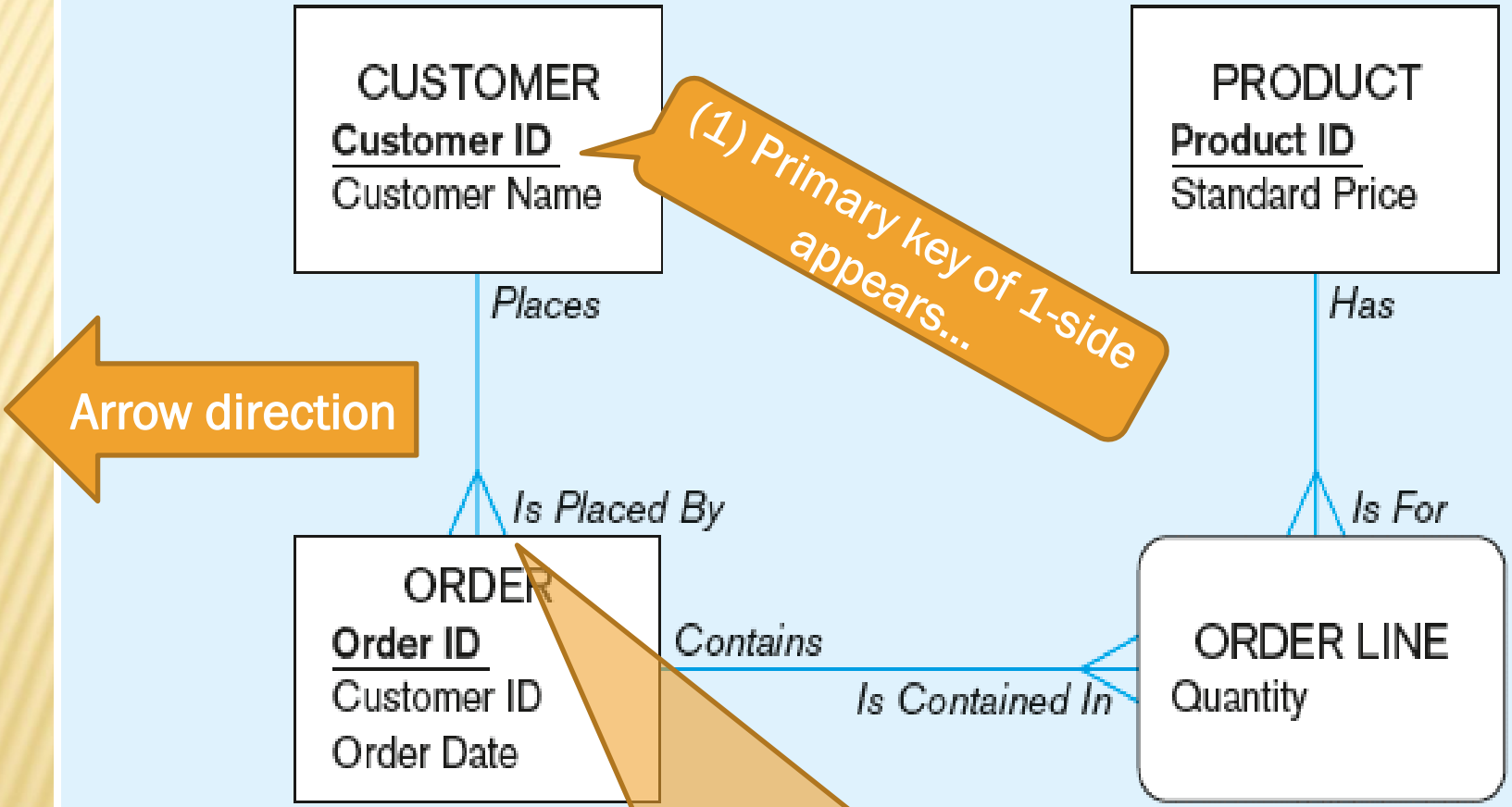


Figure 4-6 SQL table definitions

```
CREATE TABLE Customer_T
        (CustomerID                     NUMBER(11,0)      NOT NULL,
        CustomerName                    VARCHAR2(25)      NOT NULL,
        CustomerAddress                 VARCHAR2(30),
        CustomerCity                    VARCHAR2(20),
        CustomerState                   CHAR(2),
        CustomerPostalCode              VARCHAR2(9),
CONSTRAINT Customer_PK PRIMARY KEY (CustomerID));

CREATE TABLE Order_T
        (OrderID                        NUMBER(11,0)      NOT NULL,
        OrderDate                       DATE DEFAULT SYSDATE,
        CustomerID                      NUMBER(11,0),
CONSTRAINT Order_PK PRIMARY KEY (OrderID),
CONSTRAINT Order_FK FOREIGN KEY (CustomerID) REFERENCES Customer_T (CustomerID));

CREATE TABLE Product_T
        (ProductID                      NUMBER(11,0)      NOT NULL,
        ProductDescription              VARCHAR2(50),
        ProductFinish                   VARCHAR2(20),
        ProductStandardPrice            DECIMAL(6,2),
        ProductLineID                   NUMBER(11,0),
CONSTRAINT Product_PK PRIMARY KEY (ProductID));

CREATE TABLE OrderLine_T
        (OrderID                        NUMBER(11,0)      NOT NULL,
        ProductID                       NUMBER(11,0)      NOT NULL,
        OrderedQuantity                 NUMBER(11,0),
CONSTRAINT OrderLine_PK PRIMARY KEY (OrderID, ProductID),
CONSTRAINT OrderLine_FK1 FOREIGN KEY (OrderID) REFERENCES Order_T (OrderID),
CONSTRAINT OrderLine_FK2 FOREIGN KEY (ProductID) REFERENCES Product_T (ProductID));
```

Referential integrity constraints are implemented with foreign key to primary key references.

REFERENTIAL INTEGRITY (RELATIONSHIP)

- PK-FK pair (referential integrity) is the way a relationship is implemented in relational database
- PK-FK pair exists when there is 1-M relationship
  - The PK in the table on the 1-side, must appear as -
  - FK in the table on the M-side
- The ref. integ. arrow goes from the M-side to the 1-side, FK→PK: "referencing the PK on the 1-side"
- Do not flip the referential integrity arrow
- Q: In the intersection relation/associative entity, which directions do the ref. integ. arrows go?

SUMMARY OF PREVIOUS SECTION

In ERD	In Rela. Schema	Note
Entity	Table	No space in table name
Attribute	Column	attribute domains
Primary key	Primary key	can be composite
1:M relationship	Prim. key on 1-side Foreign on M-side	***Don't flip the two ! ! !***
M:N relationship can → Associative entity	MUST: New relation/table for the M:N relationship itself – Intersection relat'n	M:N → → associative entity → → table on its own

Relationship **(curvy)** arrow always points -
from ___-side to __-side;
from _____ key to _____ key

Hi error-rate concept; ref previous 4 slides

ANOMALIES

Three anomalies in this relation/table:

1. Insertion anomaly
2. Deletion anomaly
3. Modification anomaly (PP. 162~163)

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	Visual Basic	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

TRANSFORMING EER DIAGRAMS TO RELATIONS

TRANSFORMING EER DIAGRAMS INTO RELATIONS (P. 163)

Three types of entities:

1. *Regular entities* are entities that have an independent existence and generally represent real-world objects, such as persons and products. Regular entity types are represented by rectangles with a single line.
2. *Weak entities* are entities that cannot exist except with an identifying relationship with an owner (regular) entity type. Weak entities are identified by a rectangle with a double line.
3. *Associative entities* (also called gerunds) are formed from many-to-many relationships between other entity types. Associative entities are represented by a rectangle with rounded corners.

TRANSFORMING EER DIAGRAMS INTO RELATIONS – REGULAR ENTITIES

Mapping Regular Entities to Relations – handling ***attributes***:

1. Simple attributes: E-R attributes map directly onto the relation **(Fig 4-8, next slide)**

2. Composite attributes: Use only their simple, component attributes **(Figure 4-9)**

3. Multivalued attribute: Becomes a separate relation with a foreign key taken from the superior entity **(Figure 4-10)**

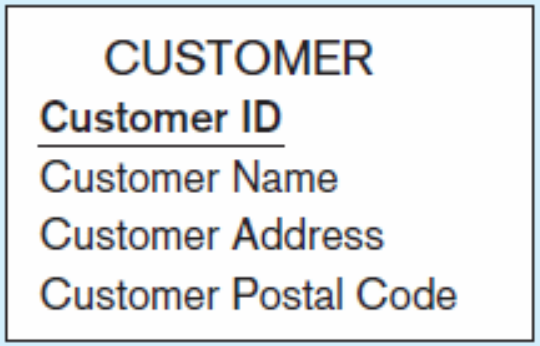
Use

"Tear apart"

Create new relation

Figure 4-8 Mapping a regular entity

(a) CUSTOMER entity type with simple attributes



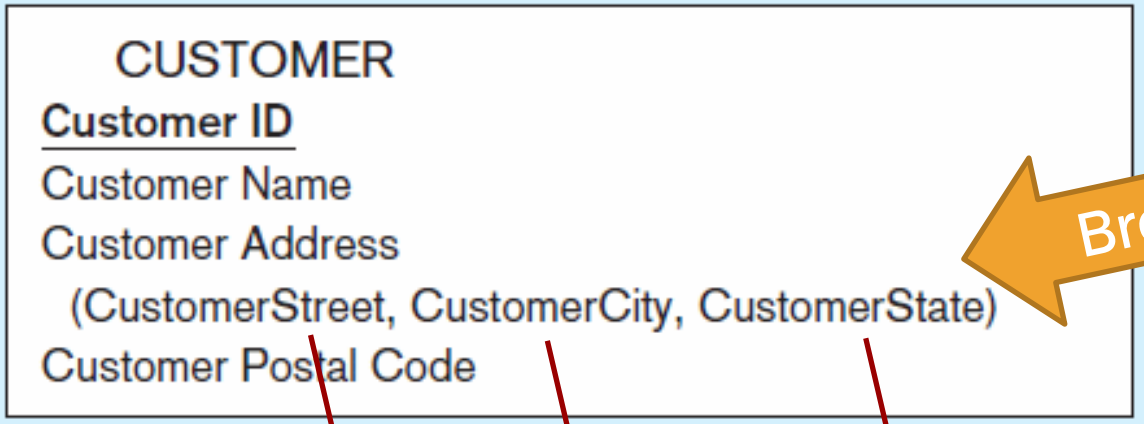
(b) CUSTOMER relation

CUSTOMER

CustomerID	CustomerName	CustomerAddress	CustomerPostalCode

Figure 4-9 Mapping a composite attribute

(a) CUSTOMER entity type with **composite** attribute



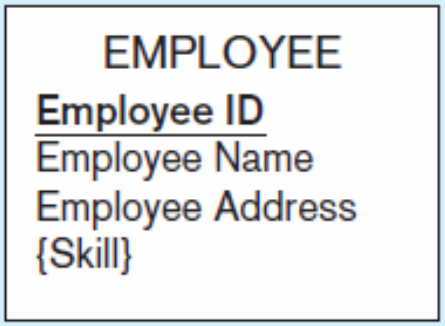
(b) CUSTOMER relation with address detail

CUSTOMER

CustomerID	CustomerName	CustomerStreet	CustomerCity	CustomerState	CustomerPostalCode

Figure 4-10 Mapping an entity with a multivalued attribute

(a)

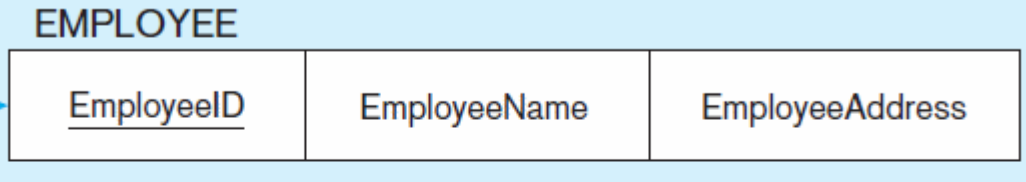


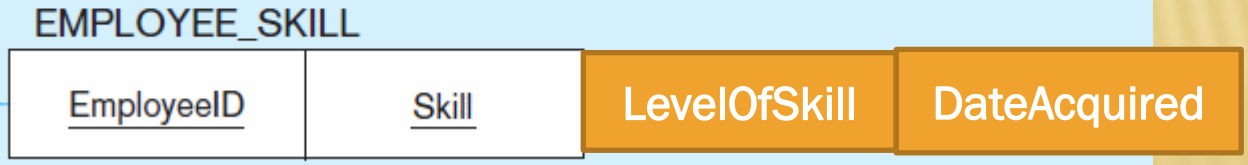
Q: In below, who's 1 and who's M?

