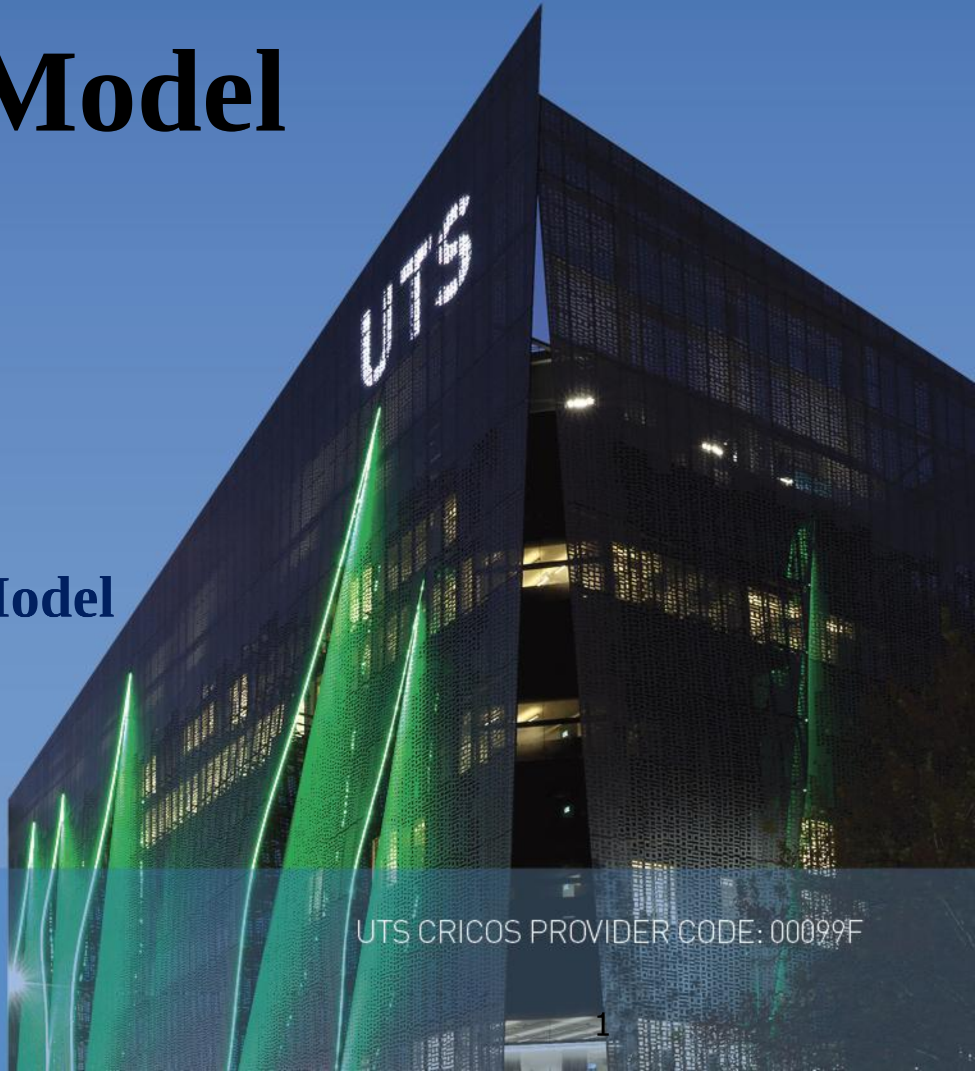
UTS:ENGINEERING AND
INFORMATION TECHNOLOGY

UTS
UNIVERSITY OF TECHNOLOGY, SYDNEY

lecture 4: Relational Model



Modern Database Management

11th Edition, International Edition

Chapter 4: Logical Database Design and the Relational Model

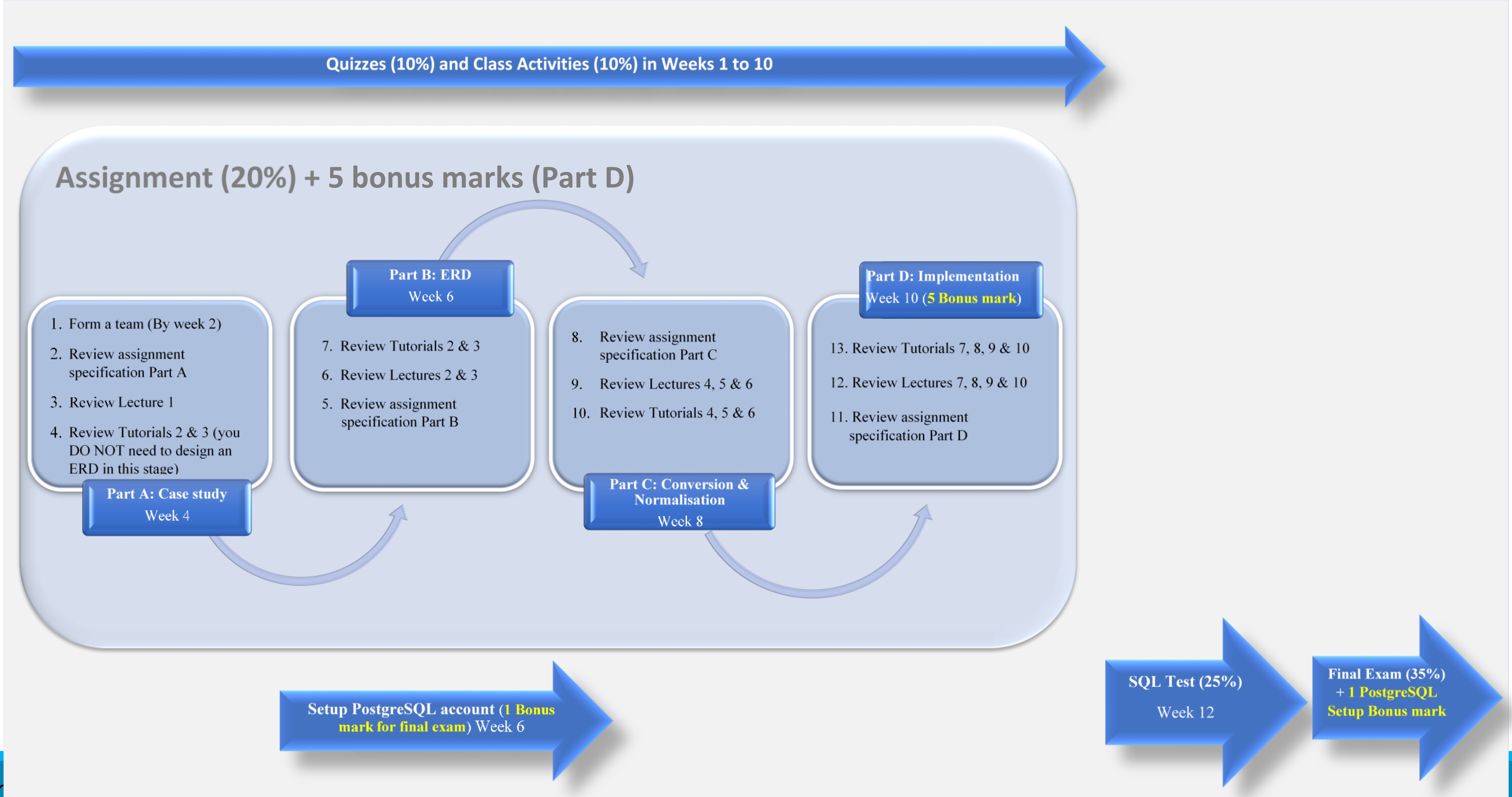
Jeffrey A. Hoffer, V. Ramesh, Heikki Topi

Participations and Discussions

If you have any question and you don’t want to share it now, send it to us via UTSOnline/Discussion Board.

However, it is better to speak out ☺

Assessment Chart and Knowledge Guideline



Quizzes (10%) and Class Activities (10%) in Weeks 1 to 10

Assignment (20%) + 5 bonus marks (Part D)

1. Form a team (By week 2)
2. Review assignment specification Part A
3. Review Lecture 1
4. Review Tutorials 2 & 3 (you DO NOT need to design an ERD in this stage)

Part A: Case study
Week 4

Part B: ERD
Week 6

7. Review Tutorials 2 & 3
6. Review Lectures 2 & 3
5. Review assignment specification Part B

8. Review assignment specification Part C
9. Review Lectures 4, 5 & 6
10. Review Tutorials 4, 5 & 6

Part C: Conversion & Normalisation
Week 8

Part D: Implementation
Week 10 (5 Bonus mark)

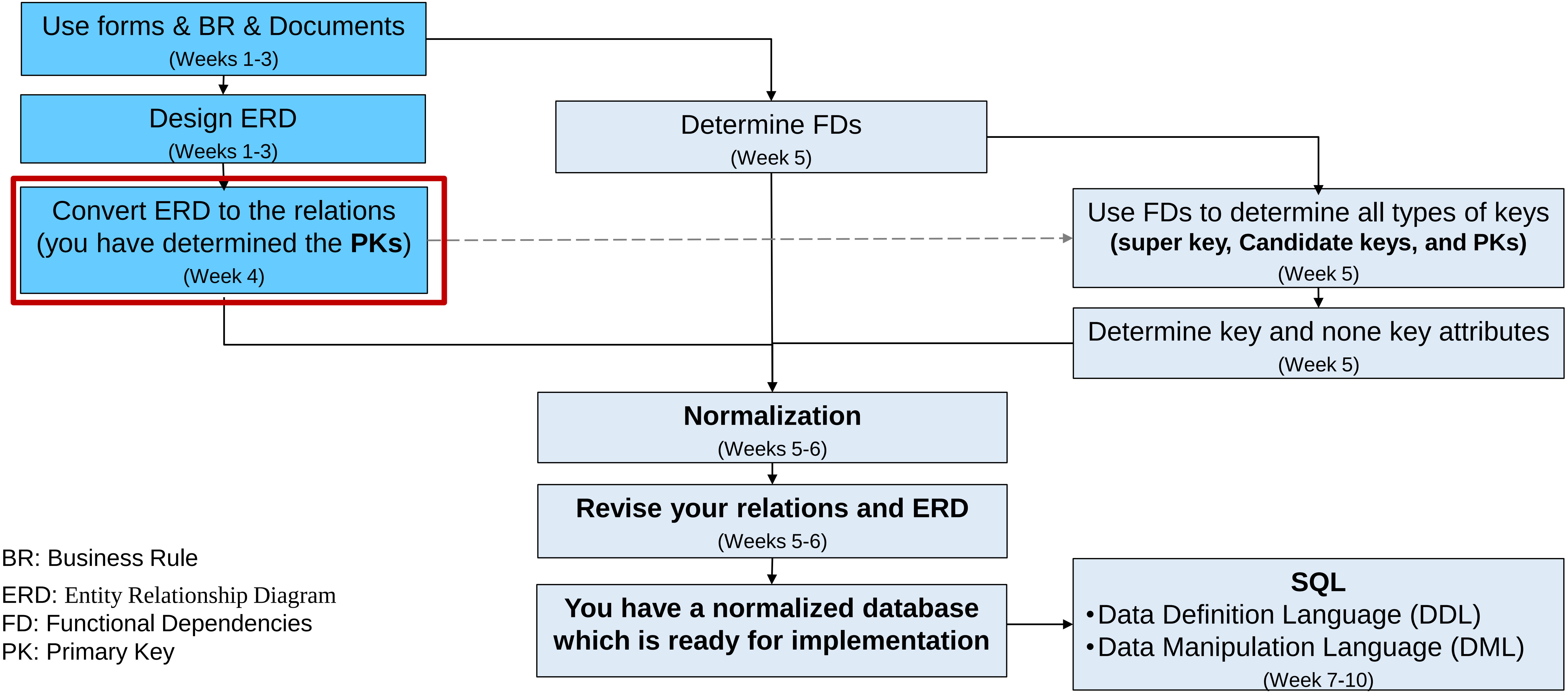
13. Review Tutorials 7, 8, 9 & 10
12. Review Lectures 7, 8, 9 & 10
11. Review assignment specification Part D

Setup PostgreSQL account (1 Bonus mark for final exam) Week 6

SQL Test (25%)
Week 12

Final Exam (35%)
+ 1 PostgreSQL Setup Bonus mark

Subject Flowchart



Use forms & BR & Documents
(Weeks 1-3)

Design ERD
(Weeks 1-3)

Convert ERD to the relations (you have determined the **PKs**)
(Week 4)

Determine FDs
(Week 5)

Use FDs to determine all types of keys **(super key, Candidate keys, and PKs)**
(Week 5)

Determine key and none key attributes
(Week 5)

Normalization
(Weeks 5-6)

Revise your relations and ERD
(Weeks 5-6)

You have a normalized database which is ready for implementation

SQL
- Data Definition Language (DDL)
- Data Manipulation Language (DML)

(Week 7-10)

BR: Business Rule

ERD: Entity Relationship Diagram
FD: Functional Dependencies
PK: Primary Key

DF Learning Plan

Description: we will have collaborative lecture at the beginning of the class. You need to do some tasks during the lecture as part of your class activities. Then you will do a quiz of what you have learned, then the tutorial will start. you will work in groups during the class.

