



# LOGICAL DATABASE DESIGN

***MSBA 7024 / MACC 7020***  
***Database Design and Management***

# OBJECTIVES

- Definition of terms
- List the properties of relations
- State properties of candidate keys
- Create tables with entity and relational integrity constraints
- Transform E-R diagrams to relations
- Define first, second, and third normal form
- Use normalization to convert anomalous tables to well-structured relations
- Describe problems from merging relations

# RELATION

- Definition: A relation is a named, two-dimensional table of data
- NOTE: The word ***relation*** (in relational database) is NOT the same as the word ***relationship*** (in E-R model)

# RELATION

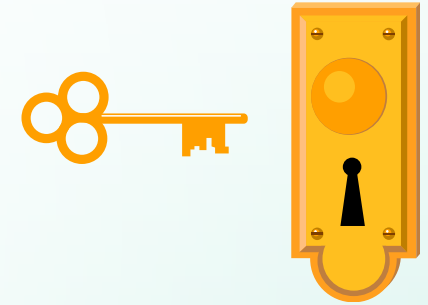
- Table consists of rows (records) and columns (attribute or field)
- Requirements for a table to qualify as a relation:
  - It must have a unique name
  - Every attribute value must be atomic (not multivalued, not composite)
  - Every row must be unique (can't have two rows with exactly the same values for all their fields)
  - Attributes (columns) in tables must have unique names
  - The order of the columns must be irrelevant
  - The order of the rows must be irrelevant

**NOTE: all *relations* are in *1<sup>st</sup> Normal form***

# CORRESPONDENCE WITH E-R MODEL

- Relations (tables) correspond with entity types and with many-to-many relationship types
- Rows correspond with entity instances and with many-to-many relationship instances
- Columns correspond with attributes

# KEY FIELDS



- Keys are special fields that serve two main purposes:
  - **Primary keys** are unique identifiers of the relation in question. Examples include employee numbers, social security numbers, etc. *This is how we can guarantee 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 (more on this in the next handout)

## Schema for four relations (Pine Valley Furniture Company)

### CUSTOMER

<u>CustomerID</u>	CustomerName	CustomerAddress	CustomerCity*	CustomerState*	CustomerPostalCode
-------------------	--------------	-----------------	---------------	----------------	--------------------

Primary Key

### ORDER

<u>OrderID</u>	OrderDate	<u>CustomerID</u>
----------------	-----------	-------------------

Foreign Key

(implements 1:N relationship  
between customer and order)

### ORDER LINE

<u>OrderID</u>	<u>ProductID</u>	OrderedQuantity
----------------	------------------	-----------------

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

### PRODUCT

<u>ProductID</u>	ProductDescription	ProductFinish	ProductStandardPrice	ProductLineID
------------------	--------------------	---------------	----------------------	---------------

# INTEGRITY CONSTRAINTS

- Domain Constraints
  - Allowable values for an attribute. See Table on next slide
- Entity Integrity
  - No primary key attribute may be null. All primary key fields **MUST** have data
- Action Assertions
  - Business rules. Recall from previous handout



**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 – rule states that any foreign key value (on 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 “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

# Referential integrity constraints (Pine Valley Furniture)

## CUSTOMER

<u>CustomerID</u>	CustomerName	CustomerAddress	CustomerCity	CustomerState	CustomerPostalCode
-------------------	--------------	-----------------	--------------	---------------	--------------------

## ORDER

<u>OrderID</u>	OrderDate	<u>CustomerID</u>
----------------	-----------	-------------------

## ORDER LINE

<u>OrderID</u>	<u>ProductID</u>	OrderedQuantity
----------------	------------------	-----------------

## PRODUCT

<u>ProductID</u>	ProductDescription	ProductFinish	ProductStandardPrice	ProductLineID
------------------	--------------------	---------------	----------------------	---------------

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

# 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

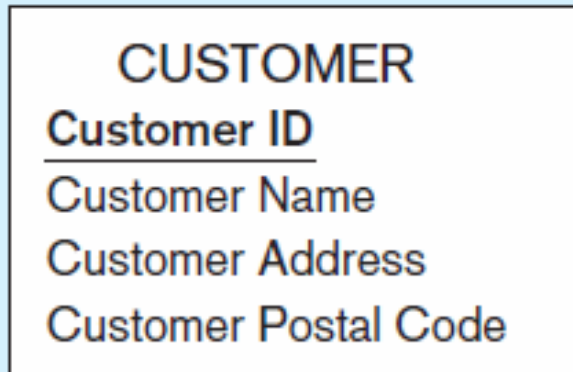
# TRANSFORMING ER DIAGRAMS INTO RELATIONS

## Mapping Regular Entities to Relations

1. Simple attributes: E-R attributes map directly onto the relation
2. Composite attributes: Use only their simple, component attributes
3. Multivalued Attribute: Becomes a separate relation with a foreign key taken from the superior entity

# Mapping a regular entity

**(a) CUSTOMER  
entity type with  
simple  
attributes**



**(b) CUSTOMER relation**

CUSTOMER			
<u>CustomerID</u>	CustomerName	CustomerAddress	CustomerPostalCode



# Mapping a composite attribute

(a) CUSTOMER  
entity type with  
composite  
attribute

**CUSTOMER**  
Customer ID  
Customer Name  
Customer Address  
(CustomerStreet, CustomerCity, CustomerState)  
Customer Postal Code

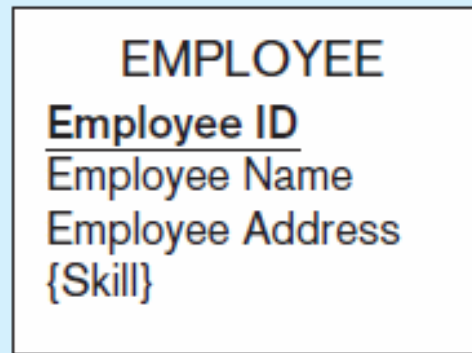
(b) CUSTOMER relation with address detail

**CUSTOMER**

<u>CustomerID</u>	CustomerName	CustomerStreet	CustomerCity	CustomerState	CustomerPostalCode
-------------------	--------------	----------------	--------------	---------------	--------------------

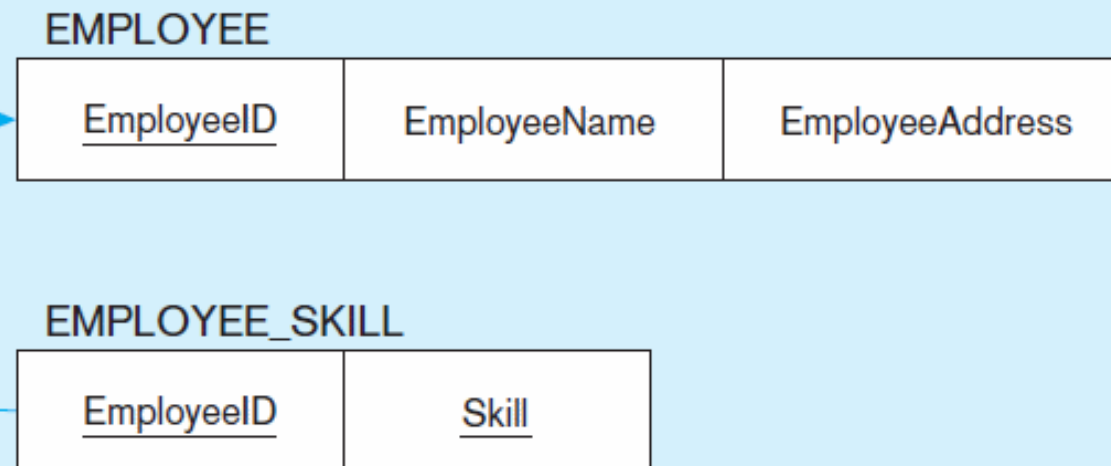
# Mapping an entity with a multivalued attribute

(a)



**Multivalued attribute becomes a separate relation with foreign key**

(b)