Multivalued attribute becomes a separate relation with foreign key

Ref back three slides

(b)





Imagine table?

One–to–many relationship between original entity and new relation

MULTI-VALUED ATTRIBUTES

- The first relation EMPLOYEE has the primary key Employee_ID
- The second relation EMPLOYEE_SKILL has attributes Employee_ID and Skill, which form the primary key
  - **"When in doubt, construct the tables"** –
  - What would these two tables look like?
  - What other field(s) could be in the second table (EMPLOYEE_SKILL)?

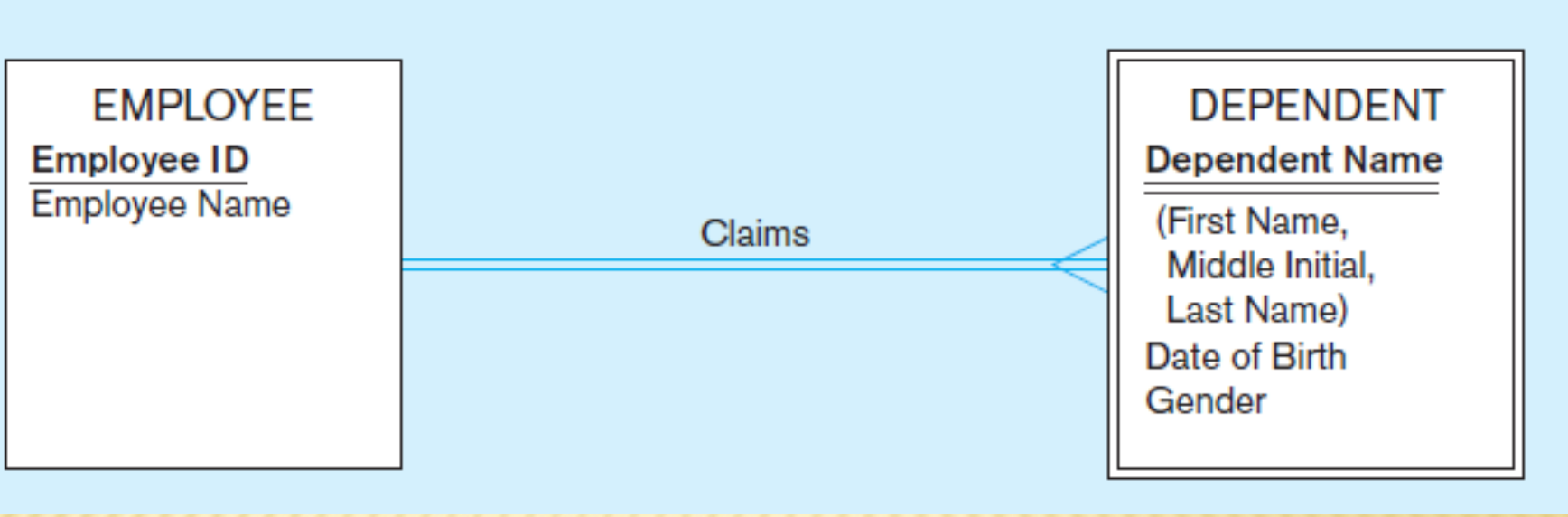
TRANSFORMING EER DIAGRAMS INTO RELATIONS (CONT.) – WEAK ENTITIES

Mapping Weak Entities

- Becomes a separate relation with a foreign key taken from the superior entity
- → → Figure 4-11(b)
- Primary key composed of:
  - Partial identifier of weak entity
  - Primary key of identifying relation (strong entity)
  - What would the table for the weak entity look like?

Figure 4-11 Example of mapping a weak entity

a) Weak entity DEPENDENT



In a relational database, a **weak entity** is an **entity** that cannot be uniquely identified by its attributes alone; therefore, it must use a foreign key in conjunction with its attributes to create a primary key.

-- https://en.wikipedia.org/wiki/Weak_entity

Figure 4-11 Example of mapping a weak entity (cont.)

b) Relations resulting from weak entity

TRANSFORMING EER DIAGRAMS INTO RELATIONS (CONT.) – BINARY RELATIONSHIPS

Mapping Binary Relationships

1. One-to-Many–Primary key on the **one**-side becomes a foreign key on the **many**-side
2. Many-to-Many–Create a ***new relation*** with the primary keys of the two entities as its primary key
3. One-to-One–Primary key on the mandatory side becomes a foreign key on the optional side

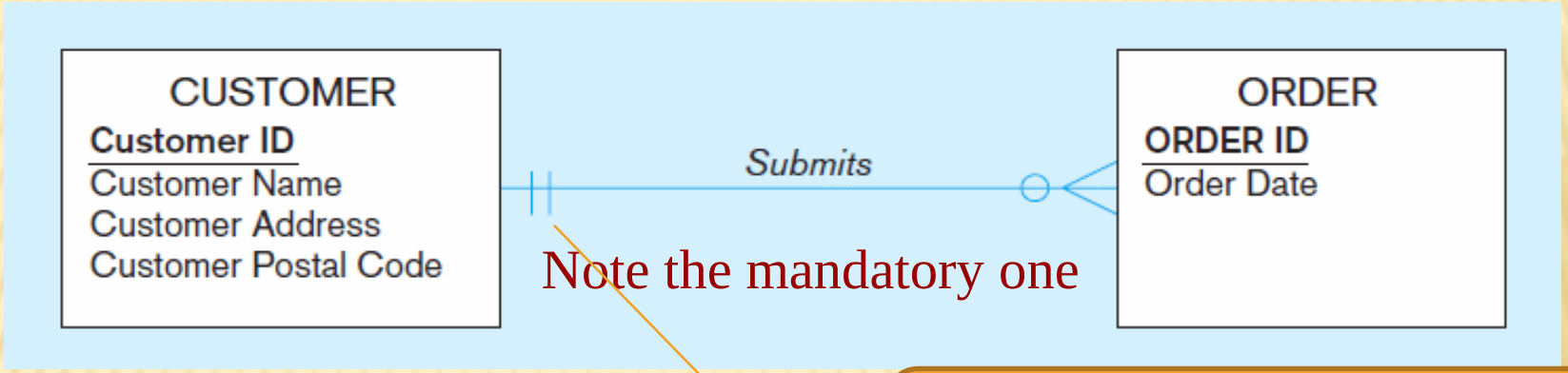
Called intersection relation

Logically from entity

+ (curvy) Arrows indicating referential integrity constraints
  - × Points from **M**-side to **1**-side
  - × From foreign key to primary key

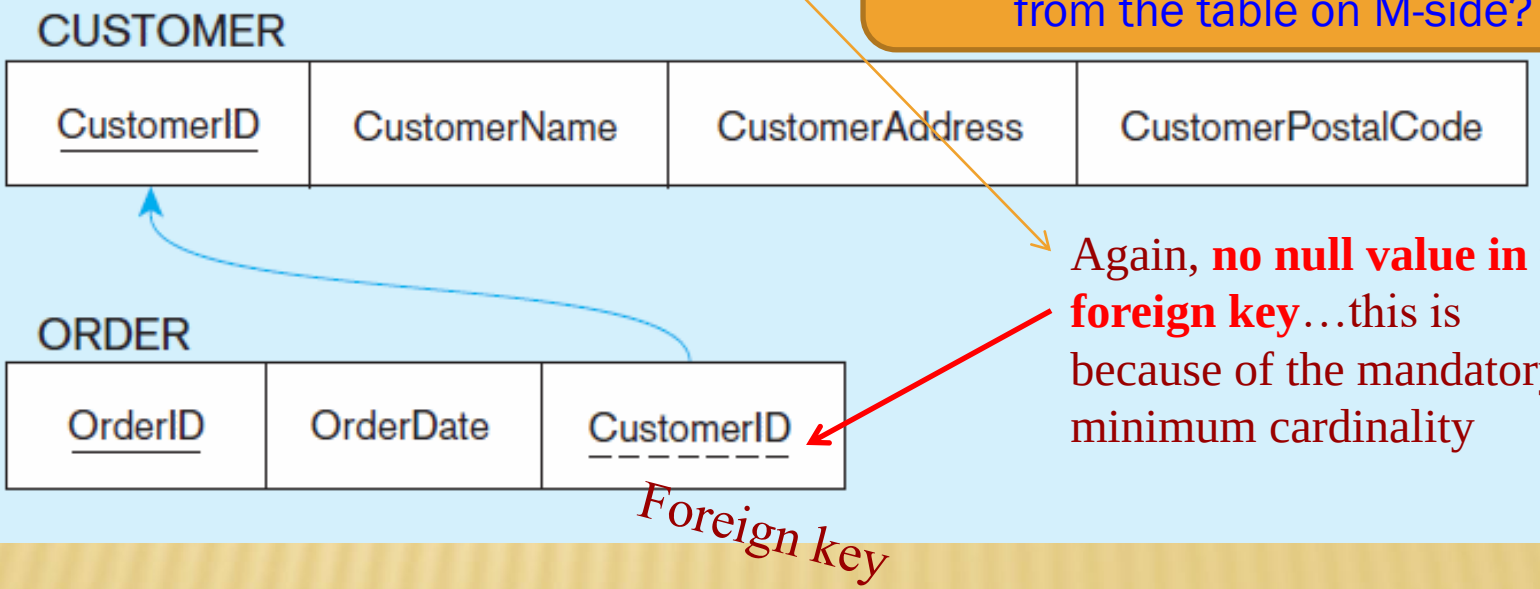
Figure 4-12 Example of mapping a 1:M relationship