Please be aware that the lecture slides with Blue title are designed for your self study.

Workshop Timetable:

Activity	Duration	Comments
Lecture	1 hour and 30 minutes	You will have 3 tasks to complete that need to take 20 minutes in total
Rest	10 minutes	Have fun
Review	10 minutes	Please review the review questions and ask your questions if you have any
Tutorial	1 hour	Have even more fun :D (you have two tasks, and need to take be completed in 40 minutes plus 20 minutes for tutors to provide you the solution)
Quiz (Open Book)	5 minutes	On today’s content. Will be run before or after the tutorial. Do your best ;)
Leave the class	5 minutes	Don’t forget to review what you have learn in this class, and check the information that is provided on **UTSOnline/Learning Material/Week 4**

Subject Overview

- **Design Entity Relationship Diagram (ERD)**
  - Week 1: Data Modelling I (Conceptual Level)
  - Week 2: Data Modelling II (Conceptual Level)
  - Week 3: Data Modelling III (Conceptual Level)
  - Week 4: Convert ERD to Relations (Logical Level)
  - Week 5: Functional Dependencies
  - Week 5: Normalization I
  - Week 6: Normalization II
- **Data manipulation**
  - Week 7: Simple Query
  - Week 8: Multiple Table Queries
  - Week 9: Subquery
  - Week 10: Correlated Subquery

Objectives

1. **Components of relational model**
2. **Relations**

    2.1. Correspondence with E-R Model

    2.2. Key Fields

    2.3. Integrity Constraints

        2.3.1. Domain Constraints

        2.3.2. Entity Integrity

        2.3.3. Referential Integrity

3. **Transforming EER Diagrams into Relations**

    3.1. Mapping Regular Entities to Relations (with simple, composite, and multivalued attributes)

    3.2. Mapping Weak Entities

    3.3. Mapping Binary Relationships (1:M, M:N, 1:1)

    3.4. Mapping Associative Entities

    3.5. Mapping Unary Relationships

    3.6. Mapping Ternary (and n-ary) Relationships

    3.7. Mapping Supertype/Subtype Relationships

1. Components of Relational Model

- **Data structure**

  Tables (relations), rows, columns

- **Data manipulation**

  Powerful SQL operations for retrieving and modifying data

- **Data integrity**

  Mechanisms for implementing business rules that maintain integrity of manipulated data

2. Relation

2. Relation

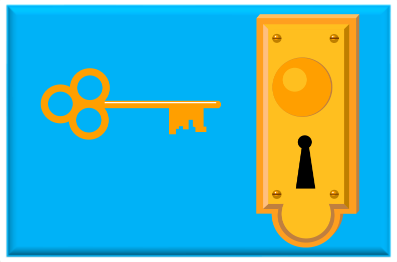
- A relation is a named, two-dimensional **table** of data.
- A table consists of rows (records) and columns (attribute or field).
- Requirements for a table to qualify as a relation:
  - It must have a **unique name**.
  - Every attribute value must be **atomic** (not multivalued, not composite) *(More on this in the next lectures)*.
  - 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.

2.1. 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**.

NOTE: The word ***relation*** (in relational database) is NOT the same as the word ***relationship*** (in E-R model).

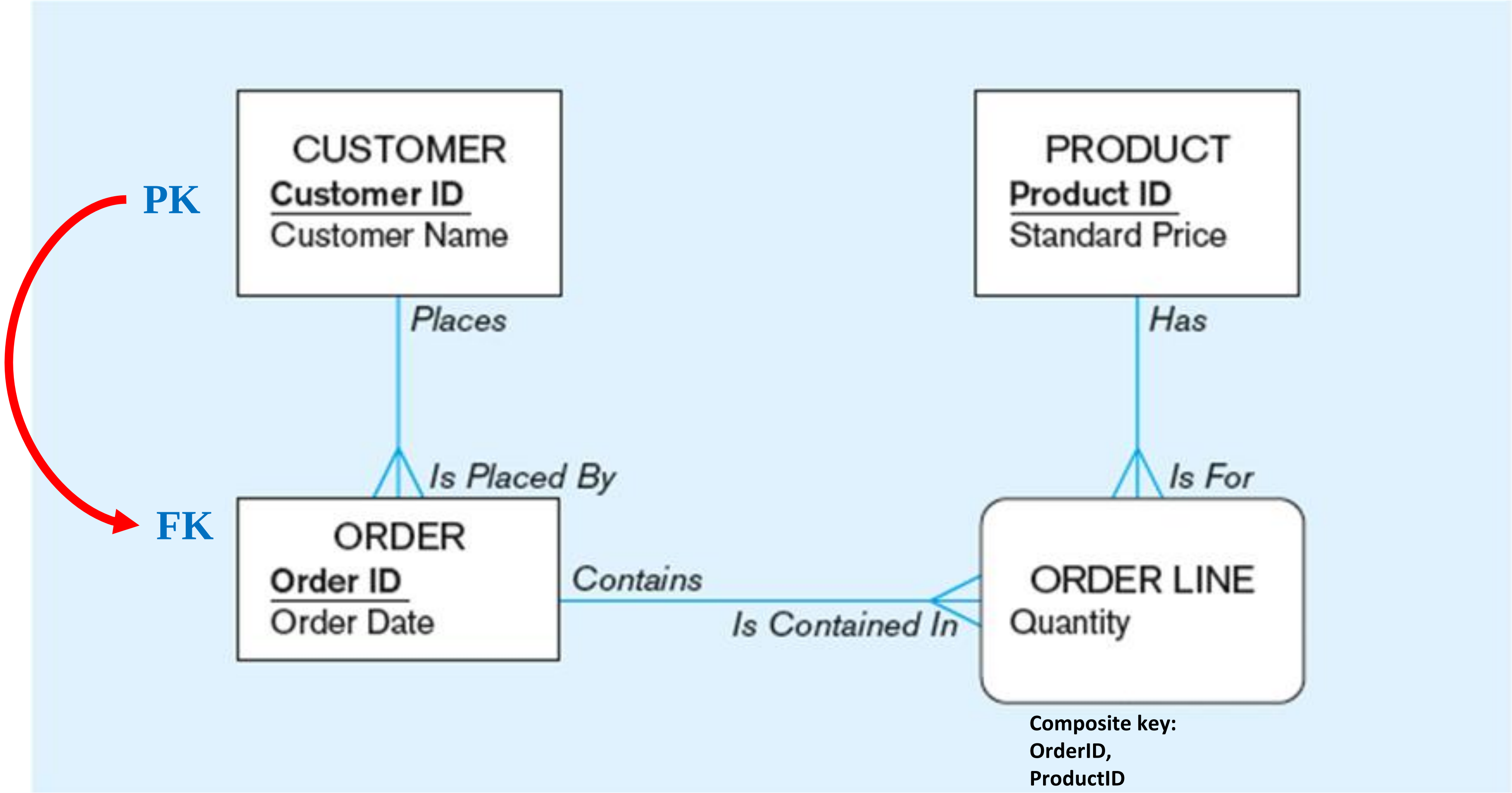
2.2. Key Fields



- Keys are special fields that serve two main purposes:
  - ***Primary keys*** are **unique** identifiers of the relation.

    •Examples include employee numbers, social security numbers, etc. This 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 *(more on this in Chapter 5).*

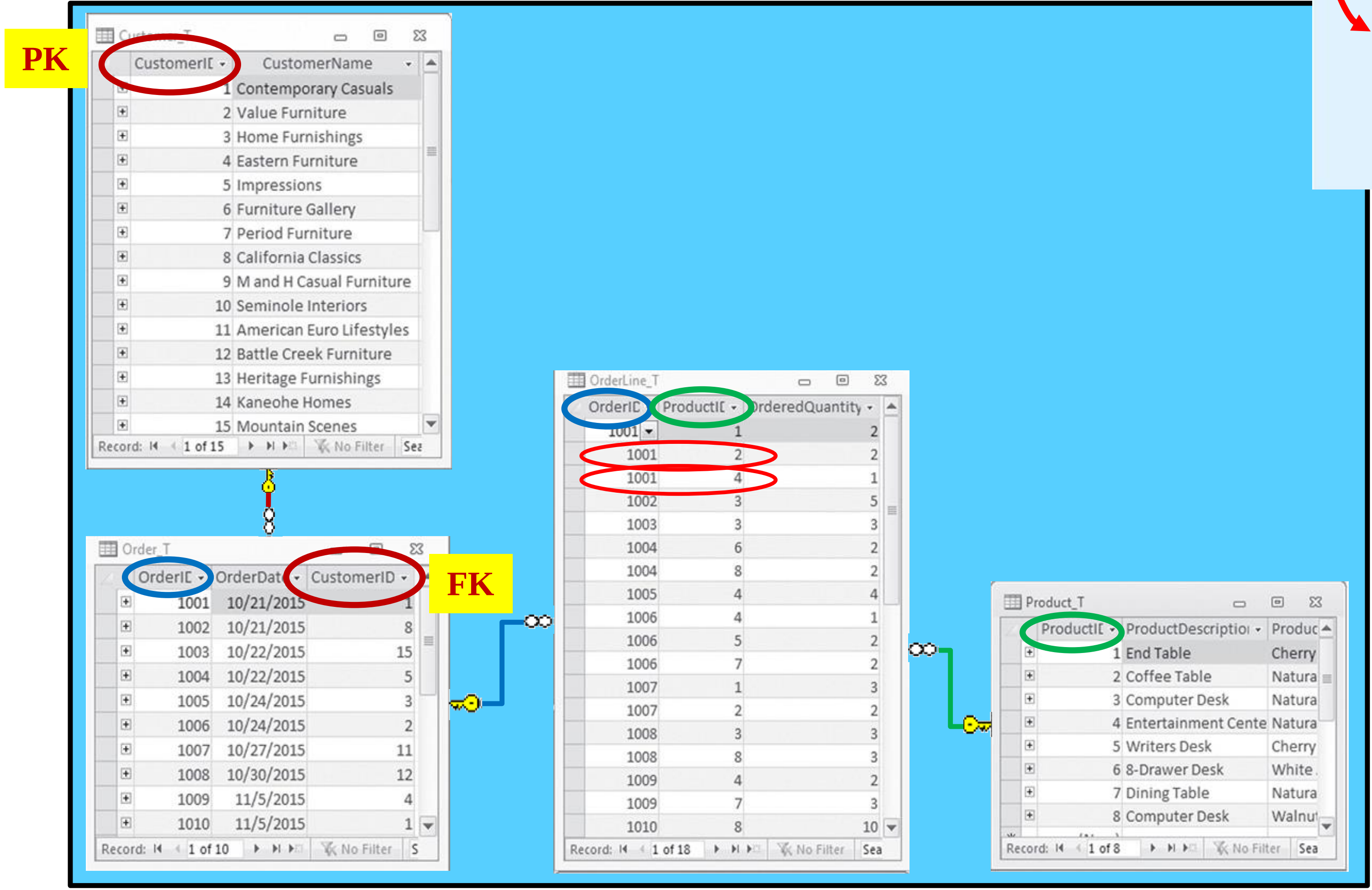
ERD of Pine Valley Furniture Company



Composite key:
OrderID,
ProductID

Primary key/Foreign key

Business Rule: any order can be related to many products. (related to OrderLine_T)