**One-to-many relationship between original entity and new relation**



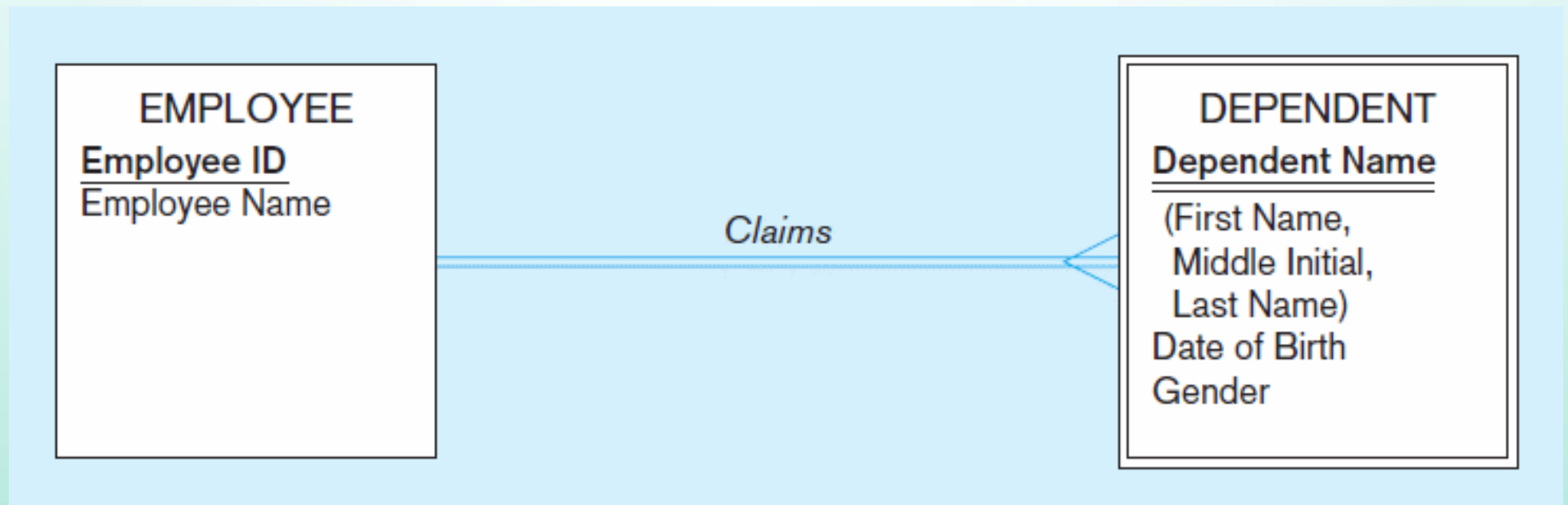
# TRANSFORMING ER DIAGRAMS INTO RELATIONS (CONT.)

## Mapping Weak Entities

- Becomes a separate relation 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 of mapping a weak entity

### a) Weak entity DEPENDENT



## Example of mapping a weak entity (cont.)

### b) Relations resulting from weak entity

#### EMPLOYEE

<u>EmployeeID</u>	EmployeeName
-------------------	--------------

NOTE: the domain constraint for the foreign key should NOT allow *null* value if DEPENDENT is a weak entity

#### DEPENDENT

<u>FirstName</u>	<u>MiddleInitial</u>	<u>LastName</u>	<u>EmployeeID</u>	DateOfBirth	Gender
------------------	----------------------	-----------------	-------------------	-------------	--------

Foreign key

Composite primary key

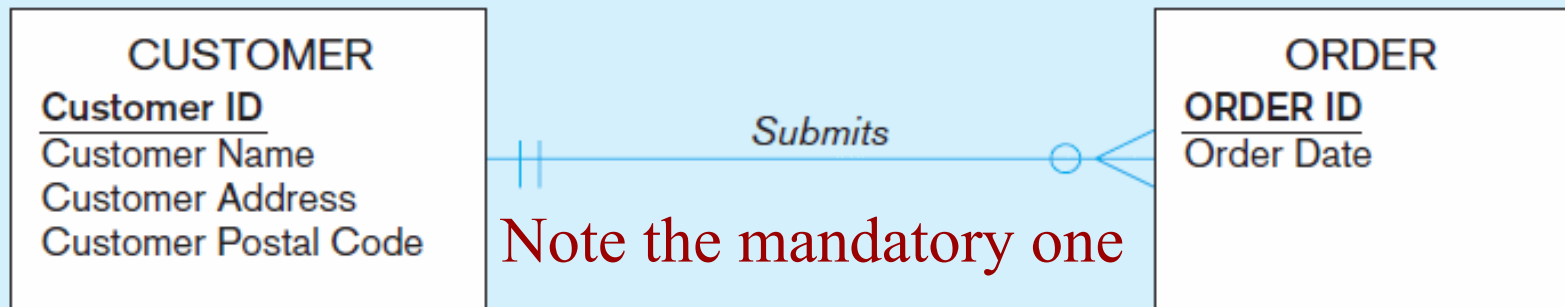
# TRANSFORMING ER DIAGRAMS INTO RELATIONS (CONT.)

## Mapping Binary Relationships

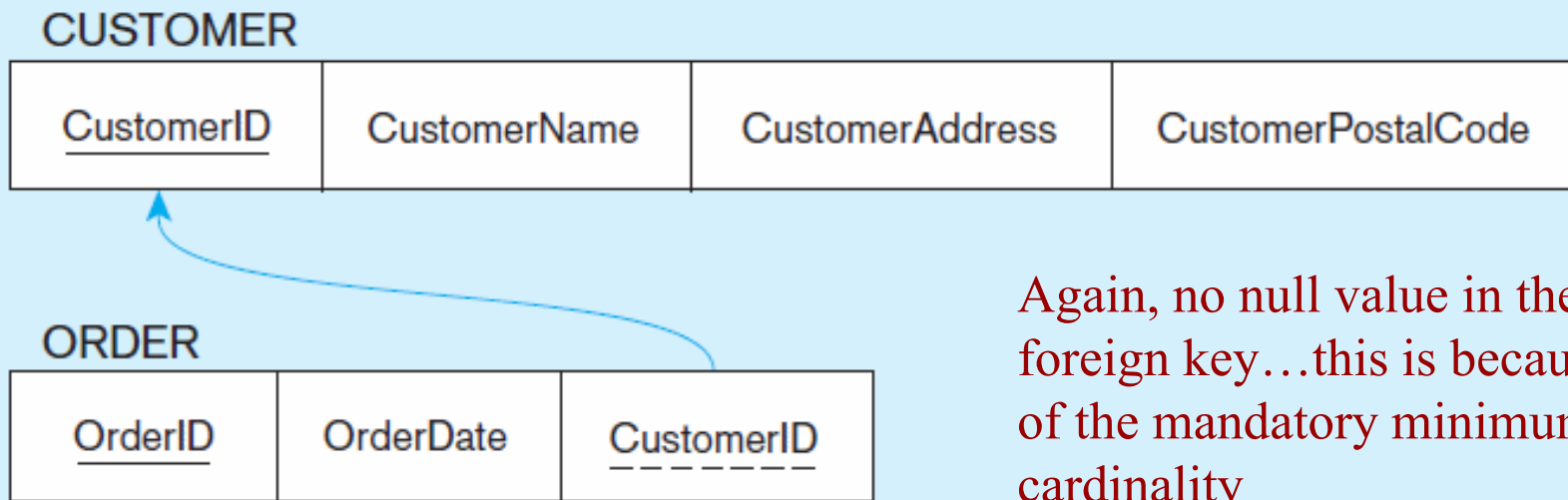
- One-to-Many – Primary key on the one side becomes a foreign key on the many side
- Many-to-Many – Create a ***new relation*** with the primary keys of the two entities as its primary key
- One-to-One – Primary key on the mandatory side becomes a foreign key on the optional side

## Example of mapping a 1:M relationship

### a) Relationship between customers and orders



### b) Mapping the relationship

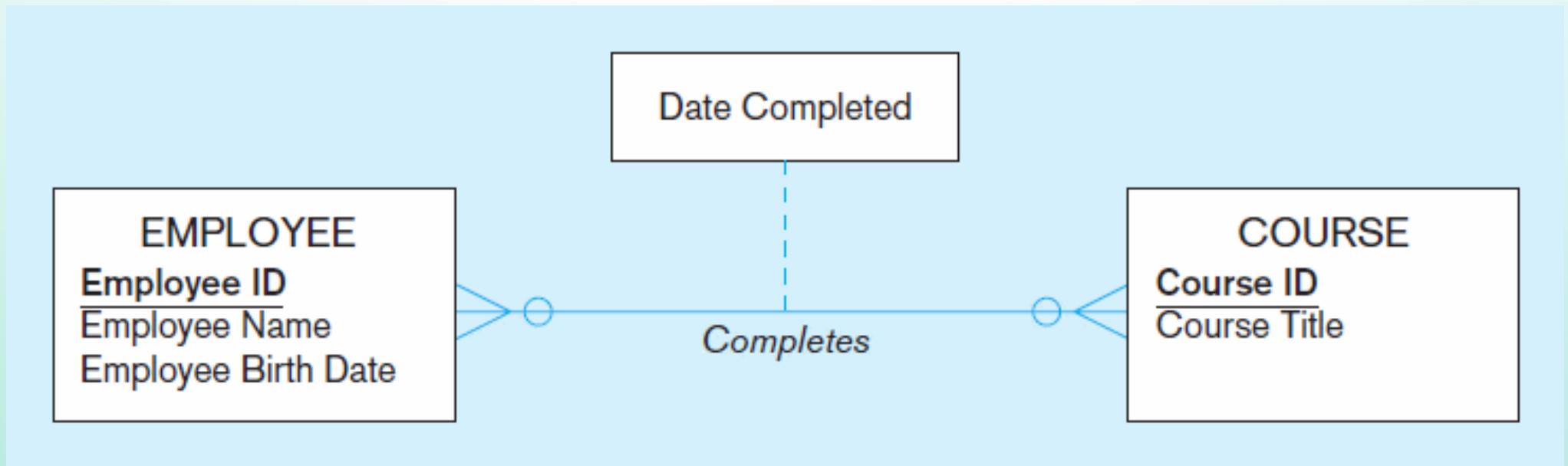


Foreign key

Again, no null value in the foreign key...this is because of the mandatory minimum cardinality