a) Relationship between customers and orders

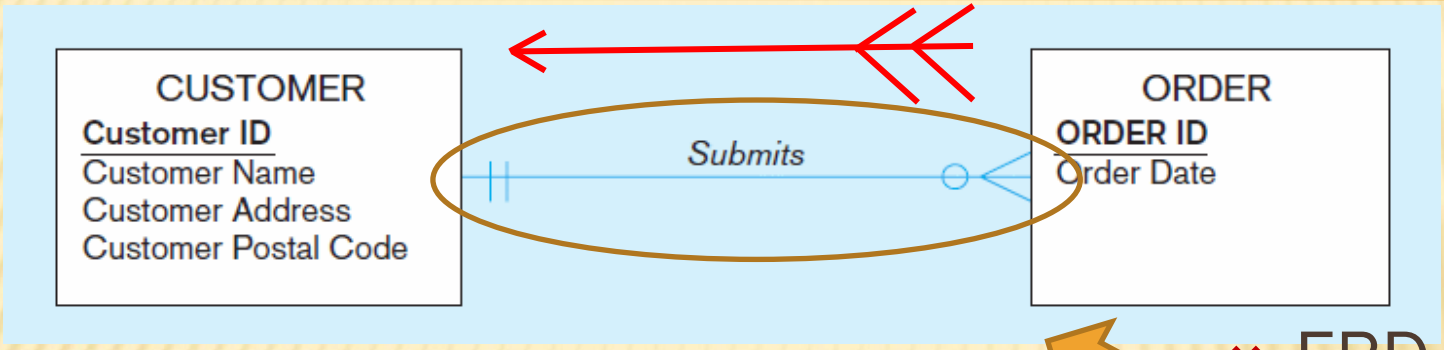


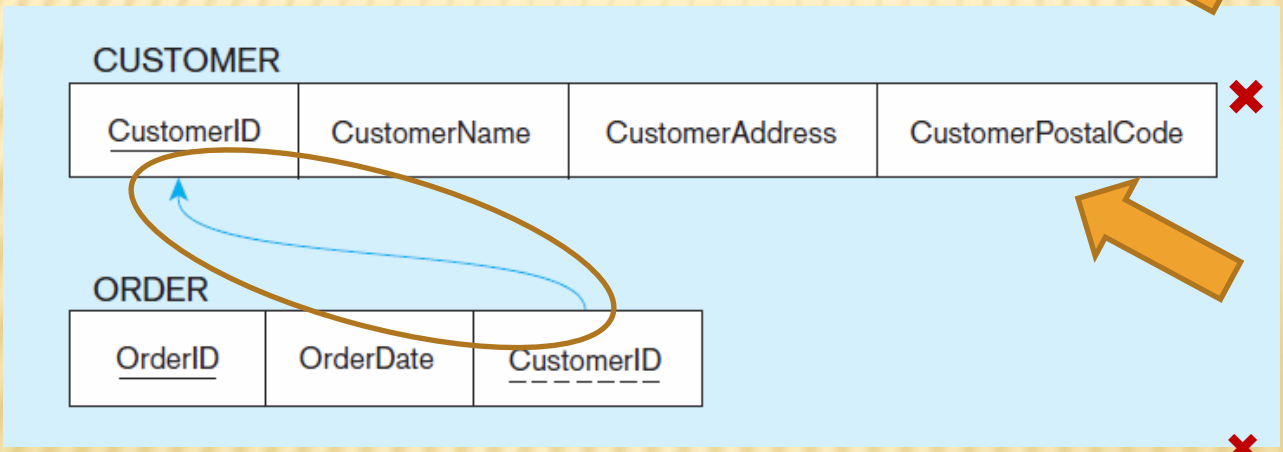
b) Mapping the relationship

If there's no field name/table name, can you still tell the table on 1-side from the table on M-side?



NEW FA'17: "SAMENESS" OF DIRECTION OF "ARROWS"





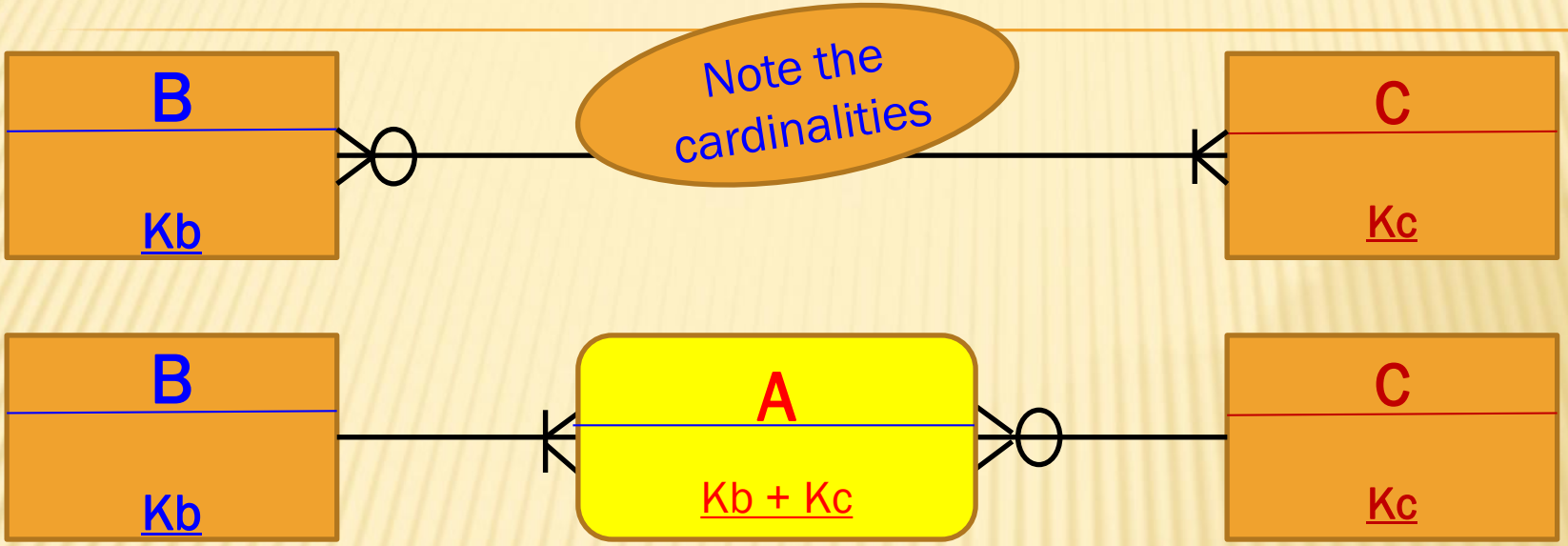
- ERD (one box, one entity/table)
- Relational model (one box, one attribute/column)
- Do not confuse the two

MAP BINARY MANY-TO-MANY (M:N) RELATIONSHIPS

- If M:N relationship exists between entity types B and C, we create a new relation A,
  - Entity → relation >>>
    Associative entity → intersection relation
  - include as foreign keys in A the primary keys for B and C,
  - these attributes, together, become the primary key of A
  - These attributes, separately, function as

EXAMPLE: next 3 slides

MAP BINARY MANY-TO-MANY (M:N) RELATIONSHIPS



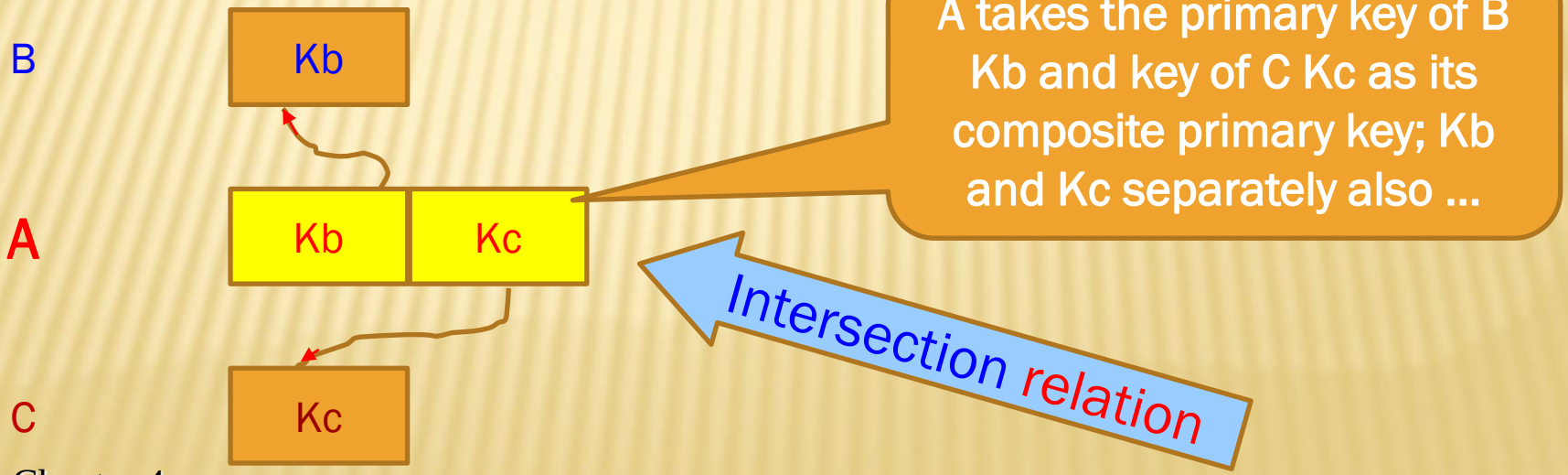
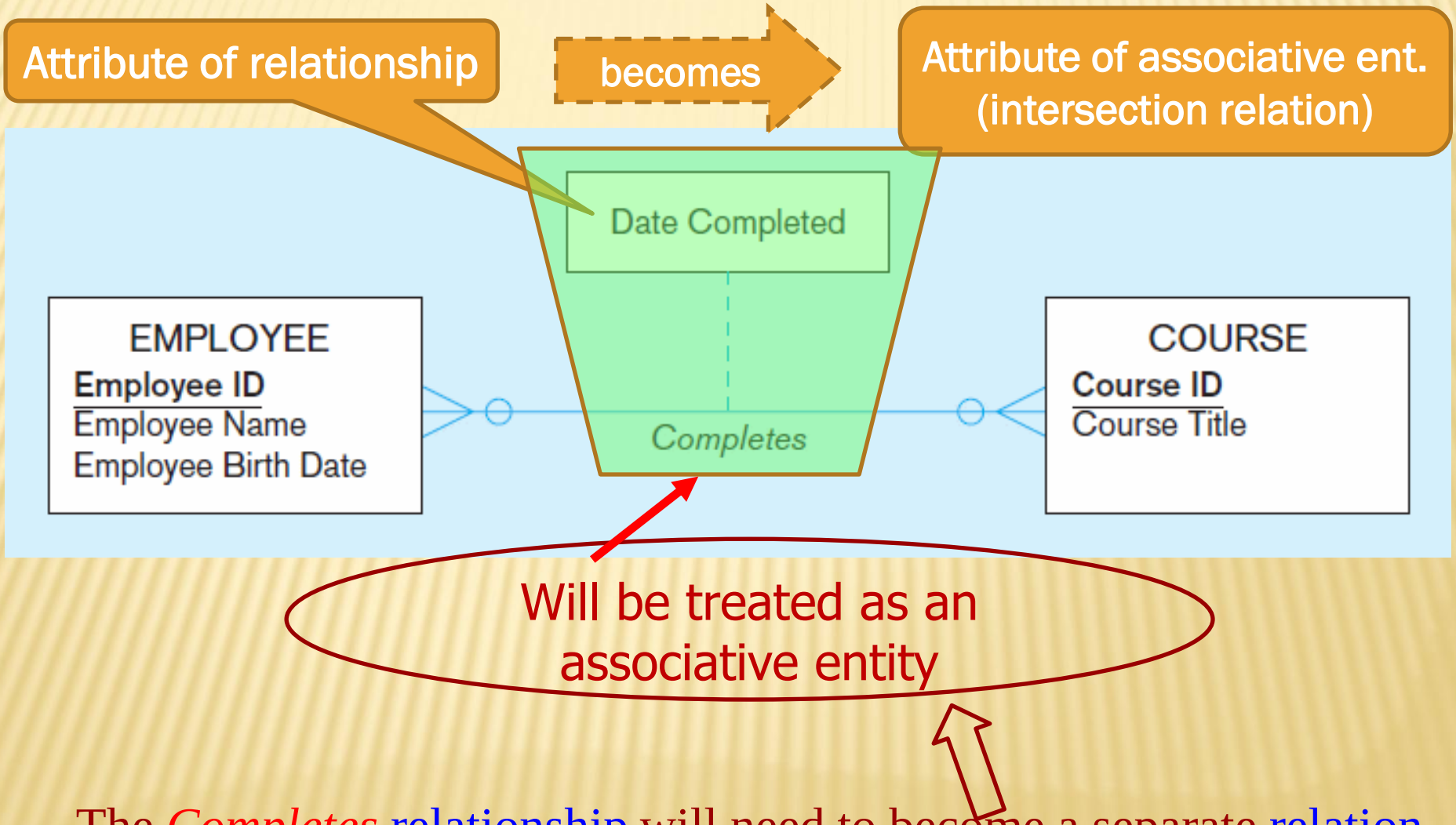