Customer_T

CustomerID	CustomerName
1	Contemporary Casuals
2	Value Furniture
3	Home Furnishings
4	Eastern Furniture
5	Impressions
6	Furniture Gallery
7	Period Furniture
8	California Classics
9	M and H Casual Furniture
10	Seminole Interiors
11	American Euro Lifestyles
12	Battle Creek Furniture
13	Heritage Furnishings
14	Kaneohe Homes
15	Mountain Scenes

Record: 1 of 15 No Filter

Order_T

OrderID	OrderDate	CustomerID
1001	10/21/2015	1
1002	10/21/2015	8
1003	10/22/2015	15
1004	10/22/2015	5
1005	10/24/2015	3
1006	10/24/2015	2
1007	10/27/2015	11
1008	10/30/2015	12
1009	11/5/2015	4
1010	11/5/2015	1

Record: 1 of 10 No Filter

OrderLine_T

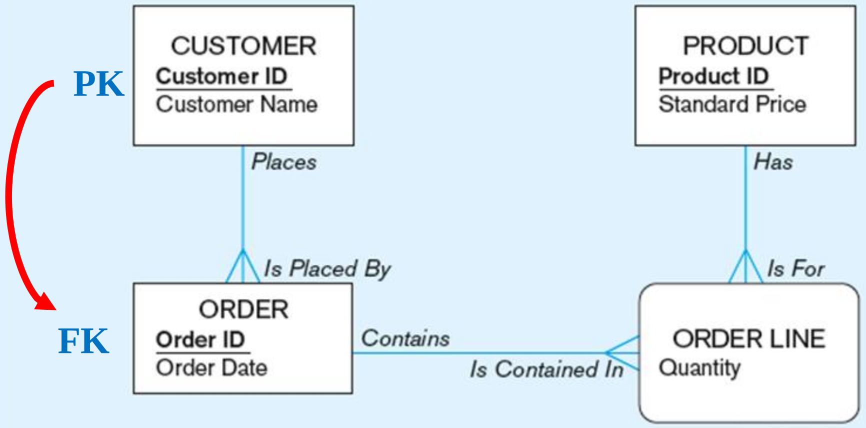
OrderID	ProductID	OrderedQuantity
1001	1	2
1001	2	2
1001	4	1
1002	3	5
1003	3	3
1004	6	2
1004	8	2
1005	4	4
1006	4	1
1006	5	2
1006	7	2
1007	1	3
1007	2	2
1008	3	3
1008	8	3
1009	4	2
1009	7	3
1010	8	10

Record: 1 of 18 No Filter

Product_T

ProductID	ProductDescription	Produc
1	End Table	Cherry
2	Coffee Table	Natura
3	Computer Desk	Natura
4	Entertainment Cente	Natura
5	Writers Desk	Cherry
6	8-Drawer Desk	White
7	Dining Table	Natura
8	Computer Desk	Walnut

Record: 1 of 8 No Filter



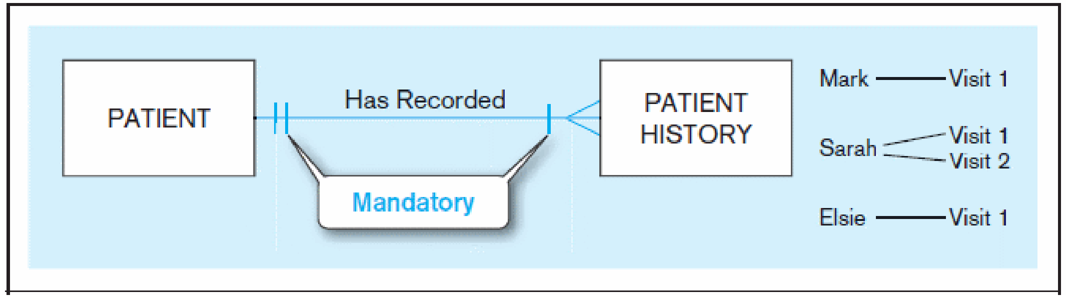
If two relations (tables) have a relationship, then the PK of the parent relation will be a FK in the dependent relation …

Question: What would be the FK in a relation (table) that need to have a relationship with OrderLine_T?

Class Activity 4.1 (5 minutes)

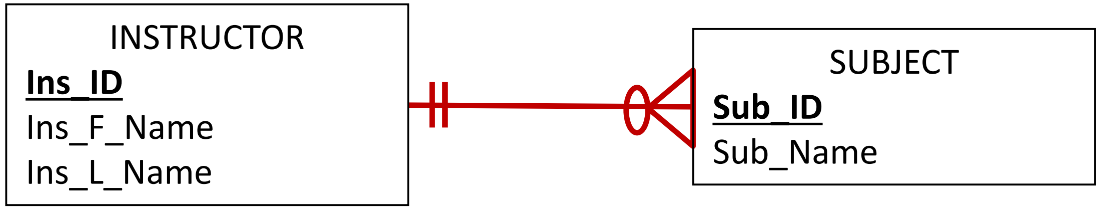
1. Explain why when there is one-to-many relationship between two entities, **PK** of the entity on the **one side** will be **FK** on the entity **on the many side**. Provide an example to explain this.

Solution to Class Activity 4.1:



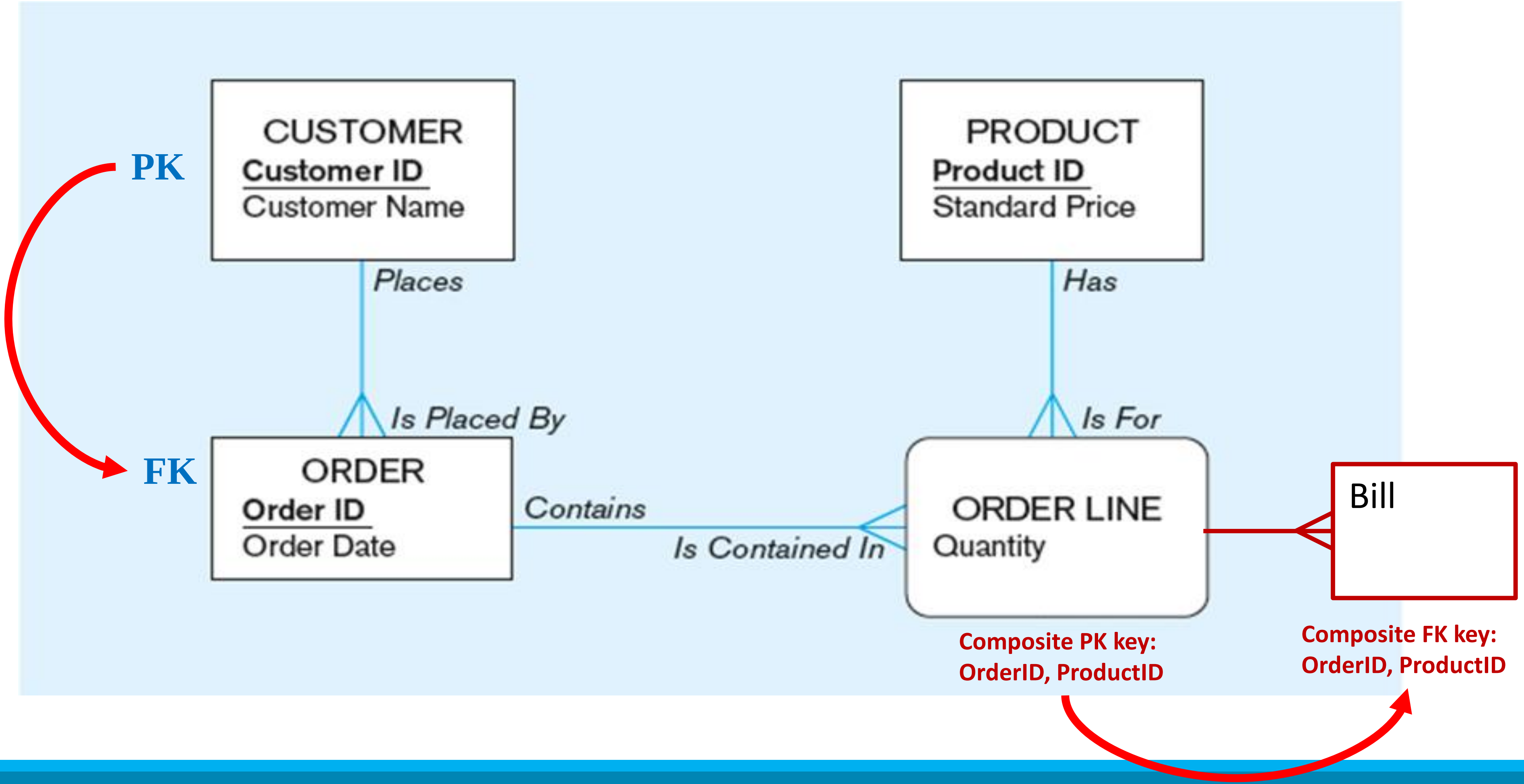
Discussion: Why in each one-to-many relationships, PK of the entity on one side is FK of the entity on the many side? (From Lecture 2)

BR: One instructor **can** teach **many** subjects, but one subject **needs to** be taught by **one** instructor.



ERD of Pine Valley Furniture Company

Question: What would be the FK in a relation (table) that need to have a relationship with OrderLine_T?