## Example of mapping an M:N relationship

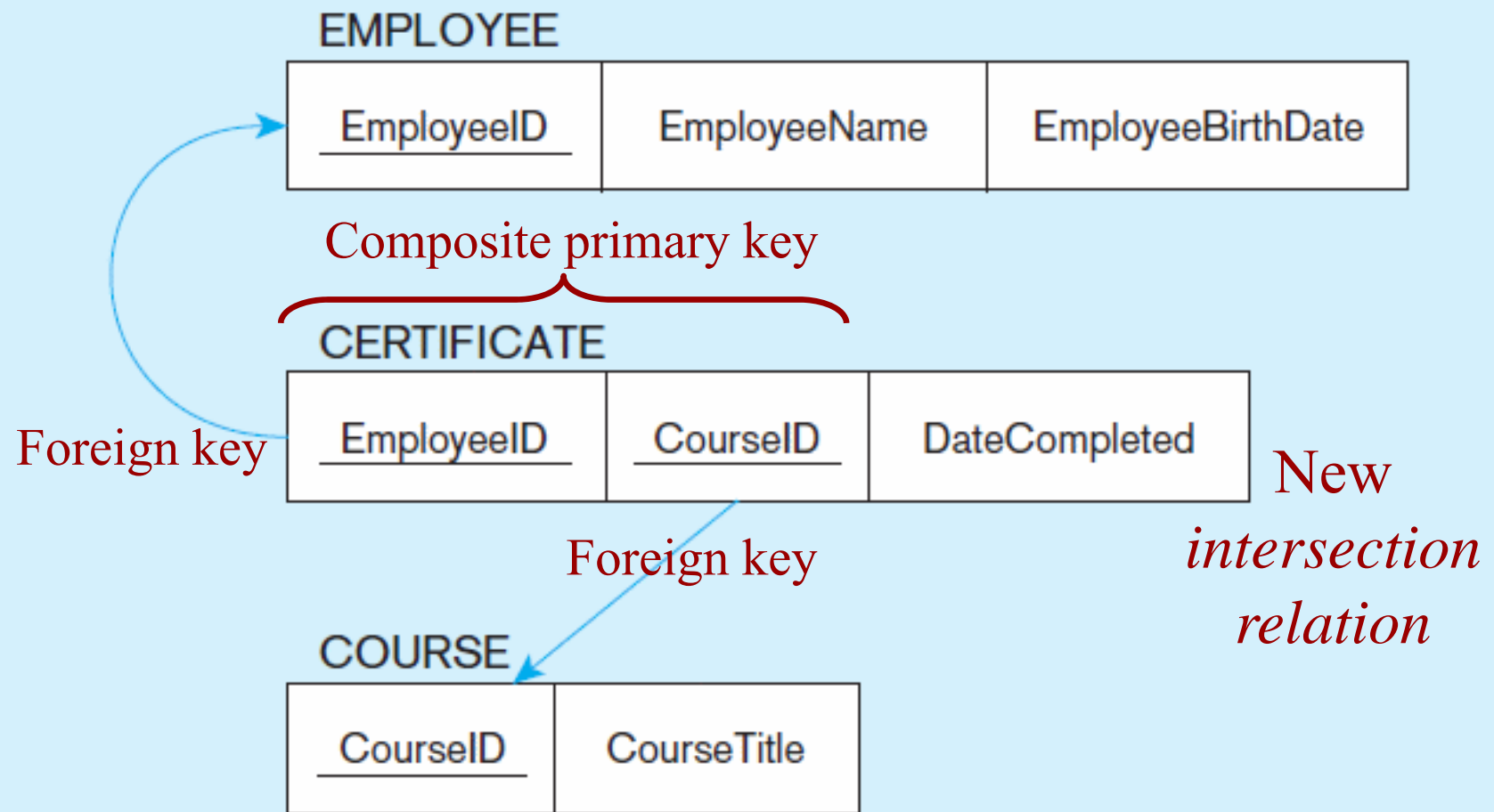
### a) Completes relationship (M:N)



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

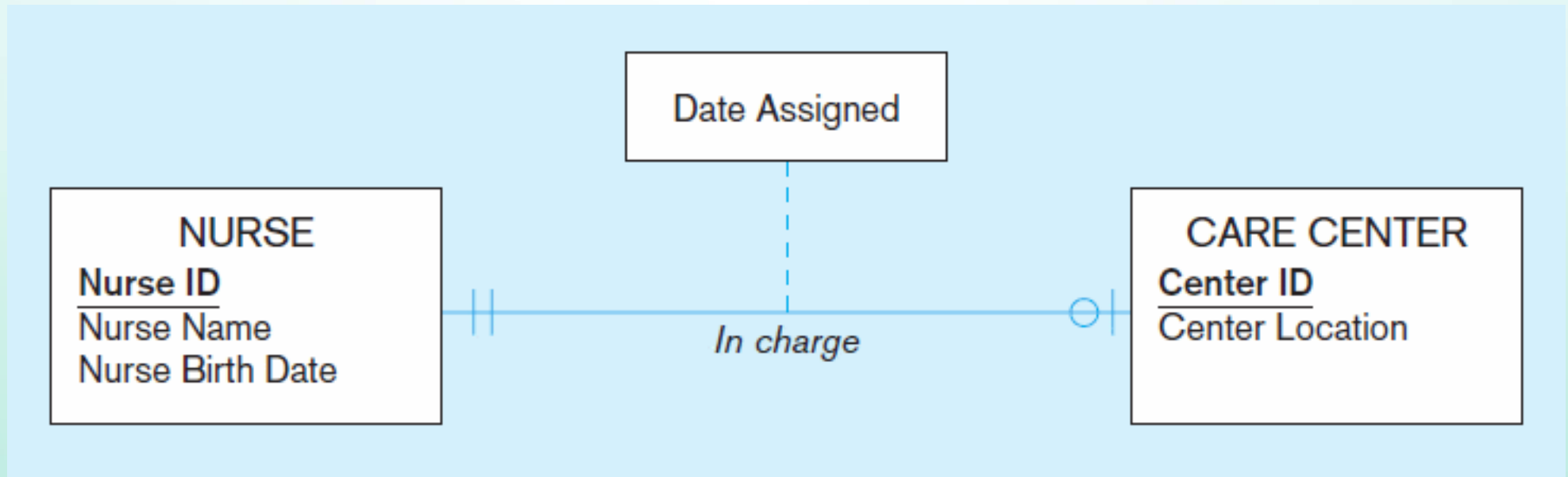
## Example of mapping an M:N relationship (cont.)

### b) Three resulting relations



## Example of mapping a binary 1:1 relationship

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

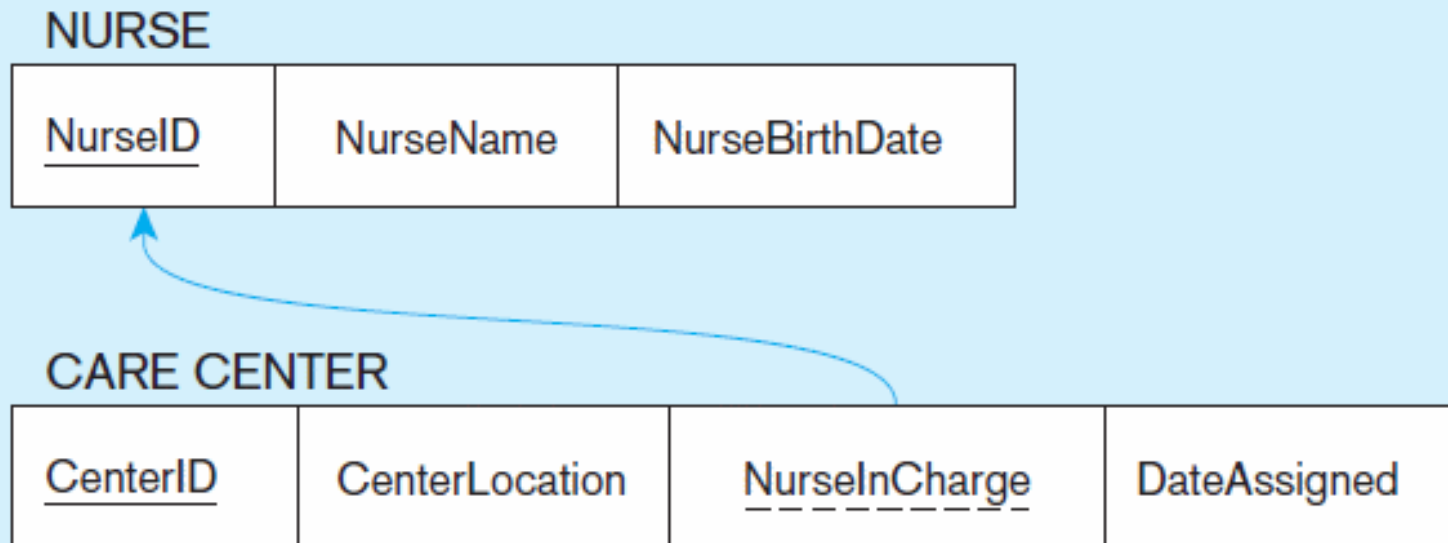


Often in 1:1 relationships, one direction is optional.