Figure 4-13 Example of mapping an M:N relationship

a) ***Completes*** relationship (M:N)



The *Completes* relationship will need to become a separate relation

Figure 4-13 Example of mapping an M:N relationship (cont.)

b) Three resulting relations

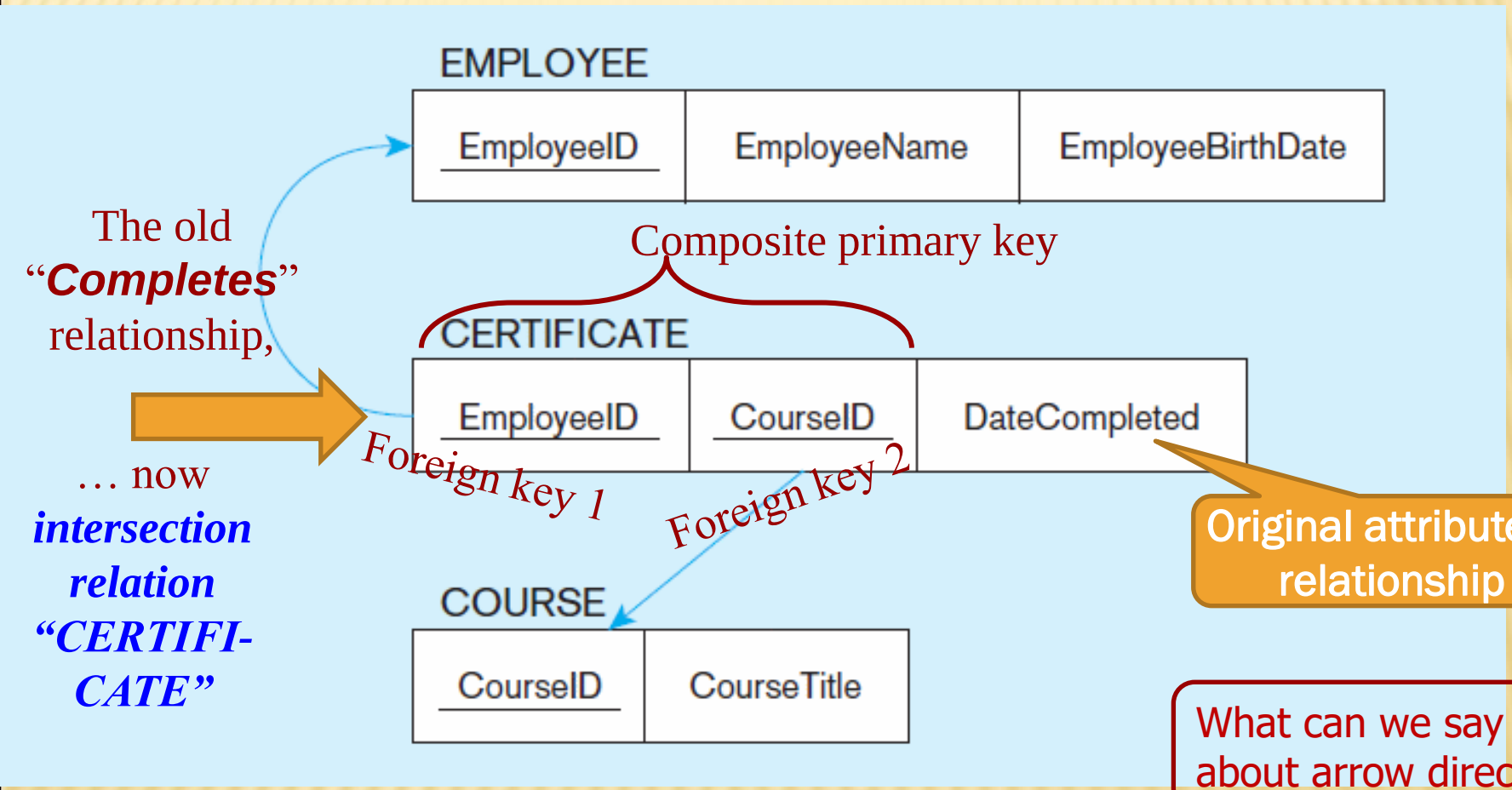
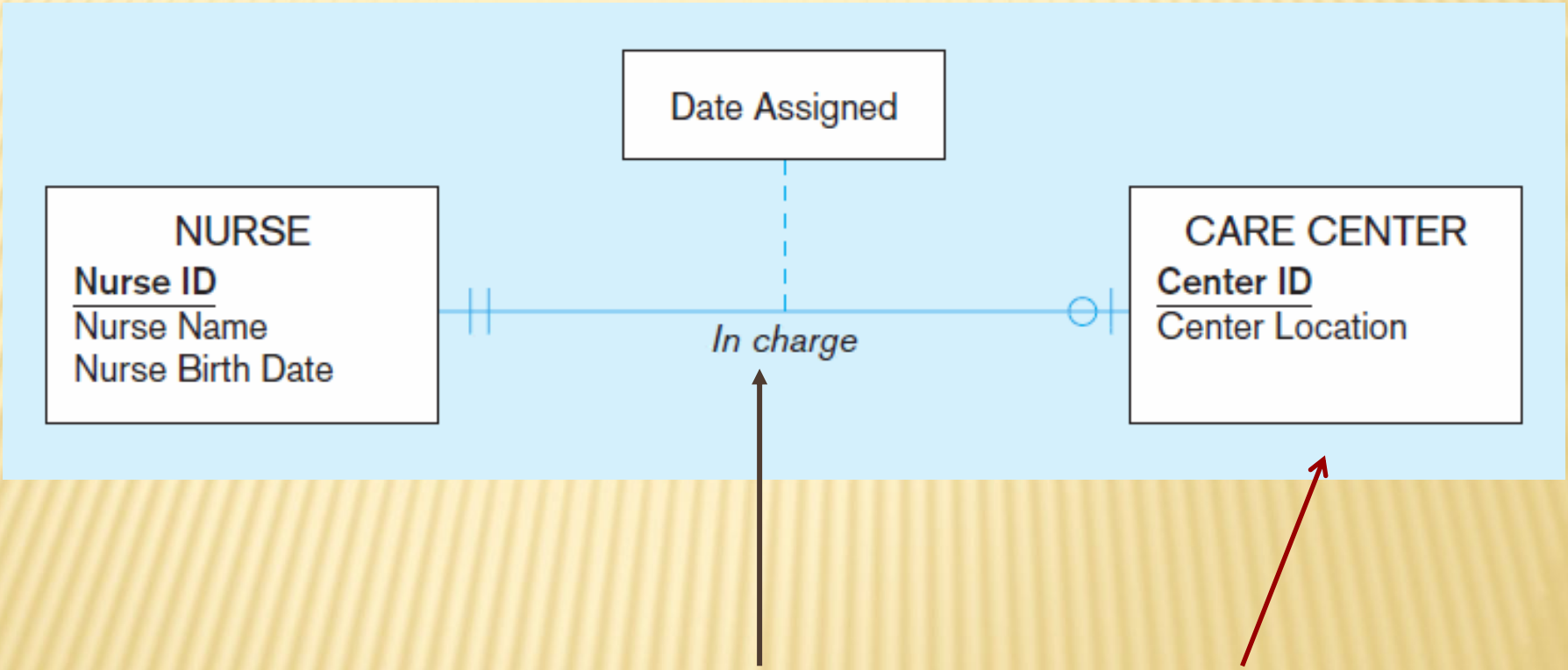


Figure 4-14 Example of mapping a binary 1:1 relationship

a) ***In charge*** relationship (1:1)



Often in 1:1 relationships, **one direction is optional**

MAP BINARY 1:1 RELATIONSHIPS

- In a 1:1 relationship, the association in one direction is often optional one, while the association in the other direction is mandatory one
  - think about STUDENT and PARKING_PERMIT
- The primary key of one of the relations is included as a foreign key in the other relation
  - The primary key of the Mandatory side ➔
  - ➔ The foreign key of the optional side