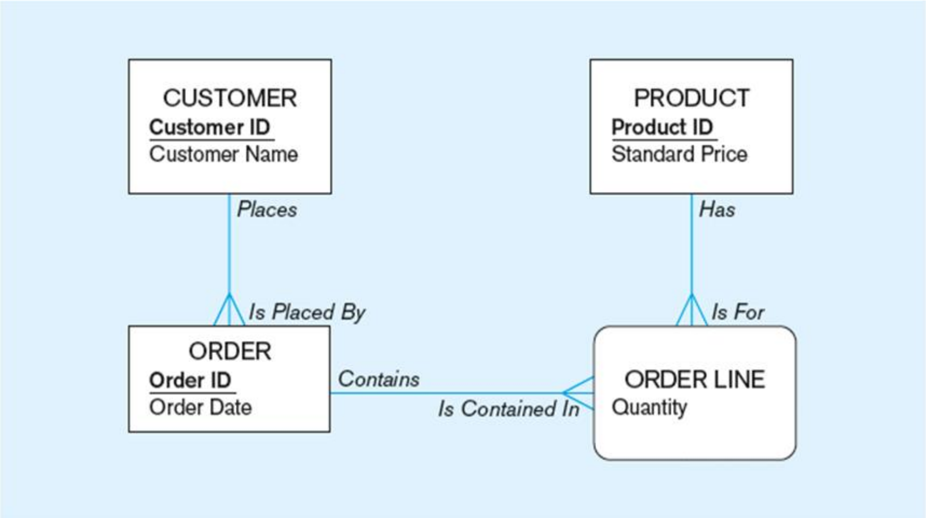
PK

FK

Composite PK key: OrderID, ProductID

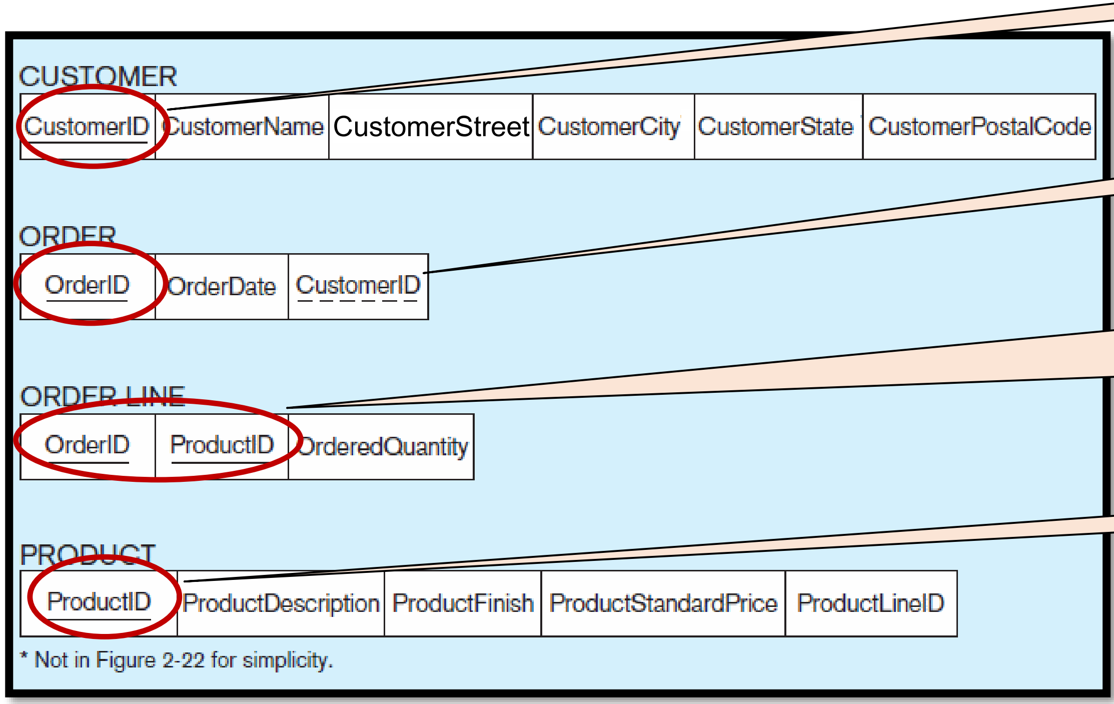
Composite FK key: OrderID, ProductID

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



ERD

Resulting Relations



CUSTOMER

CustomerID	CustomerName	CustomerStreet	CustomerCity	CustomerState	CustomerPostalCode

ORDER

OrderID	OrderDate	CustomerID

ORDER LINE

OrderID	ProductID	OrderedQuantity

PRODUCT

ProductID	ProductDescription	ProductFinish	ProductStandardPrice	ProductLineID

* Not in Figure 2-22 for simplicity.

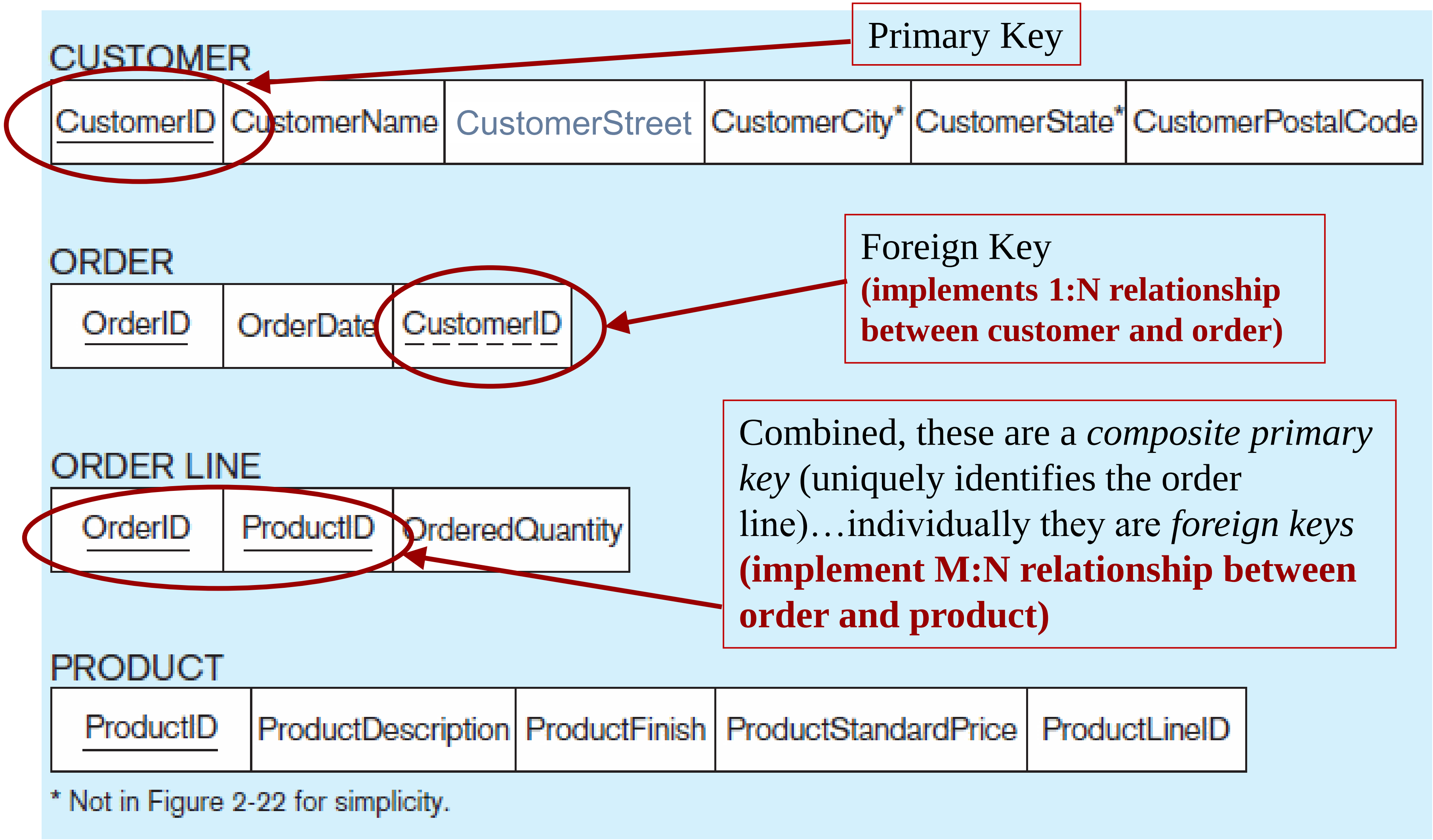
Primary key

Foreign key

Composite Primary key (Individually they are foreign keys)

Primary key

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



CUSTOMER

CustomerID	CustomerName	CustomerStreet	CustomerCity*	CustomerState*	CustomerPostalCode

ORDER

OrderID	OrderDate	CustomerID

ORDER LINE

OrderID	ProductID	OrderedQuantity

PRODUCT

ProductID	ProductDescription	ProductFinish	ProductStandardPrice	ProductLineID

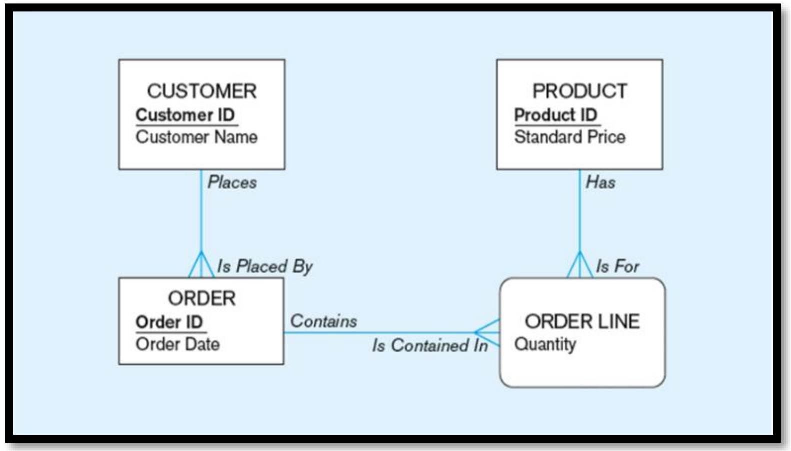
* Not in Figure 2-22 for simplicity.

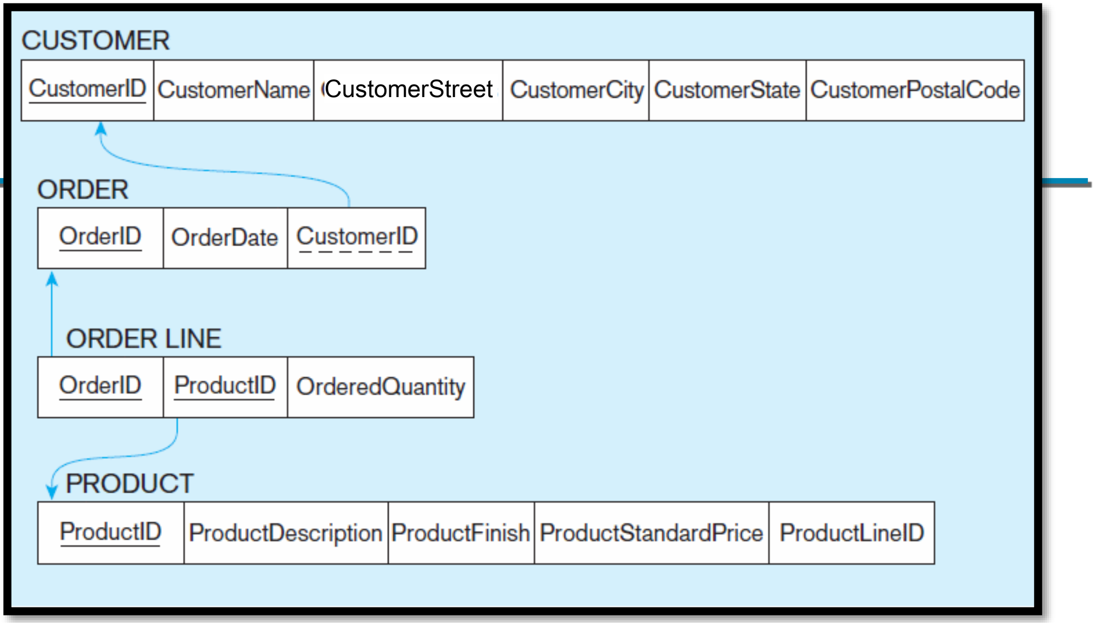
Primary Key

Foreign Key **(implements 1:N relationship between customer and order)**

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)**

Schema to relations Methods:





CUSTOMER (CustomerID, CustomerName, CustomerStreet, CustomerCity, CustomerState, CustomerPostalCode)

ORDER (OrderID, OrderDate, CustomerID*)

FK (CustomerID) references CUSTOMER

ORDERLINE (OrderID*, ProductID*, OrderQuantity)

FK (OrderID) references ORDER

FK (ProductID) references PRODUCT

PRODUCT (ProductID, ProductDescription, ProductFinish, ProductStandardPrice, ProductLineID)

Where we are … review the path ☺

- ❑ **We know about the relations**
- ❑ **We have abstract information about converting ERD to the Relations.**
- ❑ **Now we need to know which constraints need to be considered when we do the conversion and why.**

2.3. Integrity Constraints

Integrity Constraints are applied to facilitate maintaining the accuracy and integrity of data in the database. The major types of integrity constraint are:

2.3.1. Domain Constraints

- Allowable values for an attribute (See Table 4-1)

2.3.2. Entity Integrity

- No primary key attribute may be null. All primary key fields **MUST** have data.