## Example of mapping a binary 1:1 relationship (cont.)

### b) Resulting relations



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

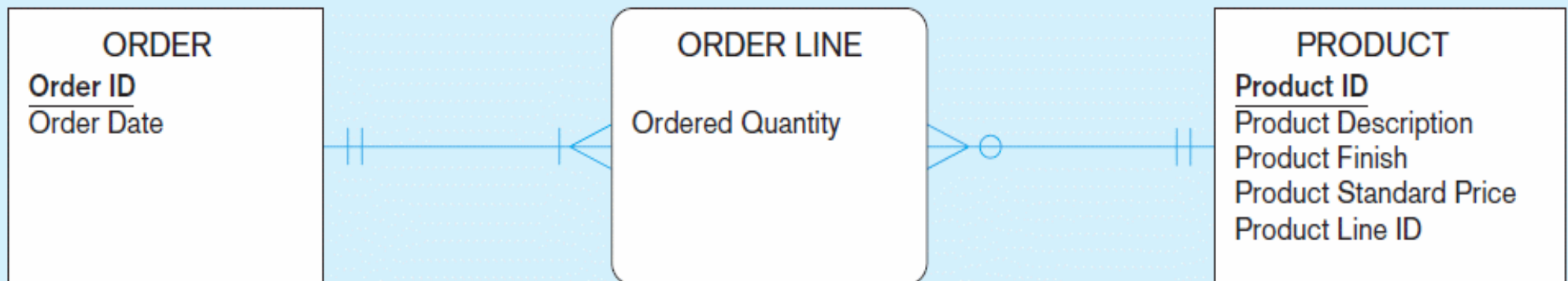
# TRANSFORMING ER DIAGRAMS INTO RELATIONS (CONT.)

## Mapping Associative Entities

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

# Example of mapping an associative entity

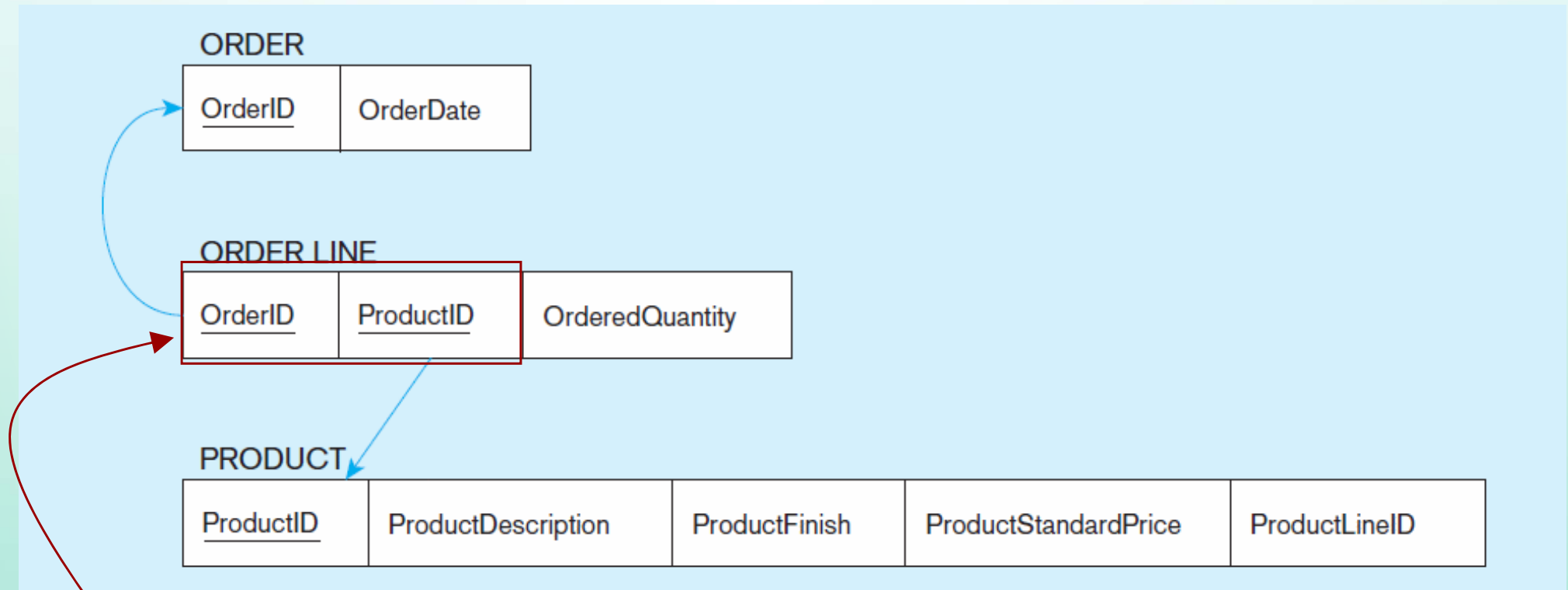
## a) An associative entity



Note: Product Line ID is included here because it is a foreign key into the PRODUCT LINE entity, not because it would normally be included as an attribute of PRODUCT

## Example of mapping an associative entity (cont.)

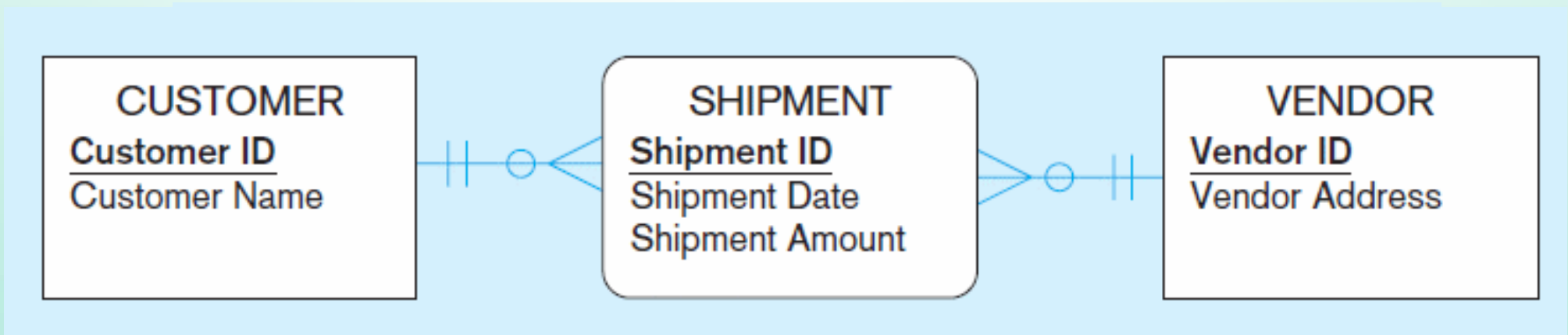
### b) Three resulting relations



Composite primary key formed from the two foreign keys

## Example of mapping an associative entity with an identifier

### a) SHIPMENT associative entity



## Example of mapping an associative entity with an identifier (cont.)

### b) Three resulting relations

CUSTOMER

<u>CustomerID</u>	CustomerName
-------------------	--------------

SHIPMENT

<u>ShipmentID</u>	<u>CustomerID</u>	<u>VendorID</u>	ShipmentDate	ShipmentAmount
-------------------	-------------------	-----------------	--------------	----------------

VENDOR

<u>VendorID</u>	VendorAddress
-----------------	---------------

Primary key differs from foreign keys

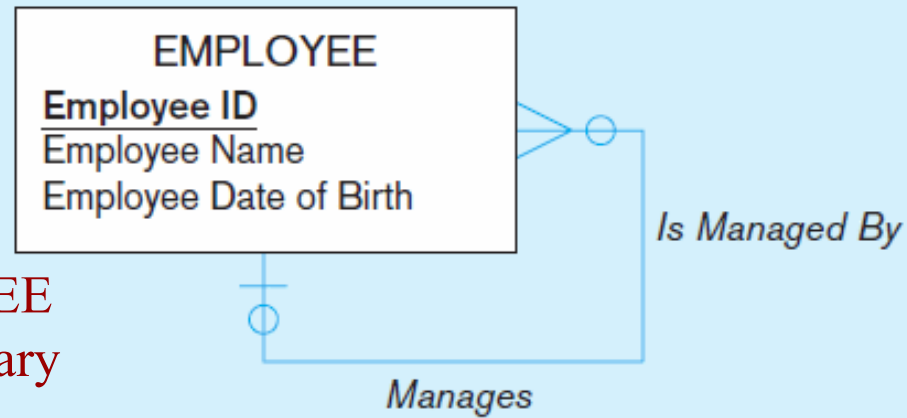
# TRANSFORMING ER DIAGRAMS INTO RELATIONS (CONT.)