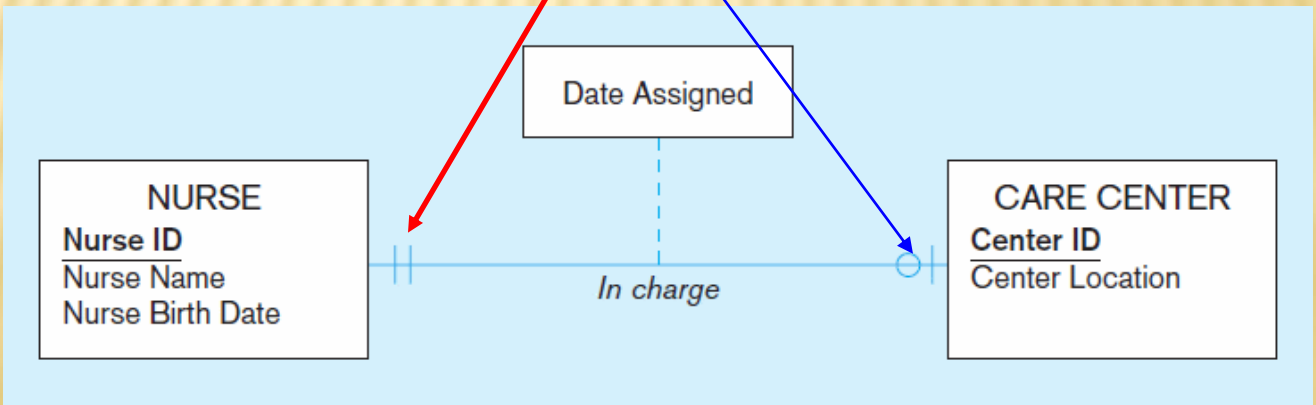


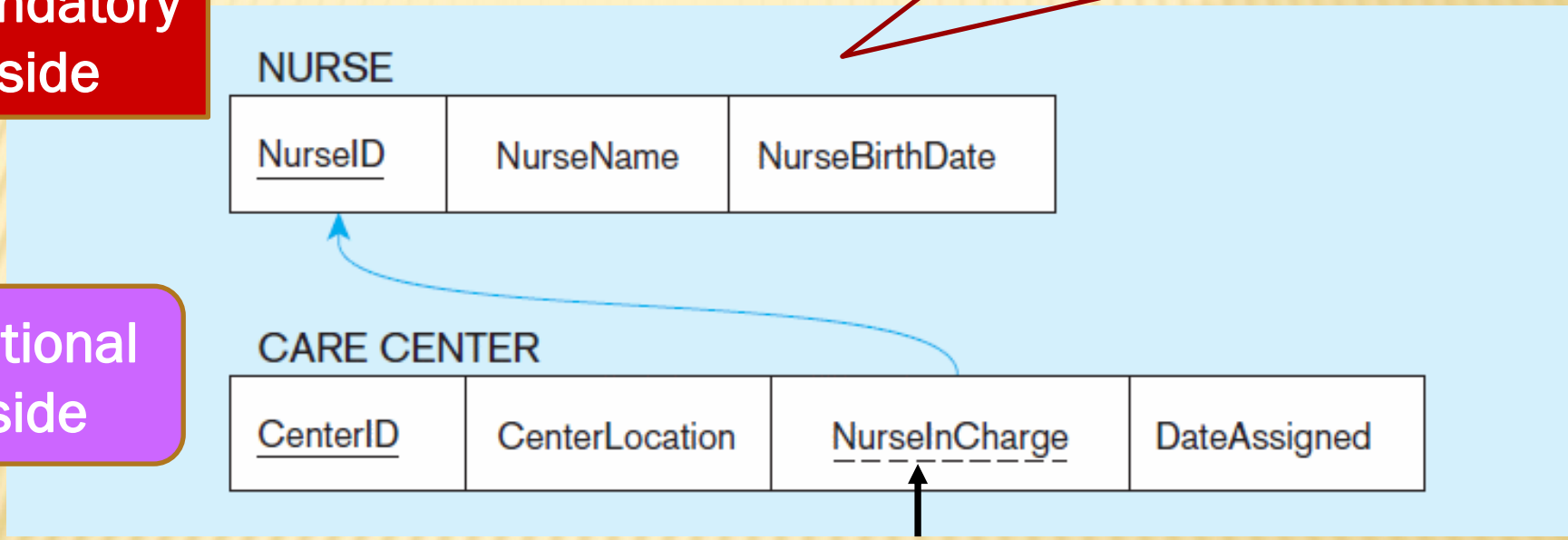
Figure 4-14 Example of mapping a binary 1:1 relationship (cont.)

b) Resulting relations

Arrow points from _____ side to _____ side, from _____ key to _____ key

Mandatory side

Optional side



Foreign key goes in the relation on the **optional side**, matching the primary key on the mandatory side

Practice the same for the STUDENT and PARKING_PERMIT relationship

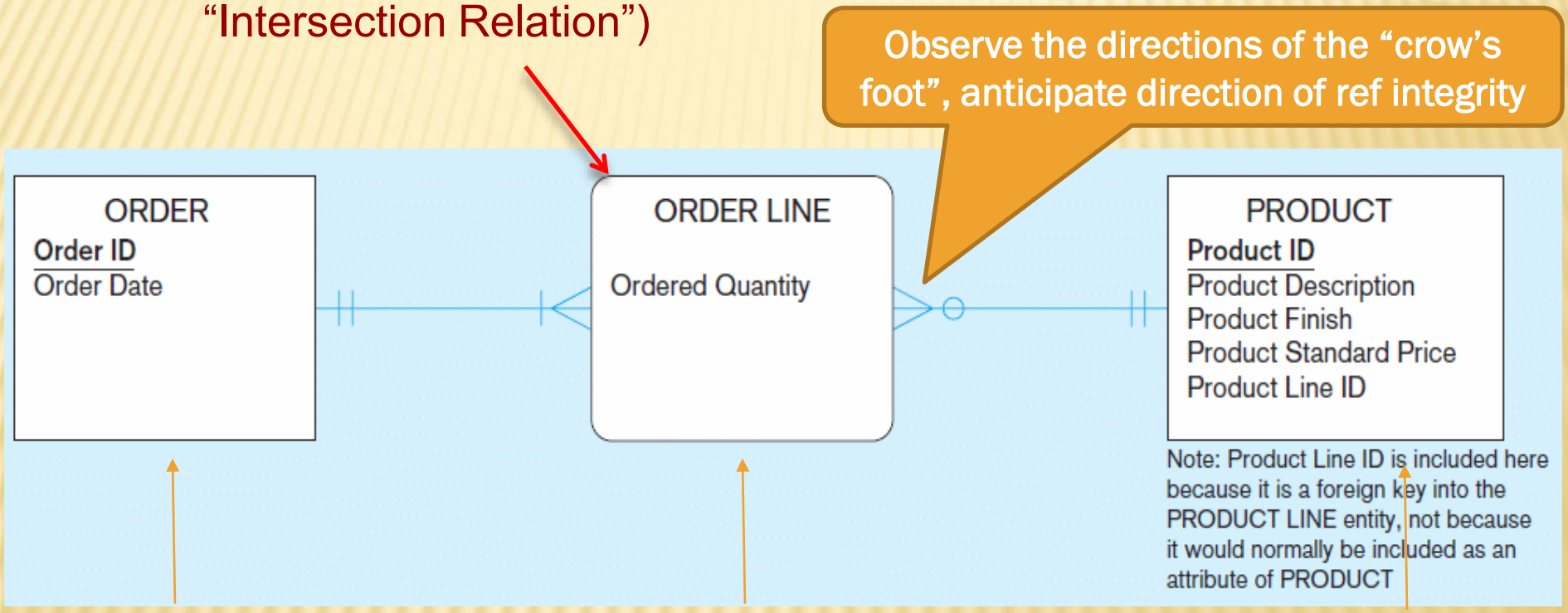
TRANSFORMING EER DIAGRAMS INTO RELATIONS (CONT.)

Mapping Associative Entities

- Scenario 1: 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)
- Scenario 2: Identifier Assigned
  - It is natural and familiar to end-users
  - Default identifier may not be unique
  - What should we do with the foreign key(?)?

Figure 4-15 Example of mapping an associative entity

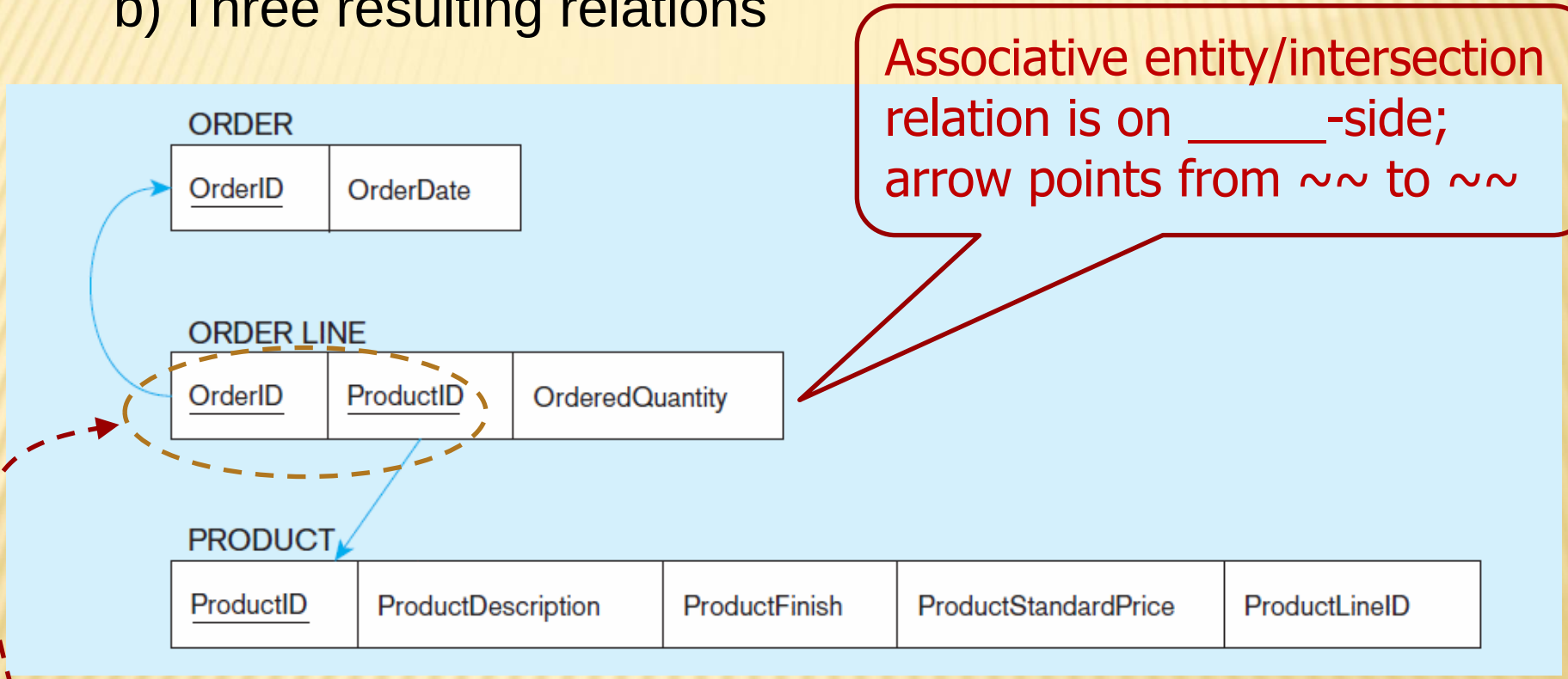
a) An associative entity (will be transformed to an "Intersection Relation")



Regular entity	Associative entity	Regular entity
Regular relation	Intersection relation	Regular relation

Figure 4-15 Example of mapping an associative entity (cont.)

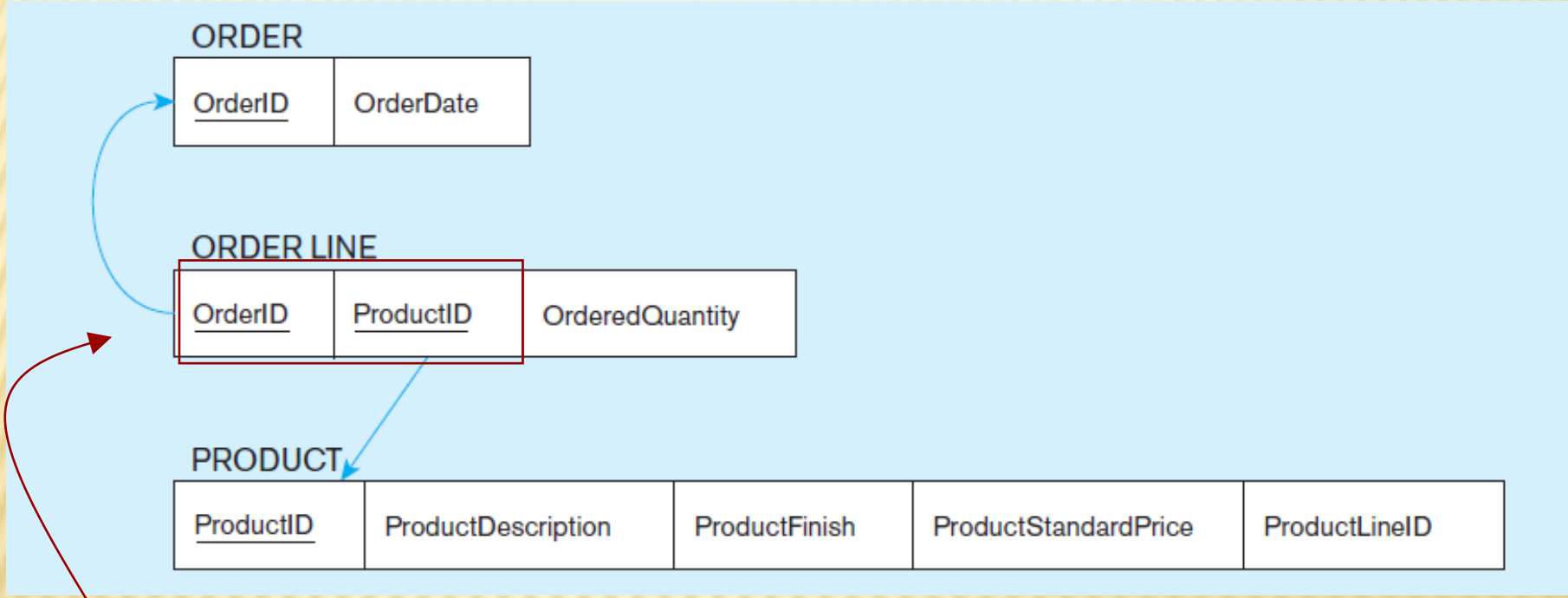
b) Three resulting relations



1. Composite primary key - two primary key attributes from the other two relations
2. **TWO** foreign keys, referencing the other two entities