2.3.3. Referential Integrity

- 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.

  Referential Integrity rule is used to maintain the consistency among rows between the two tables.

2.3.1. Domain Constraints

Domain Constraints Allowable values for an attribute (See Table 4-1)

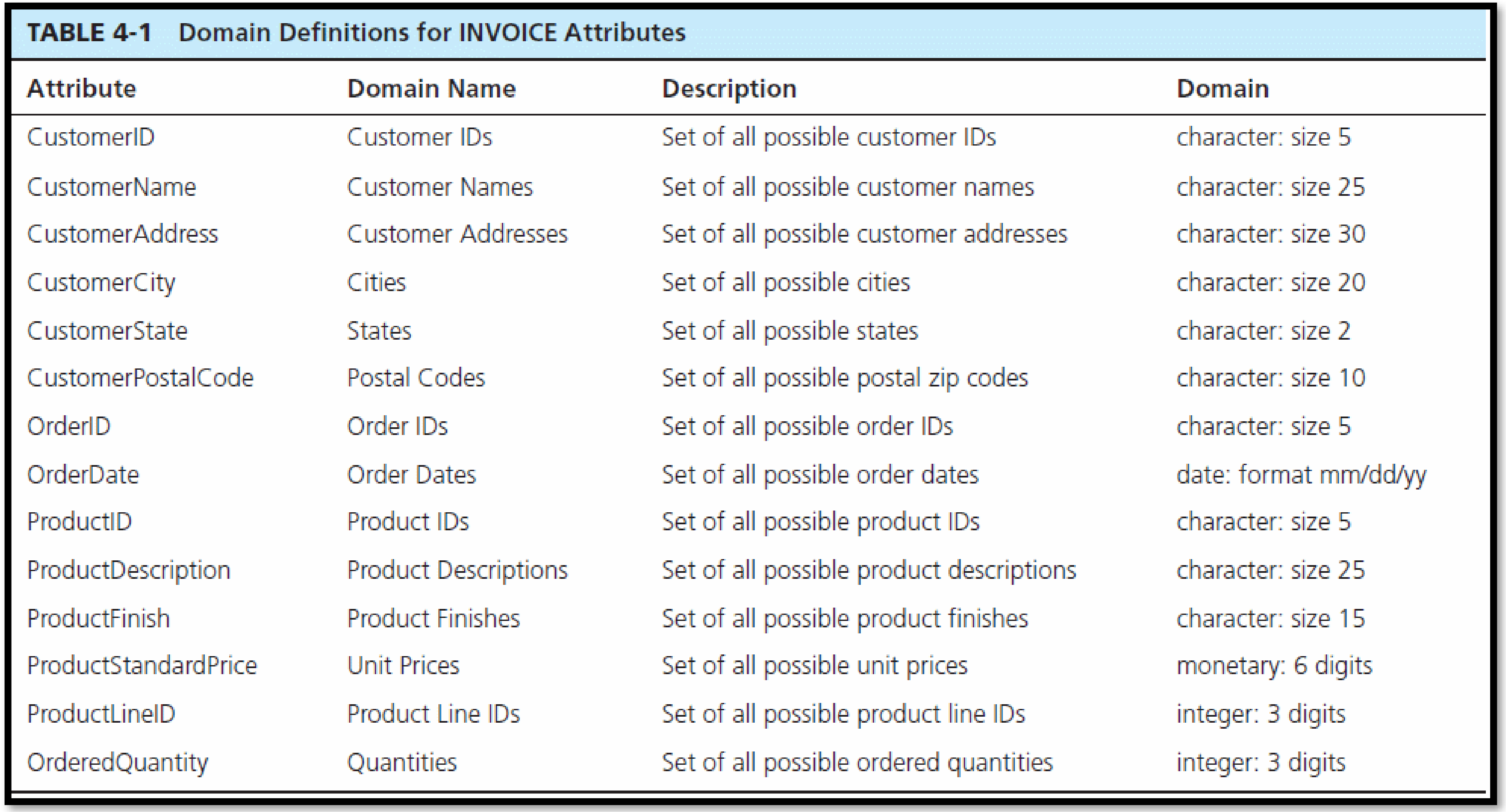


TABLE 4-1 Domain Definitions for INVOICE Attributes

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.

2.3.3. Referential Integrity

Referential Integrity 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.

Referential Integrity rule is used to maintain the consistency among rows between the two tables.

Example of integrity constraint: 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

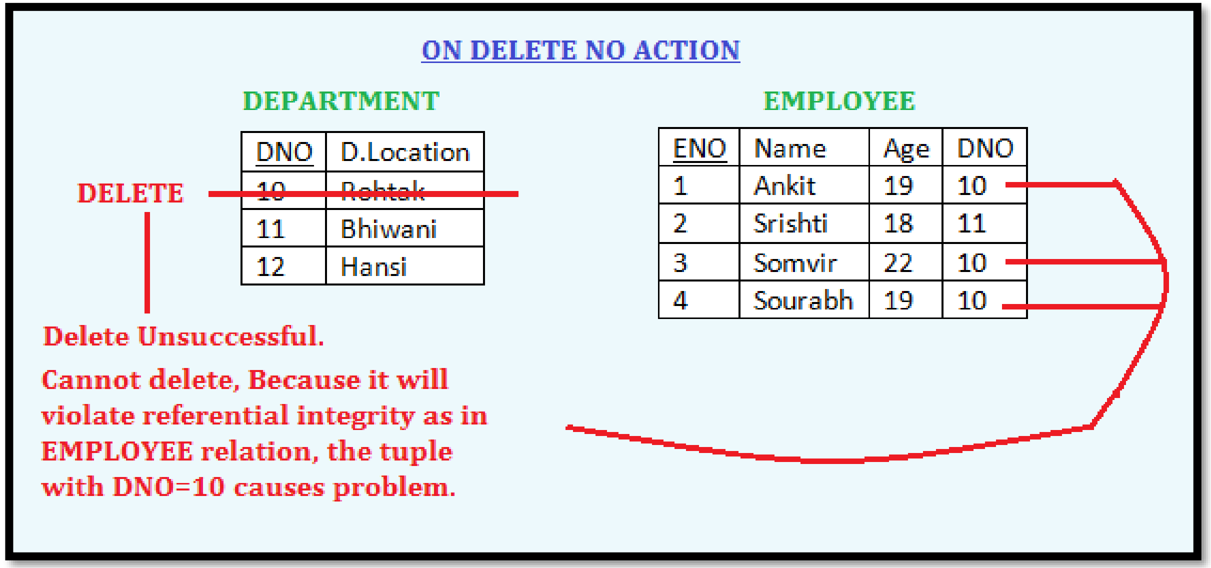
→ The foreign key can be null

→ **Set-to-Null** is not allowed for weak and associated entities

2.3.3. Referential Integrity: Restrict

Restrict: don’t allow delete of “parent” side if related rows exist in “dependent” side

- Delete of “parent” side



ON DELETE NO ACTION

DEPARTMENT

DNO	D.Location
~~10~~	~~Rohtak~~
11	Bhiwani
12	Hansi

EMPLOYEE

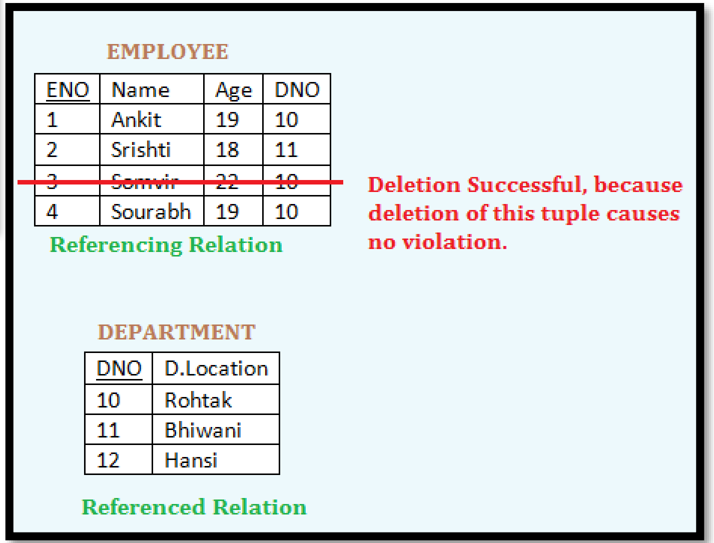
ENO	Name	Age	DNO
1	Ankit	19	10
2	Srishti	18	11
3	Somvir	22	10
4	Sourabh	19	10

DELETE

Delete Unsuccessful.

Cannot delete, Because it will violate referential integrity as in EMPLOYEE relation, the tuple with DNO=10 causes problem.

- Delete of “dependent” side



EMPLOYEE

ENO	Name	Age	DNO
1	Ankit	19	10
2	Srishti	18	11
~~3~~	~~Somvir~~	~~22~~	~~10~~
4	Sourabh	19	10

Referencing Relation

Deletion Successful, because deletion of this tuple causes no violation.

DEPARTMENT

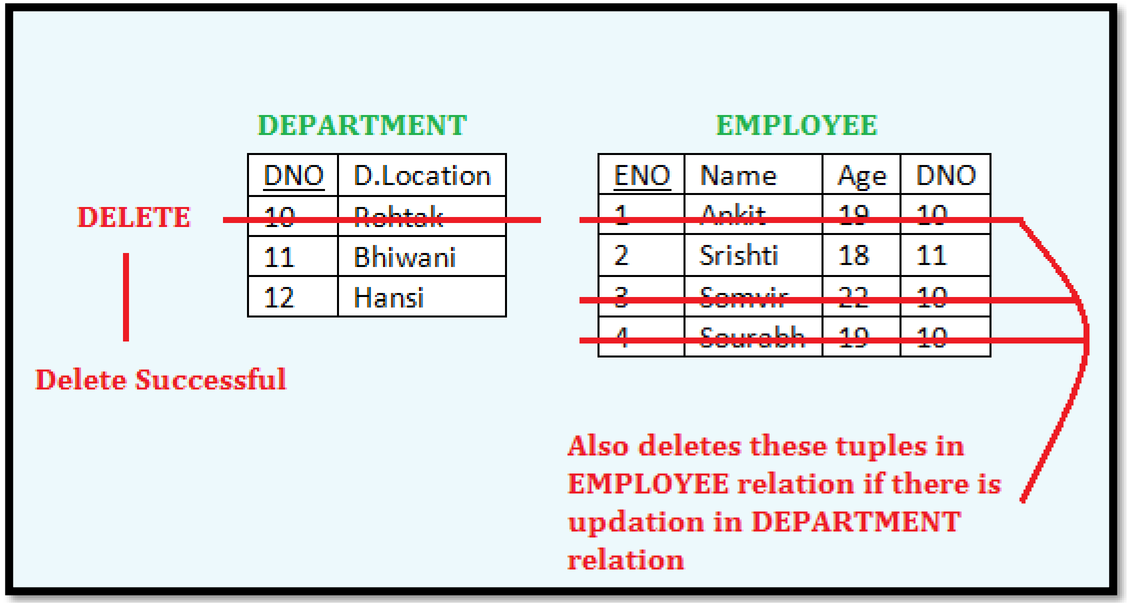
DNO	D.Location
10	Rohtak
11	Bhiwani
12	Hansi