## Mapping Unary Relationships

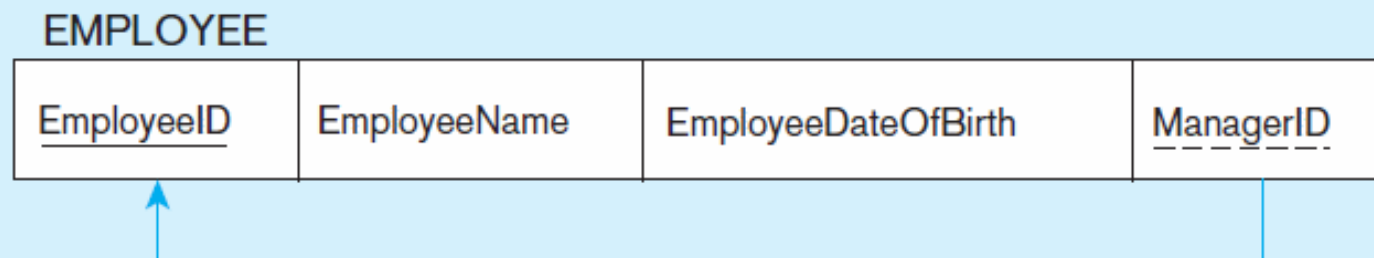
- One-to-Many – Recursive foreign key in the same relation
- Many-to-Many – Two relations:
  - One for the entity type
  - One for an associative relation in which the primary key has two attributes, both taken from the primary key of the entity

# Mapping a unary 1:N relationship

(a) EMPLOYEE  
entity with unary  
relationship

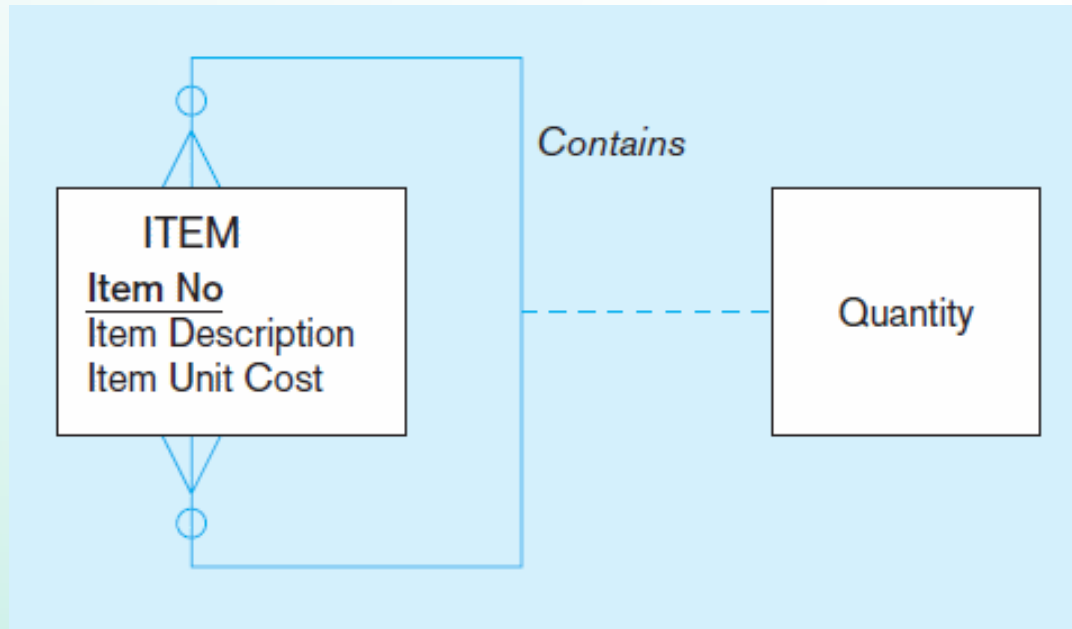


(b)  
EMPLOYEE  
relation with  
recursive  
foreign key



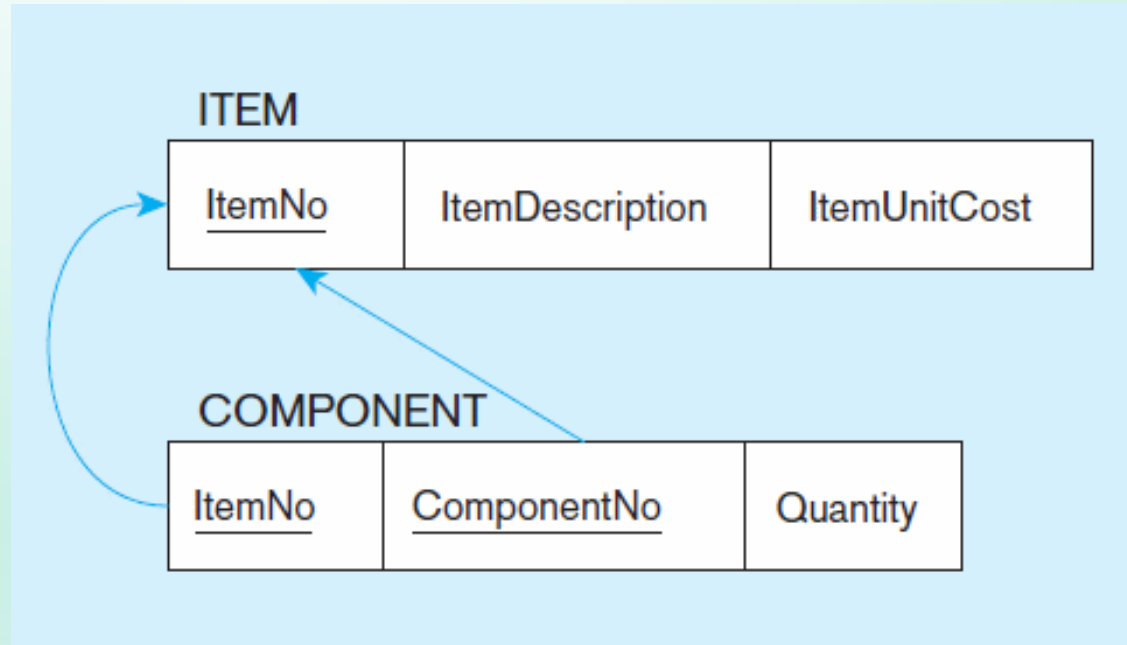


# Mapping a unary M:N relationship



(a) Bill-of-materials relationships (M:N)

(b) ITEM and COMPONENT relations



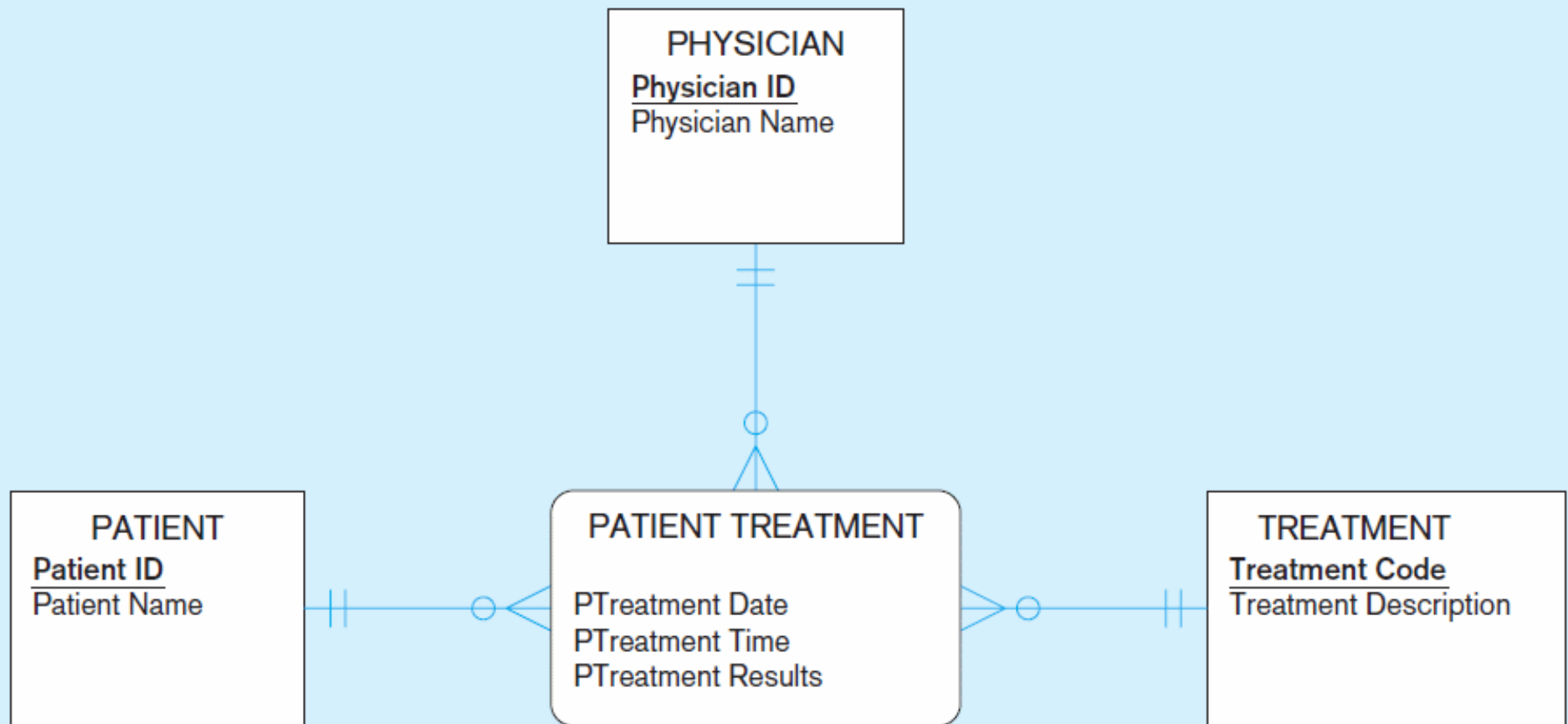
# TRANSFORMING ER DIAGRAMS INTO RELATIONS (CONT.)