Figure 4-15 Example of mapping an associative entity (cont.)

b) Three resulting relations



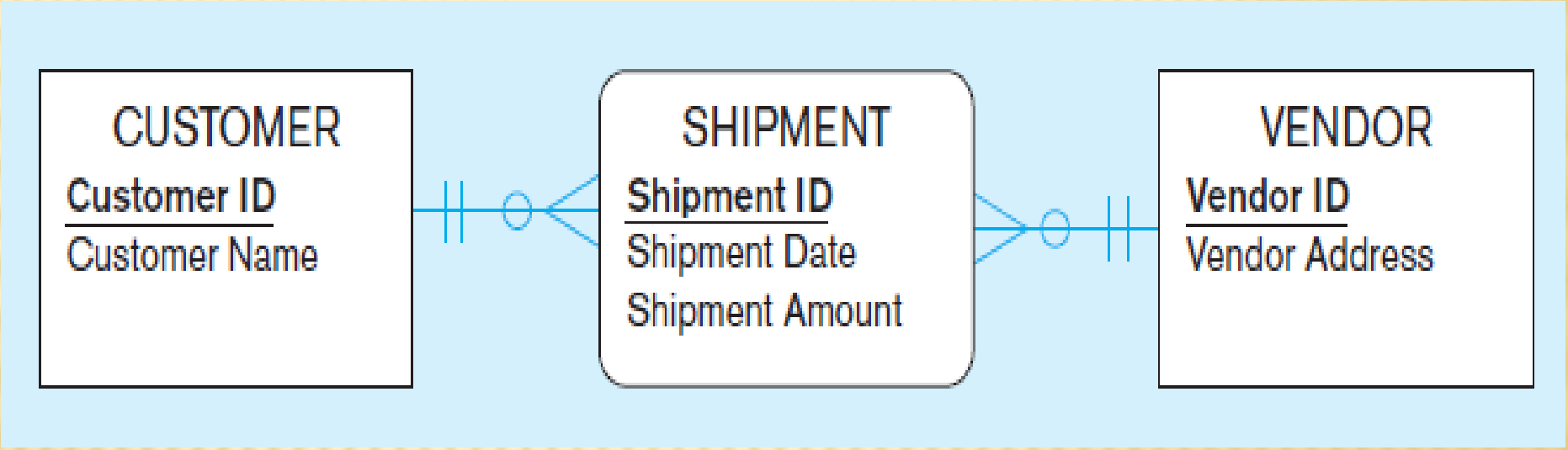
OID + PID:

(1) Together - composite primary key (PK);

(2) Separate - 2 FKs pointing to ORDER and PRODUCT tables

Figure 4-16 Example of mapping an associative entity with an identifier

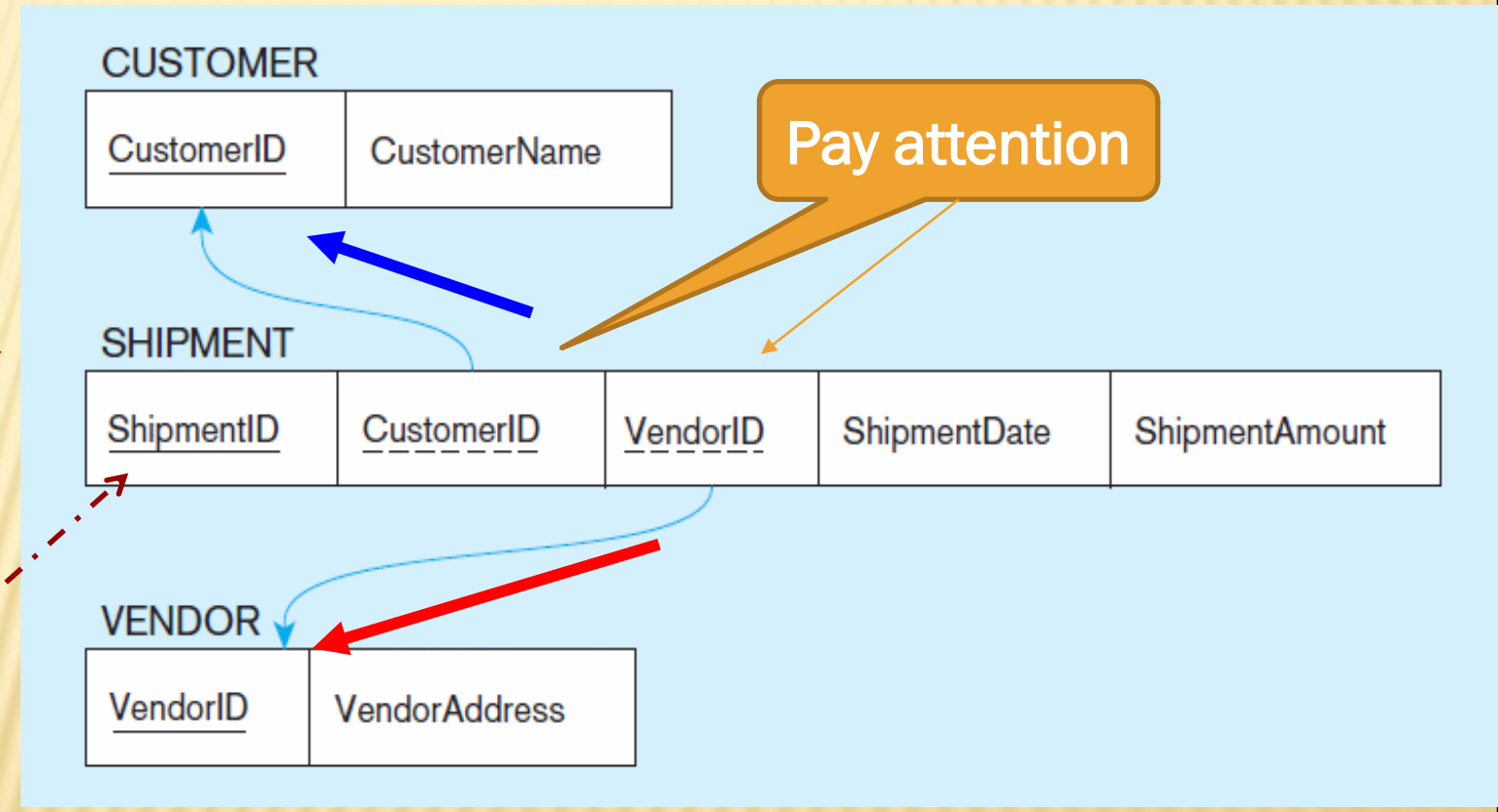
a) SHIPMENT associative entity



Associative entity: ALWAYS on M-side

Figure 4-16 Example of mapping an associative entity with an identifier (cont.)

b) Three resulting relations



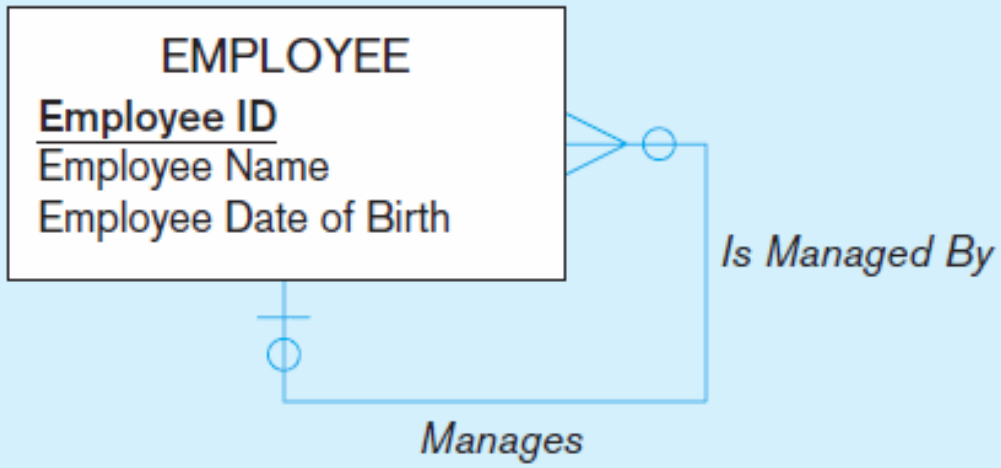
Primary key differs from foreign keys - **identifier assigned**

TRANSFORMING EER DIAGRAMS INTO RELATIONS (CONT.)

Mapping Unary Relationships

- One-to-Many: ***Recursive*** foreign key in the same relation
- Many-to-Many: 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

Figure 4-17 Mapping a unary 1:M relationship



(a) EMPLOYEE entity with unary relationship

(b) EMPLOYEE relation with recursive foreign key

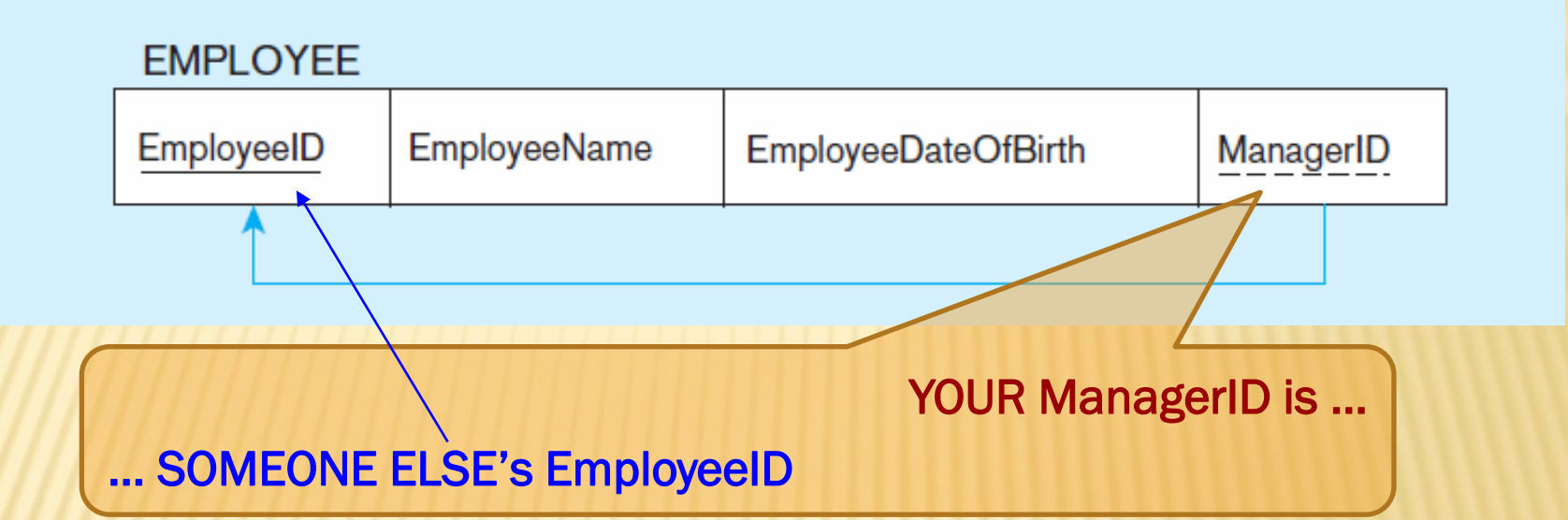
EMPLOYEE

EmployeeID	EmployeeName	EmployeeDateOfBirth	ManagerID

Note the direction of the arrow!