Referenced Relation

Photo Reference: http://www.edugrabs.com

2.3.3. Referential Integrity: Cascade

Cascade–automatically delete “dependent” side rows that correspond with the “parent” side row to be deleted



DEPARTMENT

DNO	D.Location
~~10~~	~~Rohtak~~
11	Bhiwani
12	Hansi

EMPLOYEE

ENO	Name	Age	DNO
~~1~~	~~Ankit~~	~~19~~	~~10~~
2	Srishti	18	11
~~3~~	~~Somvir~~	~~22~~	~~10~~
~~4~~	~~Sourabh~~	~~19~~	~~10~~

Also deletes these tuples in EMPLOYEE relation if there is updation in DEPARTMENT relation

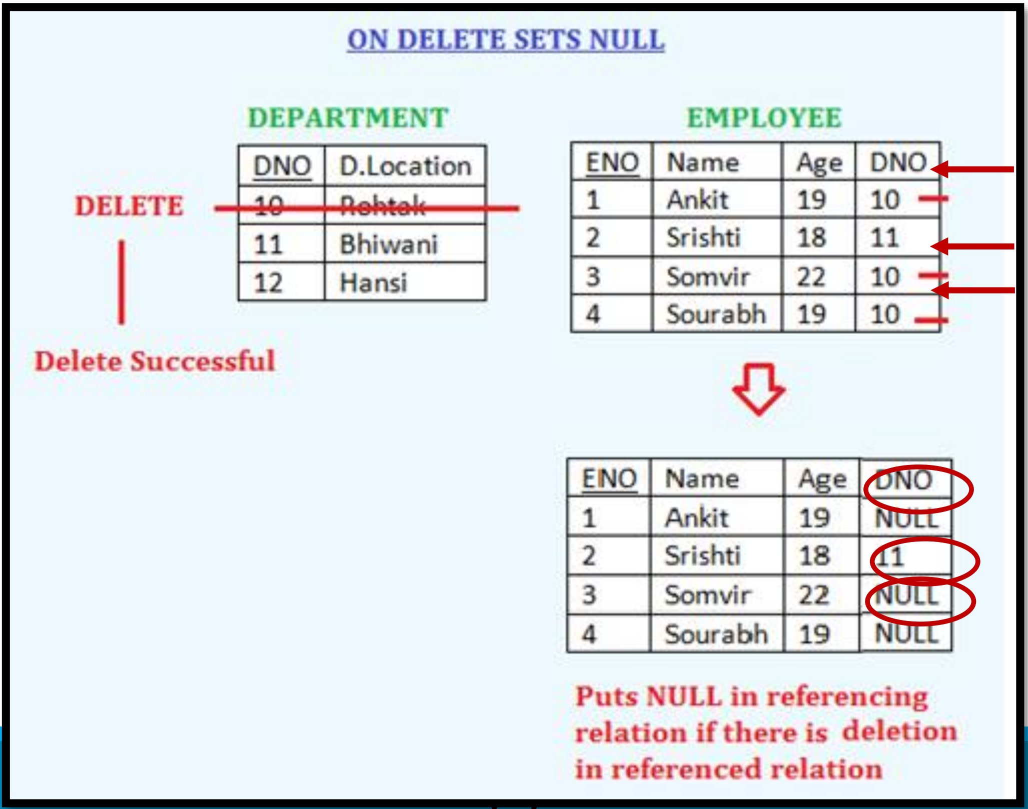
Photo Reference: http://www.edugrabs.com

2.3.3. Referential Integrity: Set-to-Null

Set-to-Null: set the foreign key in the dependent side to null if deleting from the parent side.

Notes:

- → The foreign key can be null
- → **Set-to-Null** is not allowed for weak and associated entities (where FK are part of the key).
- → **Set-to-Null** is not allowed when is related to a mandatory cardinality.



ON DELETE SETS NULL

DEPARTMENT

DNO	D.Location
~~10~~	~~Rohtak~~
11	Bhiwani
12	Hansi

DELETE

Delete Successful

EMPLOYEE

ENO	Name	Age	DNO
1	Ankit	19	10
2	Srishti	18	11
3	Somvir	22	10
4	Sourabh	19	10

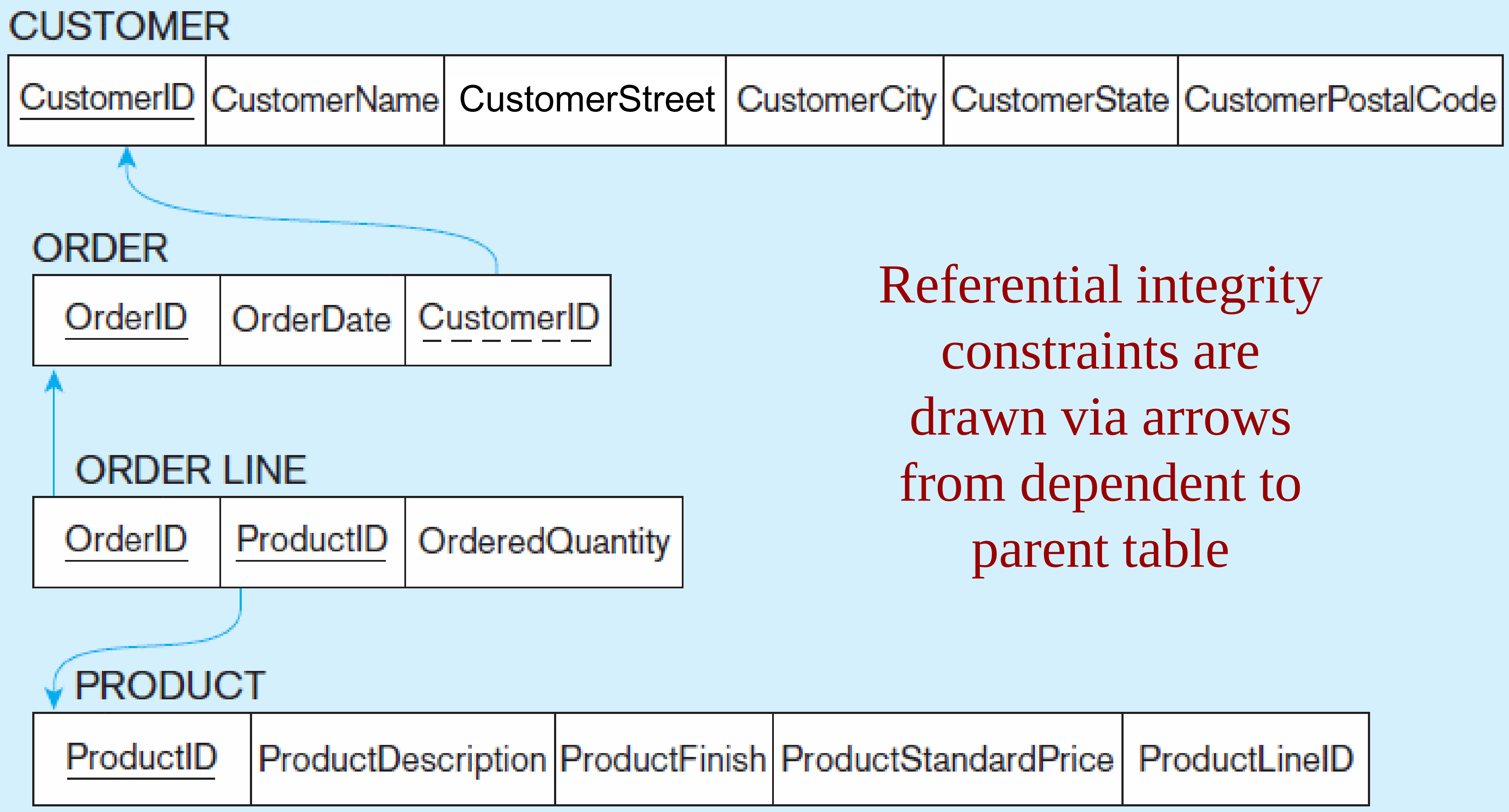
⇩

ENO	Name	Age	DNO
1	Ankit	19	NULL
2	Srishti	18	11
3	Somvir	22	NULL
4	Sourabh	19	NULL

Puts NULL in referencing relation if there is deletion in referenced relation

Photo Reference: http://www.edugrabs.com

Referential integrity constraints (Pine Valley Furniture)- (Figure 4-5)



CUSTOMER

CustomerID	CustomerName	CustomerStreet	CustomerCity	CustomerState	CustomerPostalCode