## Mapping Ternary (and n-ary) Relationships

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

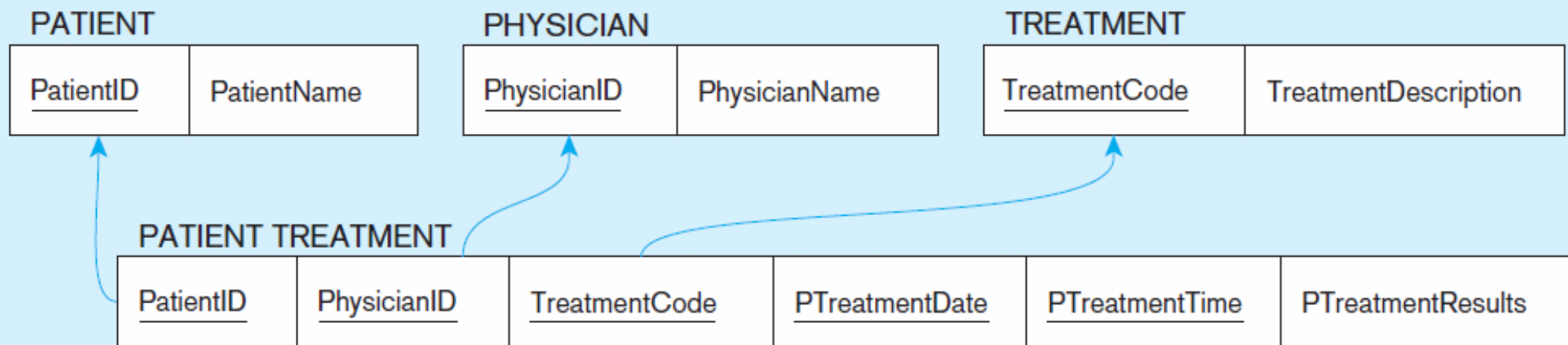
## Mapping a ternary relationship

### a) PATIENT TREATMENT Ternary relationship with associative entity



## Mapping a ternary relationship (cont.)

### b) Mapping the ternary relationship PATIENT TREATMENT



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  
very  
cumbersome  
key...

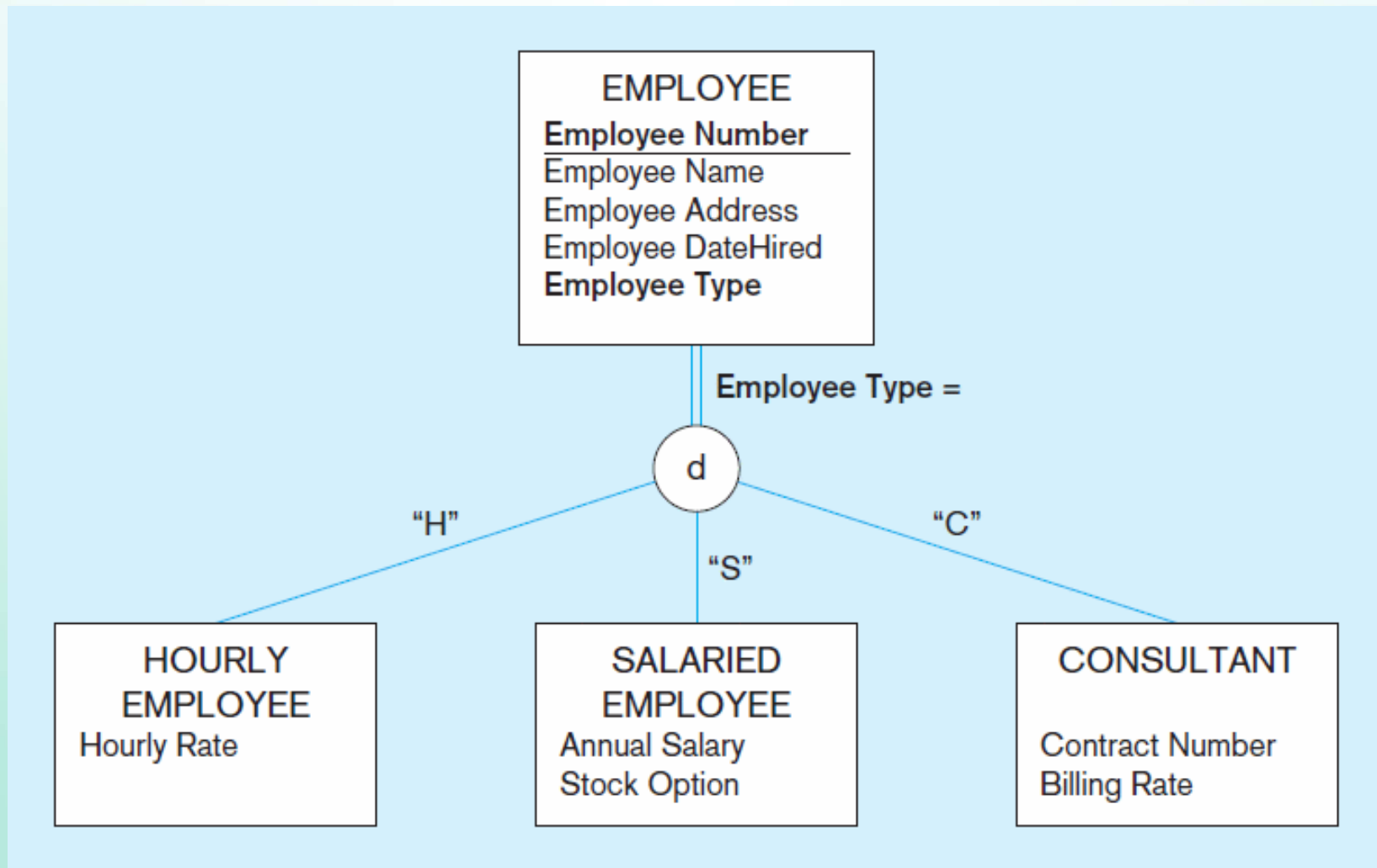
It would be  
better to create a  
surrogate key  
like Treatment#

# TRANSFORMING ER DIAGRAMS INTO RELATIONS (CONT.)

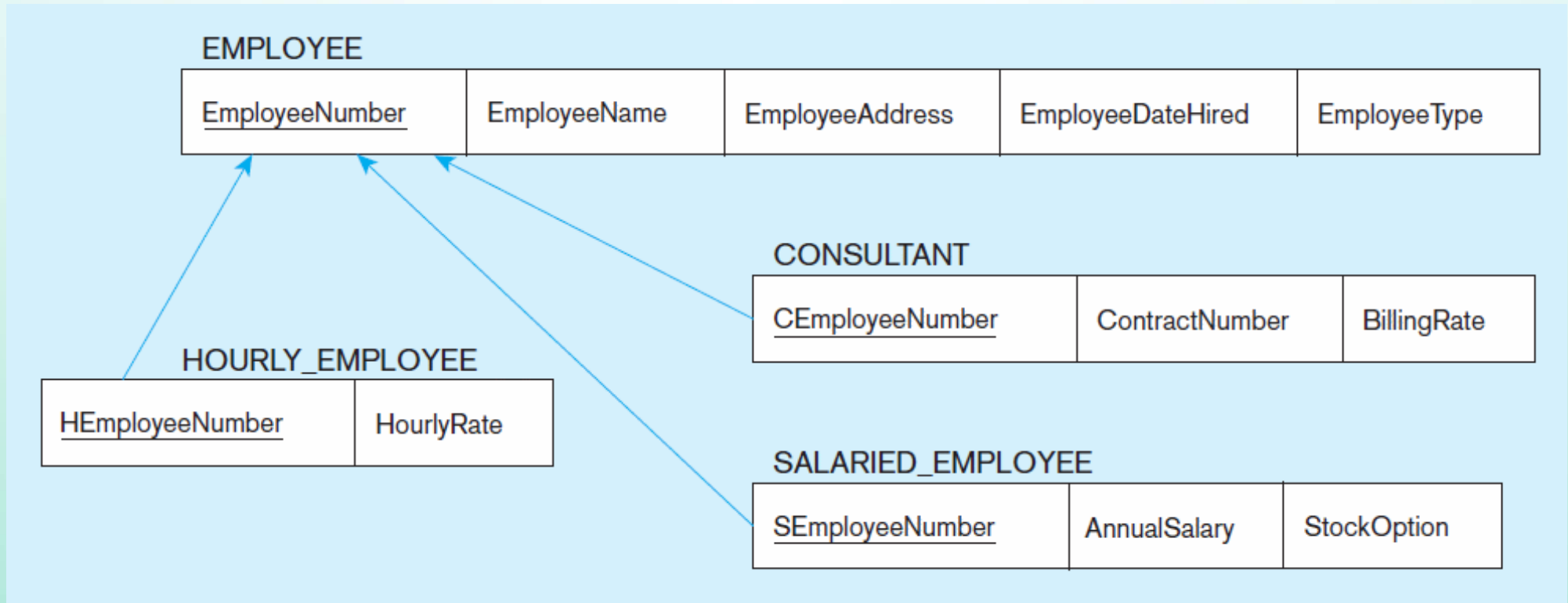
## Mapping Supertype/Subtype Relationships