Examine the tables: ...

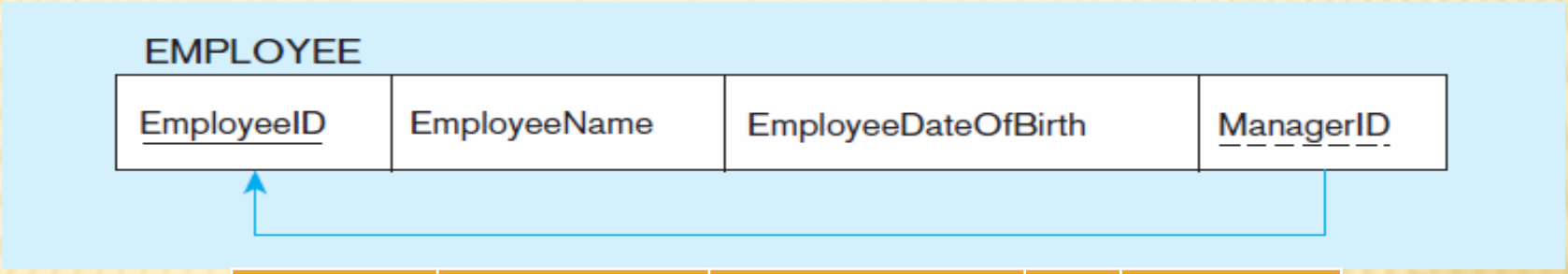
Figure 4-17 Mapping a unary 1:M – More explanation



Recursive is NOT "circular":

- A record's ManagerID does **NOT** point to **ITS OWN** EmployeeID;
- It points to SOMEONE ELSE's EmployeeID – the EmployeeID of the person's manager's
- One row points to **ANOTHER row**

Visualizing unary 1:M



EID	LName	Position	...	MgrID
E011	Davis	Acct Mgr		
E014	Edwards	Acct I		E011
E018	Fowler	Acct II		E011
E021	Gonzalez	Mkt Mgr		
E023	Hanks	Mkt Speclst		E021
E027	Kong	Mkt Speclst		E021

Recursive is NOT "circular":

- A record's ManagerID does **NOT** point to **ITS OWN** EmployeeID;
- It points to SOMEONE ELSE's EmployeeID
  - –One row points to **ANOTHER row**

Figure 4-18 Mapping an unary M:N relationship

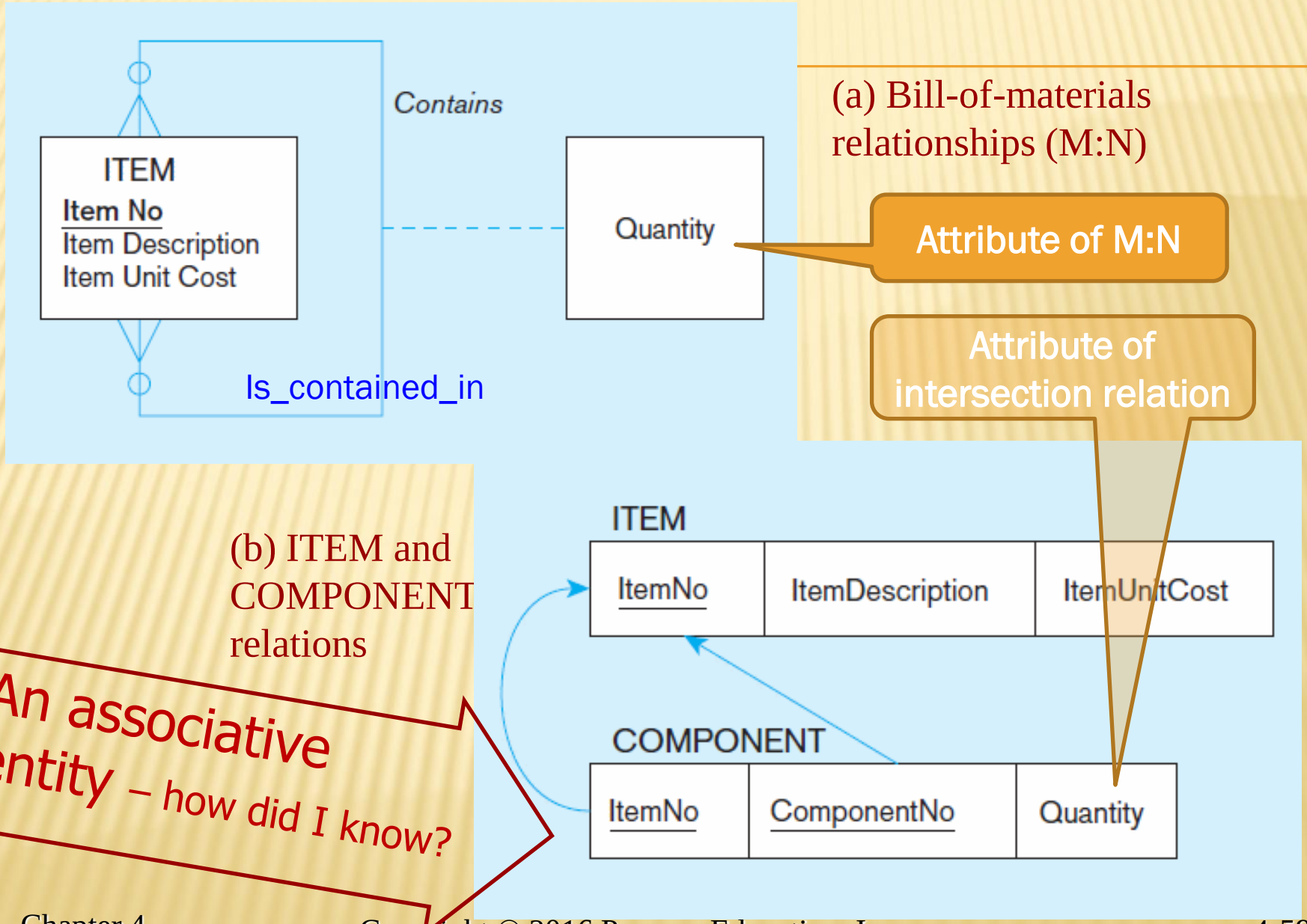
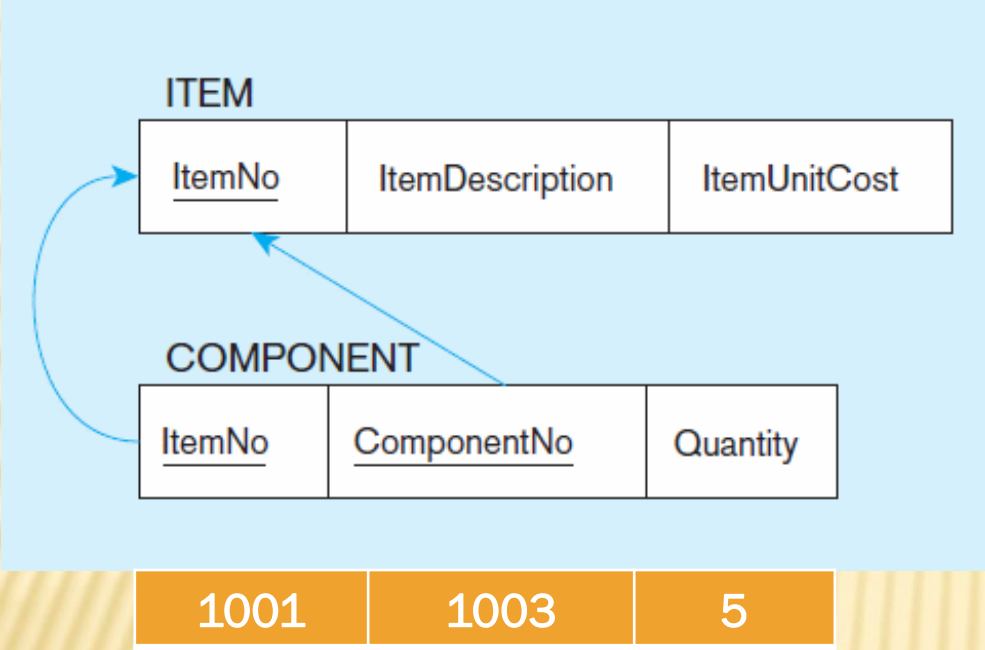
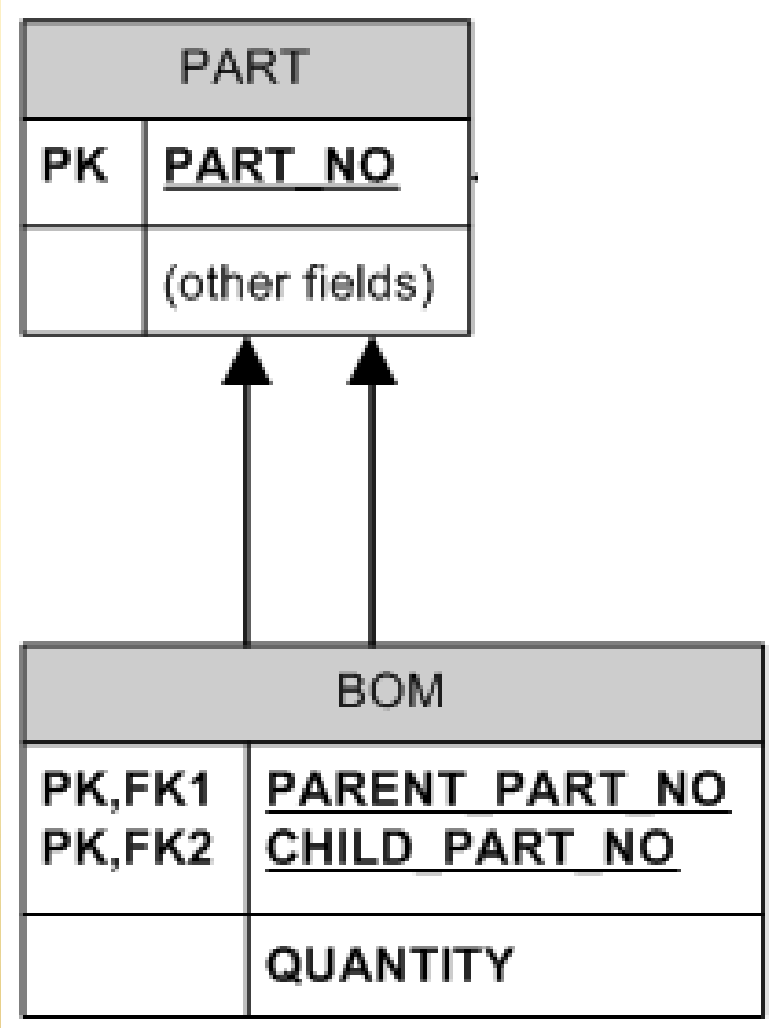