ORDER

OrderID	OrderDate	CustomerID

ORDER LINE

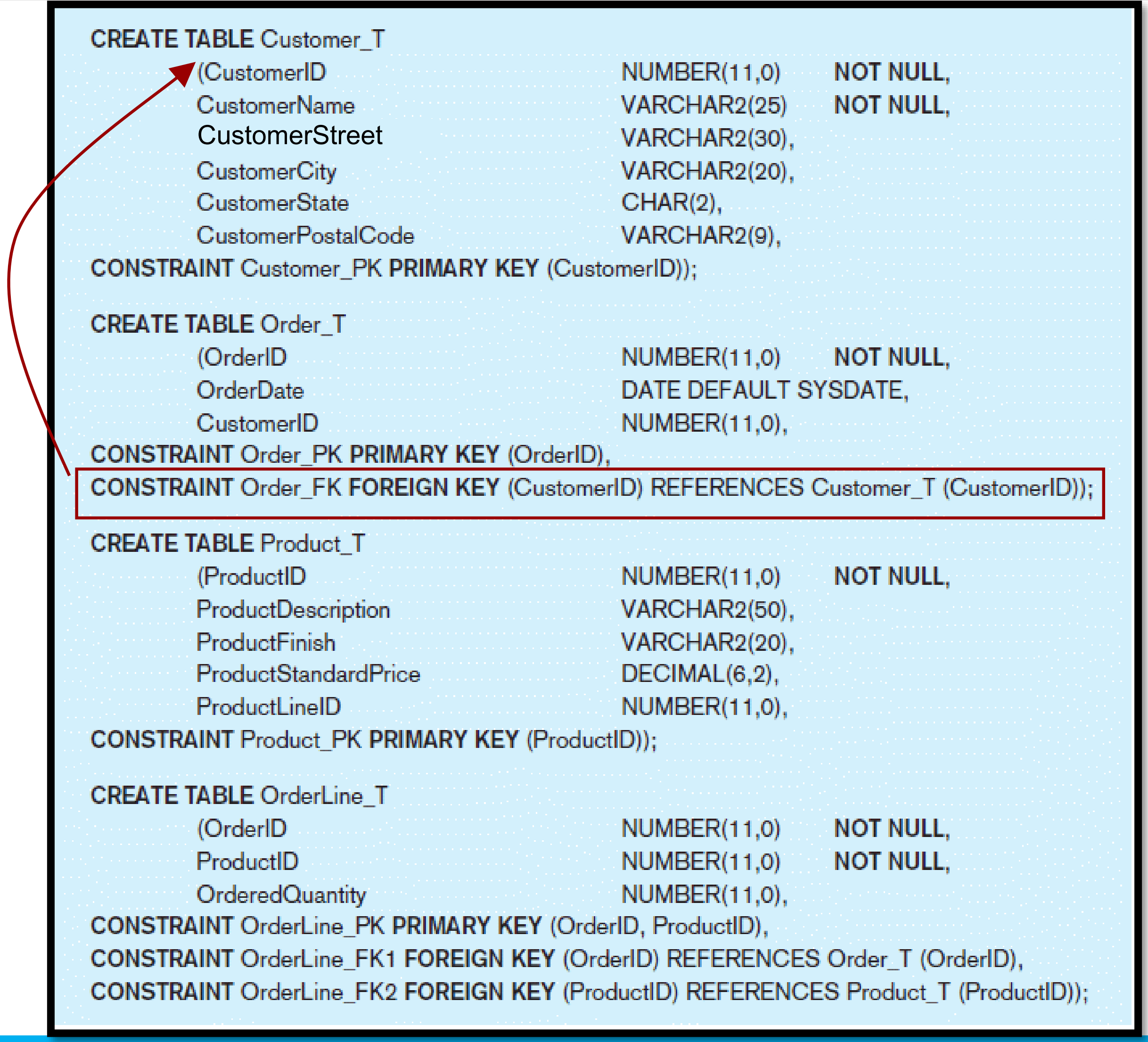
OrderID	ProductID	OrderedQuantity

PRODUCT

ProductID	ProductDescription	ProductFinish	ProductStandardPrice	ProductLineID

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

SQL table definitions (Figure 4-6)



```
CREATE TABLE Customer_T
        (CustomerID                  NUMBER(11,0)     NOT NULL,
        CustomerName                 VARCHAR2(25)     NOT NULL,
        CustomerStreet               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.

Note: Review this slide after Lecture 7 when you learn about DDL ☺

Where we are … review the path ☺

- ❑ **We know about the relations**
- ❑ **We have abstract information about converting ERD to the Relations.**
- ❑ **We know which constraints (integrity constraints) need to be considered when we do the conversion and why.**
- ❑ **Now we need to know how to convert different type of attributes and entities to the relations.**

3. Transforming ERD into Relations

3. Transforming ERD into Relations

3.1. Mapping Regular Entities to Relations

3.2. Mapping Weak Entities

3.3. Mapping Binary Relationships

3.4. Mapping Associative Entities

3.5. Mapping Unary Relationships

3.6. Mapping Ternary (and n-ary) Relationships

3.7. Mapping Supertype/Subtype Relationships

3.1. Mapping Regular Entities to Relations

3.1.1. Simple attributes: E-R attributes map directly onto the relation

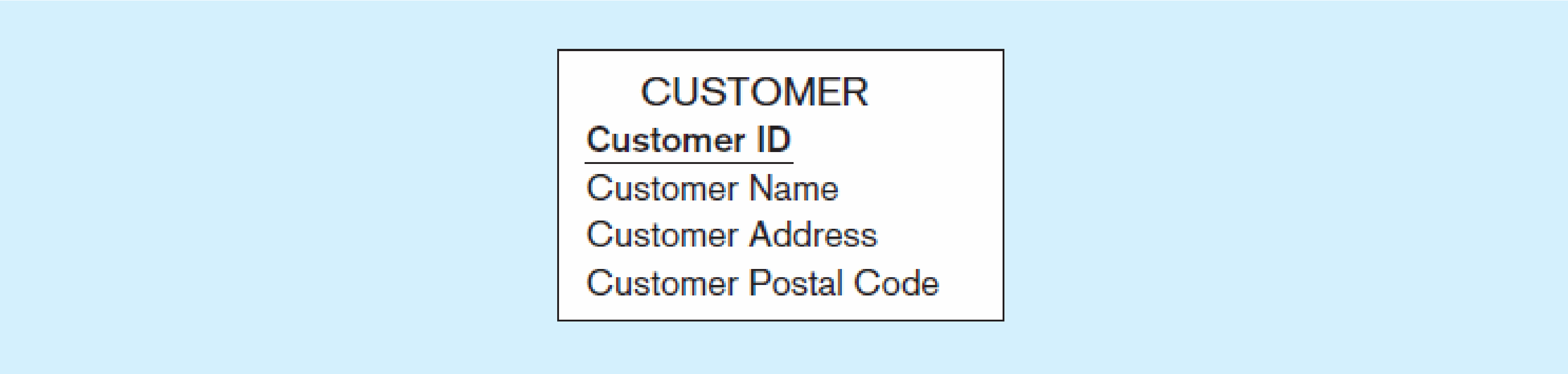
3.1.2. Composite attributes: Use only their simple component attributes

3.1.3. Multivalued Attribute: Becomes a separate relation with a foreign key taken from the superior entity

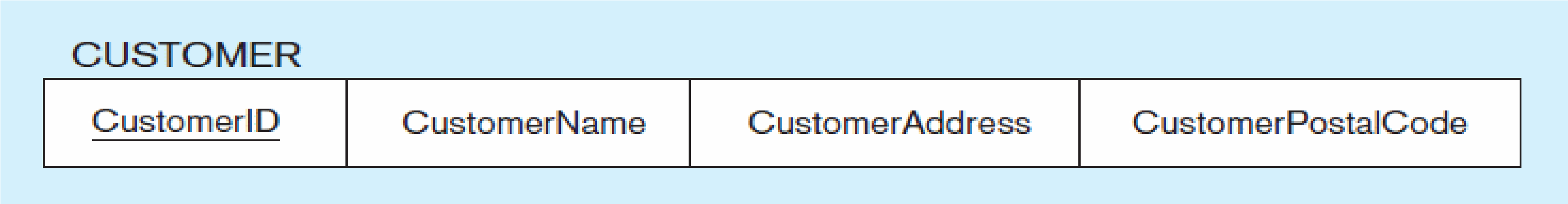
3.1.1. Mapping a regular entity (Figure 4-8) with simple attribute

3.1.1. Simple attributes: E-R attributes map directly onto the relation

(a) CUSTOMER entity type with simple attributes



(b) CUSTOMER relation



CUSTOMER

CustomerID	CustomerName	CustomerAddress	CustomerPostalCode

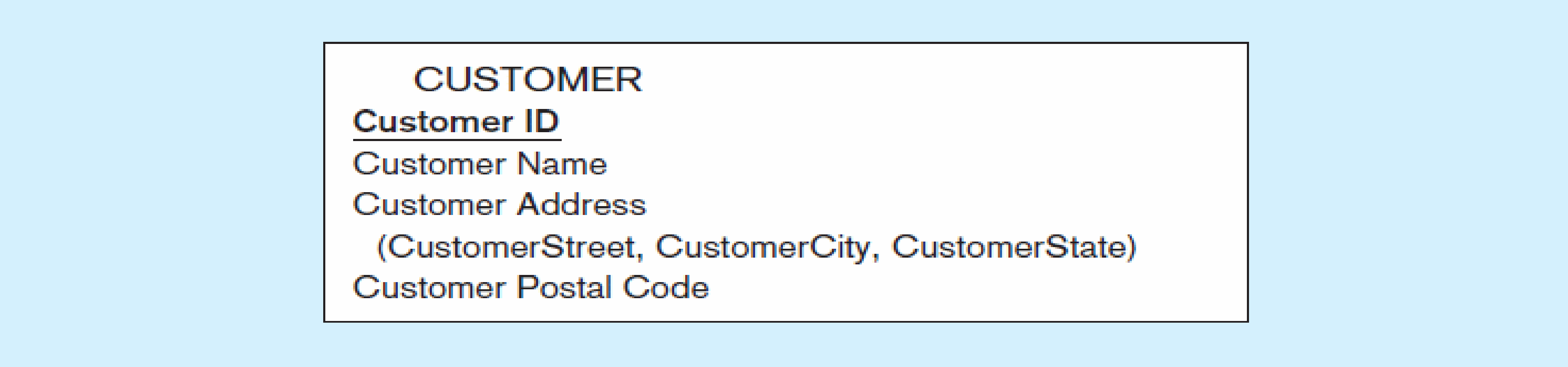
or

CUSTOMER (CustomerID, CustomerName, CustomerAddress, CustomerPostalCode)

3.1.2. Mapping a composite attribute (Figure 4-9)

3.1.2. Composite attributes: Use only their simple component attributes

(a) CUSTOMER entity type with composite attribute



CUSTOMER

Customer ID

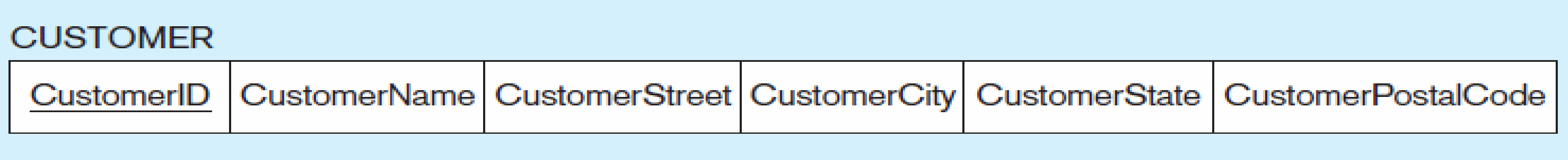
Customer Name

Customer Address

(CustomerStreet, CustomerCity, CustomerState)

Customer Postal Code