- 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

# Supertype/subtype relationships



# Mapping Supertype/subtype relationships to relations



These are implemented as one-to-one relationships

# DATA NORMALIZATION

- Primarily a tool to validate and improve a logical design so that it satisfies certain constraints that ***avoid unnecessary duplication of data***
- The process of decomposing relations with anomalies to produce smaller, ***well-structured*** relations



# WELL-STRUCTURED RELATIONS

- A relation that contains minimal data redundancy and allows users to insert, delete, and update rows without causing data inconsistencies
- Goal is to avoid anomalies
  - **Insertion Anomaly**—adding new rows forces user to create duplicate data
  - **Deletion Anomaly**—deleting rows may cause a loss of data that would be needed for other future rows
  - **Modification Anomaly**—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**

# EXAMPLE

EMPLOYEE2

<u>EmpID</u>	Name	DeptName	Salary	<u>CourseTitle</u>	DateCompleted
100	Margaret Simpson	Marketing	48,000	SPSS	6/19/201X
100	Margaret Simpson	Marketing	48,000	Surveys	10/7/201X
140	Alan Beeton	Accounting	52,000	Tax Acc	12/8/201X
110	Chris Lucero	Info Systems	43,000	Visual Basic	1/12/201X
110	Chris Lucero	Info Systems	43,000	C++	4/22/201X
190	Lorenzo Davis	Finance	55,000		
150	Susan Martin	Marketing	42,000	SPSS	6/19/201X
150	Susan Martin	Marketing	42,000	Java	8/12/201X

Question—Is this a relation?

Answer—Yes: Unique rows and no multivalued attributes

Question—What's the primary key?

Answer—Composite: Emp\_ID, Course\_Title

# ANOMALIES IN THIS TABLE

- **Insertion**—can't enter a new course without having an employee taking it
- **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

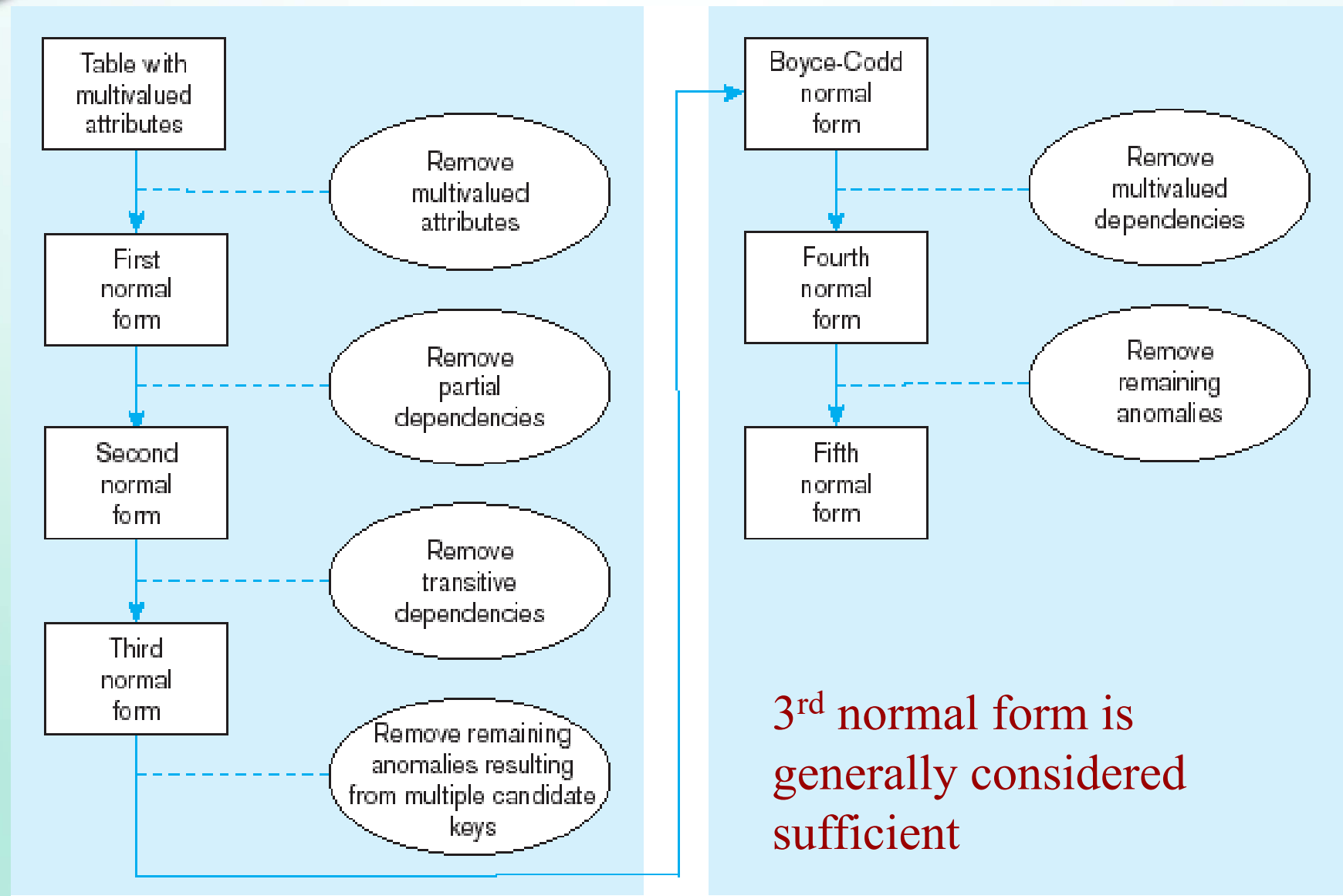
Why do these anomalies exist?

Because there are two themes (entity types) in this one relation. This results in data duplication and an unnecessary dependency between the entities

# FUNCTIONAL DEPENDENCIES AND KEYS

- Functional Dependency: The value of one attribute (the ***determinant***) determines the value of another attribute
- Candidate Key:
  - A unique identifier. One of the candidate keys will become the primary key
    - E.g. perhaps there is both credit card number and HKID in a table...in this case both are candidate keys
  - Each non-key field is functionally dependent on every candidate key

# Steps in normalization



# FIRST NORMAL FORM

- No multivalued attributes
- Every attribute value is atomic
- ***All relations are in 1<sup>st</sup> Normal Form***

# Table with multivalued attributes, not in 1<sup>st</sup> normal form

**FIGURE 4-25** INVOICE data (Pine Valley Furniture Company)

<u>OrderID</u>	Order Date	Customer ID	Customer Name	Customer Address	<u>ProductID</u>	Product Description	Product Finish	Product StandardPrice	Ordered Quantity
1006	10/24/2010	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	10/25/2010	6	Furniture Gallery	Boulder, CO	11	4-Dr Dresser	Oak	500.00	4
					4	Entertainment Center	Natural Maple	650.00	3

Note: this is NOT a relation

Table with no multivalued attributes and unique rows, in 1<sup>st</sup> normal form

<u>OrderID</u>	Order Date	Customer ID	Customer Name	Customer Address	<u>ProductID</u>	Product Description	Product Finish	Product StandardPrice	Ordered Quantity
1006	10/24/2010	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
1006	10/24/2010	2	Value Furniture	Plano, TX	5	Writer's Desk	Cherry	325.00	2
1006	10/24/2010	2	Value Furniture	Plano, TX	4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2010	6	Furniture Gallery	Boulder, CO	11	4-Dr Dresser	Oak	500.00	4
1007	10/25/2010	6	Furniture Gallery	Boulder, CO	4	Entertainment Center	Natural Maple	650.00	3