Figure 4-18 Mapping an unary M:N – More explanation





Example:
Item #1001 takes Item #1003 as its component, in the number of 5 (each 1001 needs 5 units of 1003)

Figure 4-18 "B" – "Unfolding" an unary M:N relationship

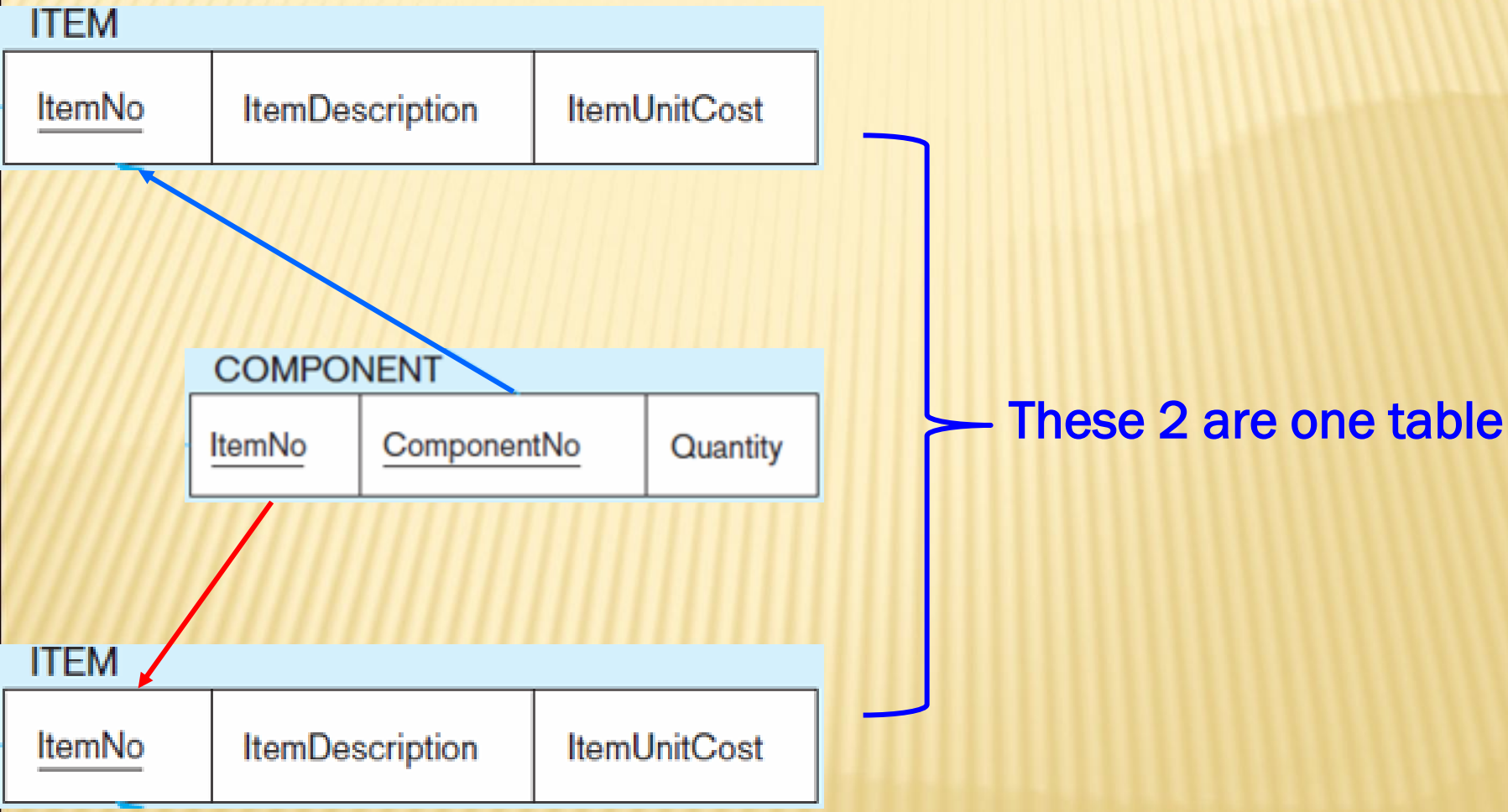


Figure 4-18 Mapping a unary M:N – More explanation

- The PART table is either a top-assembly or a sub-assembly or a leaf part. It uses a publicly known "part number" to identify its rows, which is not actually a number at all and can contain non-numeric characters.
- The BOM table models many-to-many relationship of PART by itself. It's really no different from any other junction table, except both "endpoint" tables are actually the same table. This way, one sub-assembly or part can be reused in multiple parent assemblies.
- On top of this model, you can fairly naturally add things like "drawing position" or "unit of measure" (e.g. a paint can be part of BOM but is measured in "kilograms" instead of "pieces").
- http://stackoverflow.com/questions/17651424/bill-of-materials-database-model

TRANSFORMING EER DIAGRAMS INTO RELATIONS (CONT.)

Mapping Ternary (and n-ary) Relationships

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

Figure 4-19 Mapping a ternary relationship

a) PATIENT TREATMENT Ternary relationship with associative entity

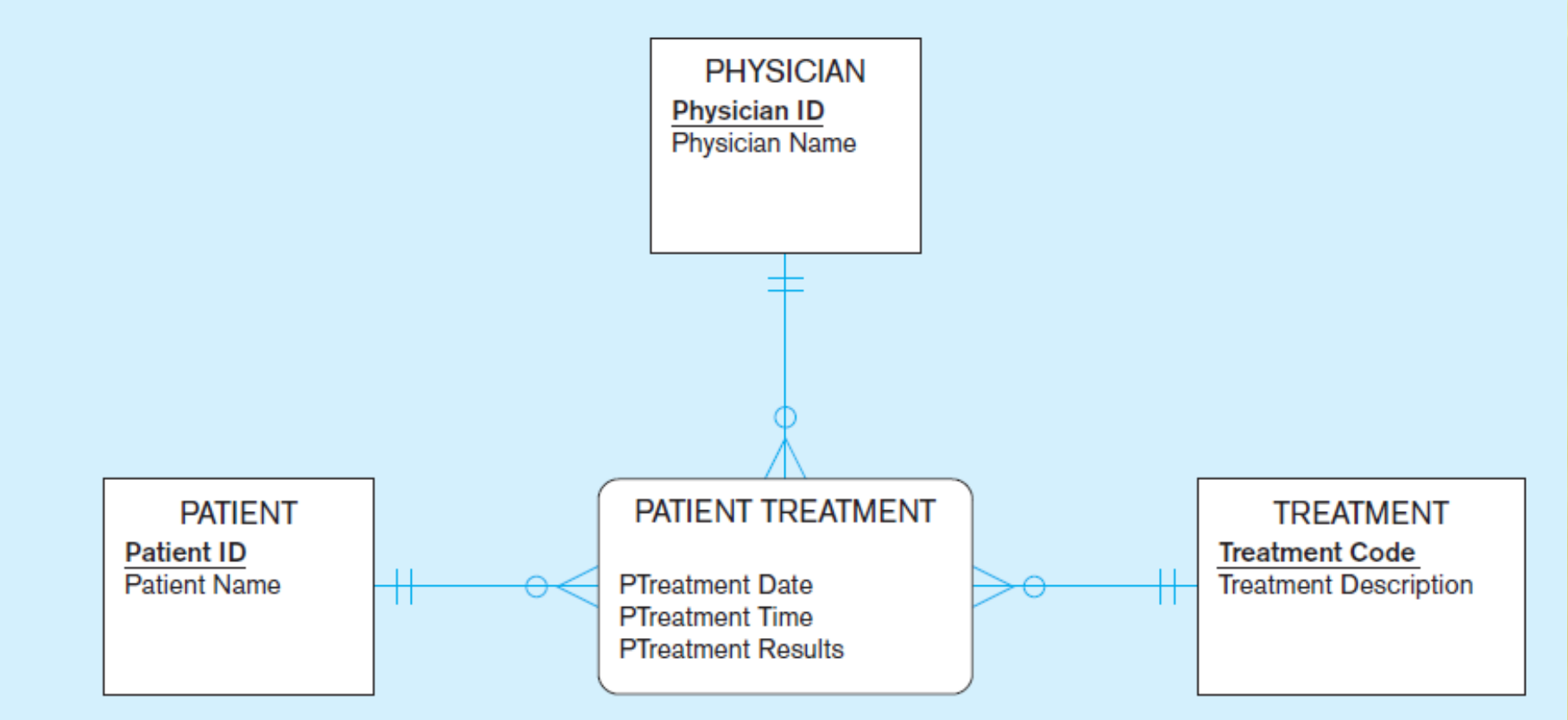
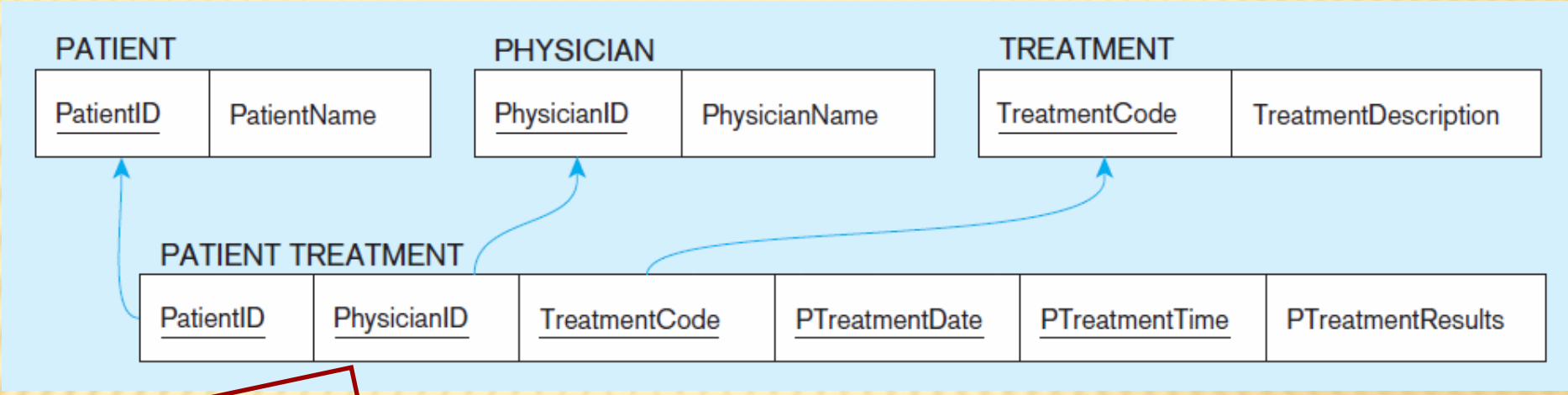


Figure 4-19 Mapping a ternary relationship (cont.)

b) Mapping the ternary relationship PATIENT TREATMENT



An associative entity

Remember that the primary key MUST be unique

This is why treatment date and time are included in the composite primary key

But this makes a cumbersome key…

It would be better to create a surrogate key like Treatment#

TRANSFORMING EER DIAGRAMS INTO RELATIONS (CONT.)

Mapping Supertype/Subtype Relationships

- One relation for supertype and for each subtype
- Supertype attributes (including identifier and subtype discriminator) go into supertype relation
- Subtype attributes go into each subtype; primary key of supertype relation also becomes primary key of subtype relation
- 1:1 relationship established between supertype and each subtype, with supertype as primary table