Note: this is relation, but not a well-structured one



# ANOMALIES IN THIS TABLE

- **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

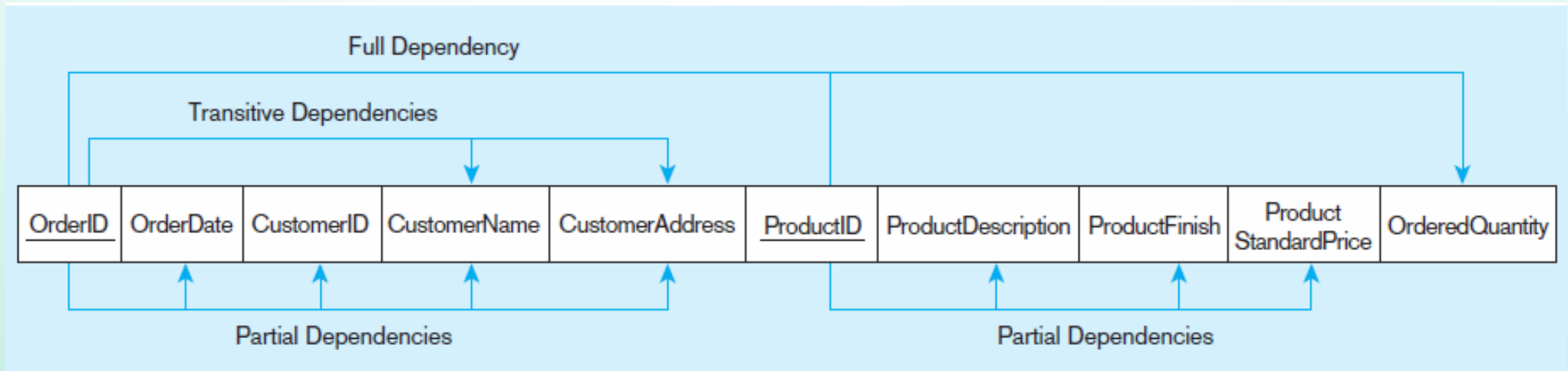
Why do these anomalies exist?

Because there are multiple themes (entity types) in one relation. This results in duplication and an unnecessary dependency between the entities

# SECOND NORMAL FORM

- 1NF PLUS ***every non-key attribute is fully functionally dependent on the ENTIRE primary key***
  - Every non-key attribute must be defined by the entire key, not by only part of the key
  - No partial functional dependencies

# Functional dependency diagram for INVOICE



**OrderID → OrderDate, CustomerID, CustomerName, CustomerAddress**

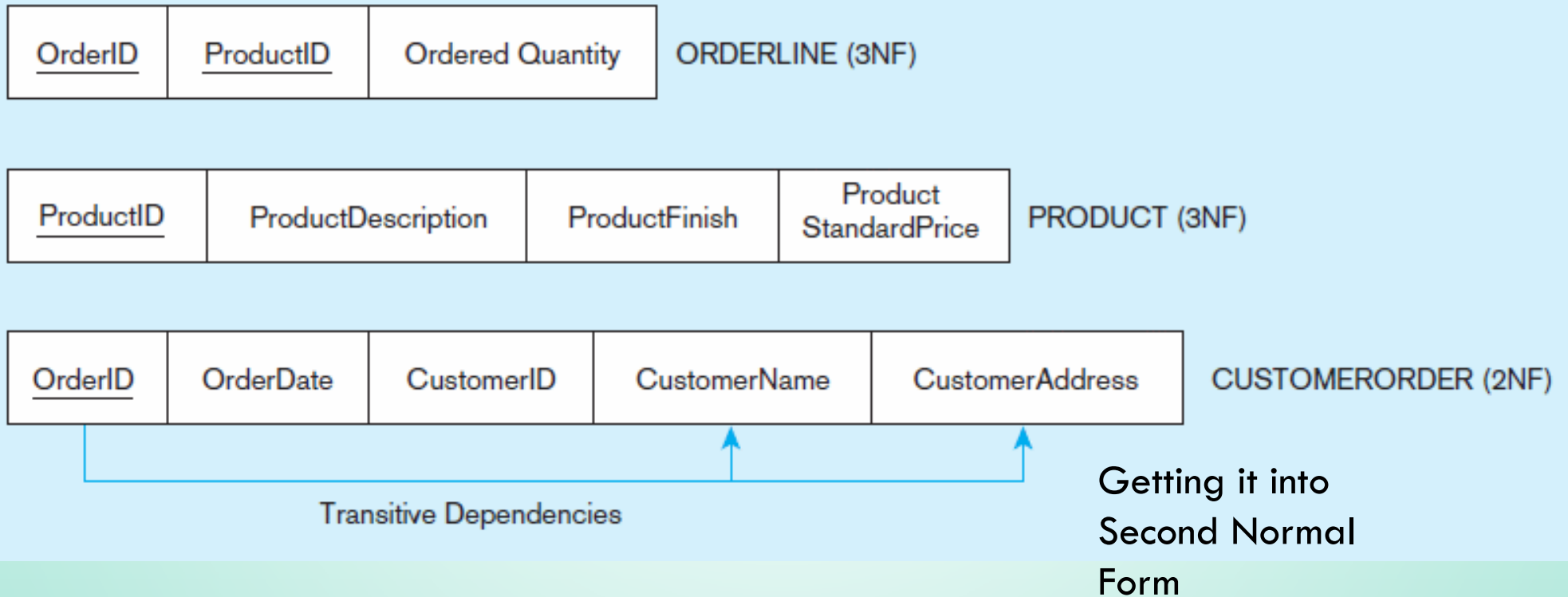
**CustomerID → CustomerName, CustomerAddress**

**ProductID → ProductDescription, ProductFinish, ProductStandardPrice**

**OrderID, ProductID → OrderQuantity**

**Therefore, NOT in 2<sup>nd</sup> Normal Form**

# Removing partial dependencies

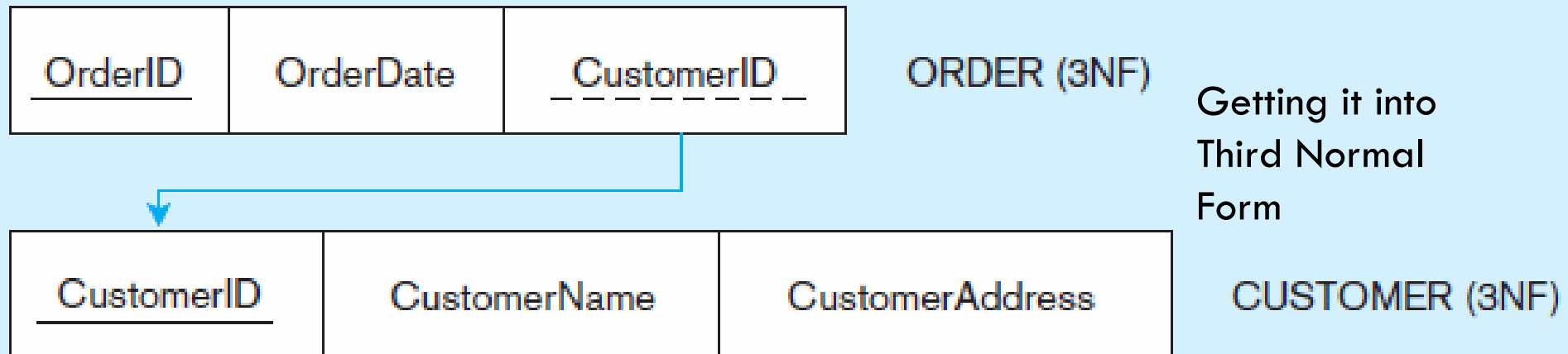


Partial dependencies are removed, but there are still transitive dependencies

# THIRD NORMAL FORM

- 2NF PLUS ***no transitive dependencies*** (functional dependencies on non-primary-key attributes)
- Note: This is called transitive, because the primary key is a determinant for another attribute, which in turn is a determinant for a third
- Solution: Non-key determinant with transitive dependencies go into a new table; non-key determinant becomes primary key in the new table and stays as foreign key in the old table

## Removing transitive dependencies



Transitive dependencies are removed

# MERGING RELATIONS

- View Integration—Combining entities from multiple ER models into common relations
- Issues to watch out for when merging entities from different ER models:
  - Synonyms—two or more attributes with different names but same meaning
  - Homonyms—attributes with same name but different meanings
  - Transitive dependencies—even if relations are in 3NF prior to merging, they may not be after merging
  - Supertype/subtype relationships—may be hidden prior to merging

# ENTERPRISE KEYS

- Primary keys that are unique in the whole database, not just within a single relation
- Corresponds with the concept of an object ID in object-oriented systems



# Enterprise keys

OBJECT (OID, ObjectType)  
EMPLOYEE (OID, EmpID, EmpName, DeptName, Salary)  
CUSTOMER (OID, CustID, CustName, Address)

a) Relations with  
enterprise key

b) Sample data  
with enterprise  
key

OBJECT

<u>OID</u>	ObjectType
1	EMPLOYEE
2	CUSTOMER
3	CUSTOMER
4	EMPLOYEE
5	EMPLOYEE
6	CUSTOMER
7	CUSTOMER

EMPLOYEE

<u>OID</u>	EmpID	EmpName	DeptName	Salary
1	100	Jennings, Fred	Marketing	50000
4	101	Hopkins, Dan	Purchasing	45000
5	102	Huber, Ike	Accounting	45000

CUSTOMER

<u>OID</u>	CustID	CustName	Address
2	100	Fred's Warehouse	Greensboro, NC
3	101	Bargain Bonanza	Moscow, ID
6	102	Jasper's	Tallahassee, FL
7	103	Desks 'R Us	Kettering, OH