(b) CUSTOMER relation with address detail



CUSTOMER

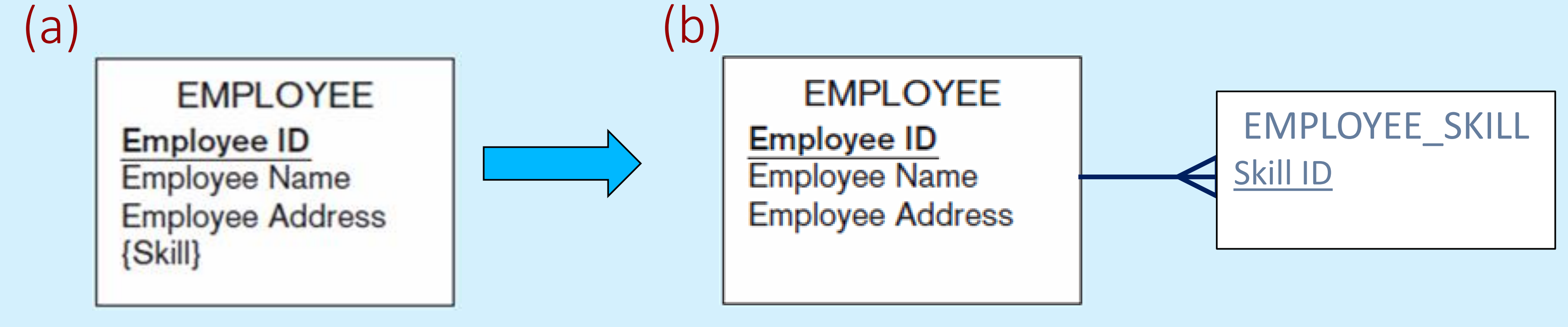
CustomerID	CustomerName	CustomerStreet	CustomerCity	CustomerState	CustomerPostalCode

or

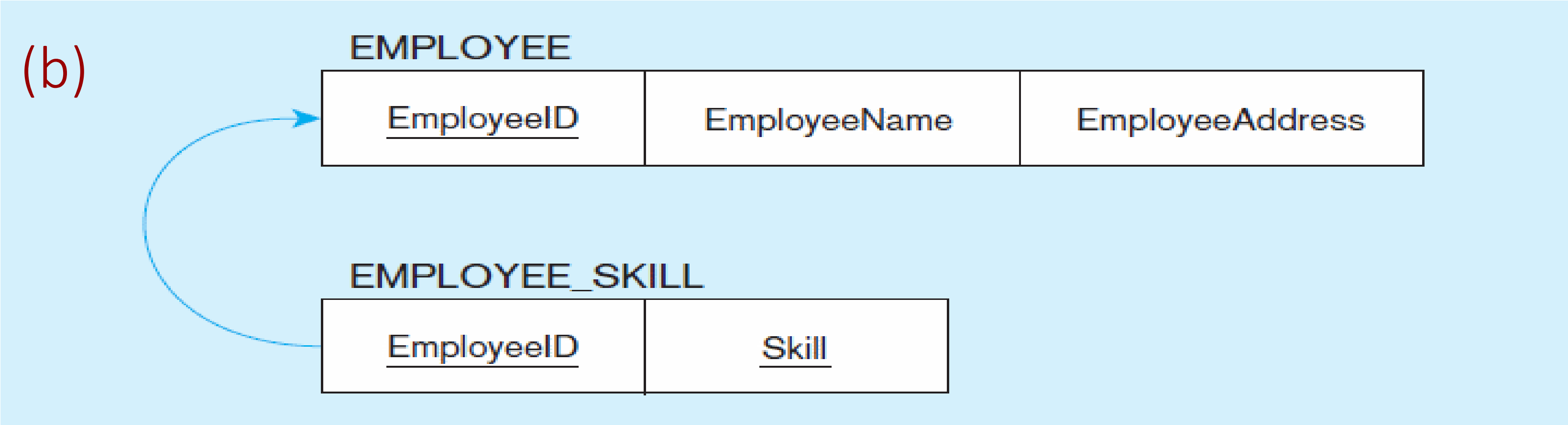
CUSTOMER (CustomerID, CustomerName, CustomerStreet, CustomerCity, CustomerState, CustomerPostalCode)

3.1.3. Mapping a multivalued attribute (Figure 4-10)

3.1.3. Multivalued Attribute: Becomes a separate relation with a foreign key taken from the superior entity.



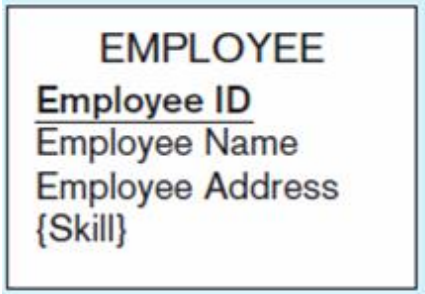
- Multivalued attribute becomes a separate relation with foreign key
- One–to–many relationship between original entity and new relation



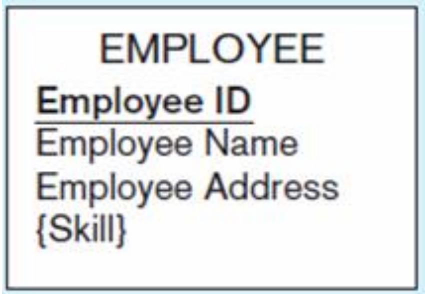
Class Activity 4.2 (10 minutes)

2. Redesign the following entity, where every employee need to have at least one skill and every skill can be chosen by any employee. The information about skills like skill ID and name need to be stored in the database.

   Explain why this need to be redesigned, and then convert your new ERD to the relations.



Solution to Class Activity 4.2:

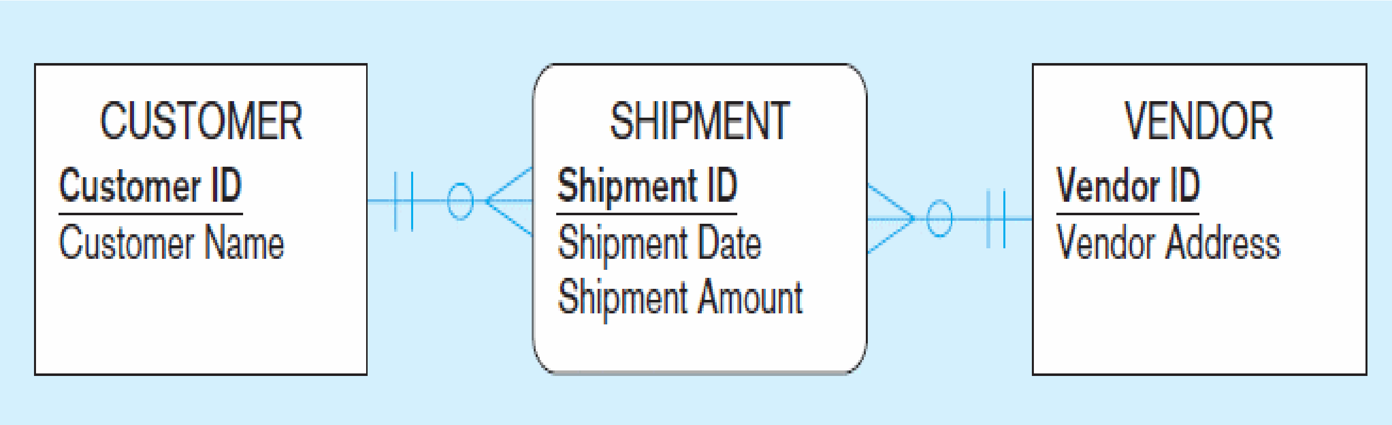


EMPLOYEE

Employee ID
Employee Name
Employee Address
{Skill}

Class Activity 4.3 (5 minutes)

3. Convert the associative entity in this ERD to a relation?



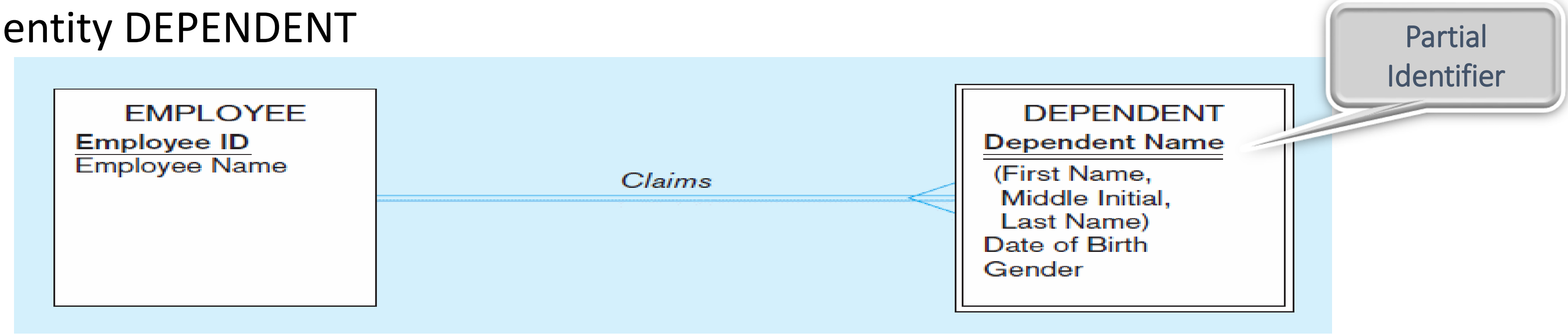
3.2. Mapping Weak Entities

- 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)

NOTE: **Foreign keys can have null values**, **but** the domain constraint for the foreign key <u>should NOT allow null</u> value if DEPENDENT is a weak entity or an associative entity, or is related to a mandatory cardinality.

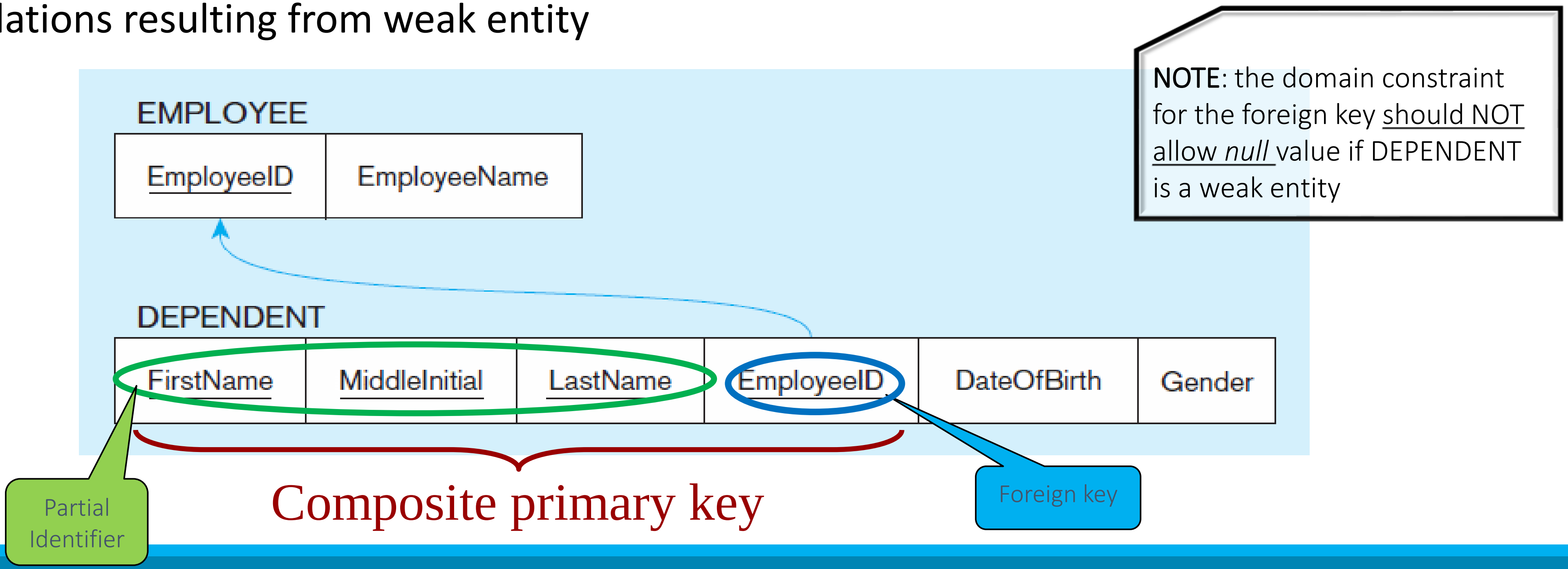
Example of mapping a weak entity (Figure 4-11)

a) Weak entity DEPENDENT



Partial Identifier

b) Relations resulting from weak entity



EMPLOYEE

EmployeeID	EmployeeName

DEPENDENT

FirstName	MiddleInitial	LastName	EmployeeID	DateOfBirth	Gender

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

Partial Identifier

Composite primary key

Foreign key

3.3. Mapping Binary Relationships

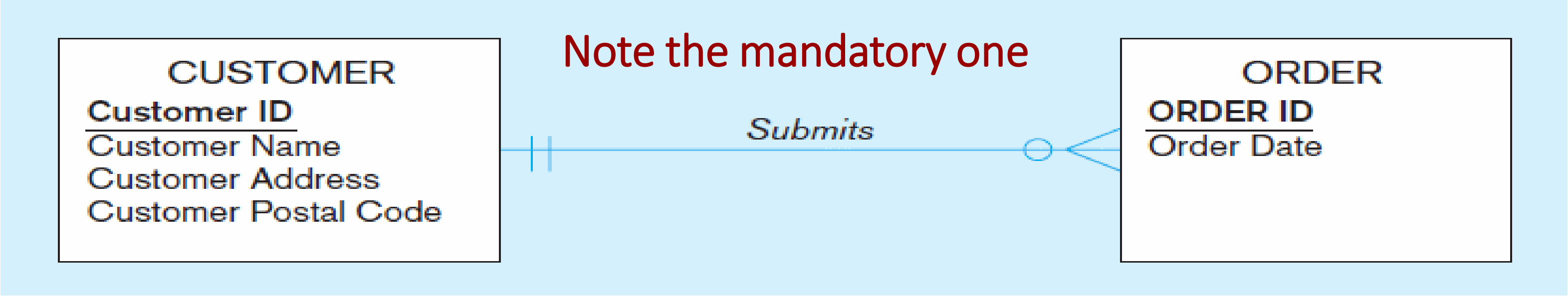
3.3.1. One-to-Many–Primary key on the one side becomes a foreign key on the many side

3.3.1. Many-to-Many–Create a ***new relation*** with the primary keys of the two entities as its primary key

3.3.1. One-to-One–Primary key on mandatory side becomes a foreign key on optional side

3.3.1. Mapping a 1:M relationship (Figure 4-12)

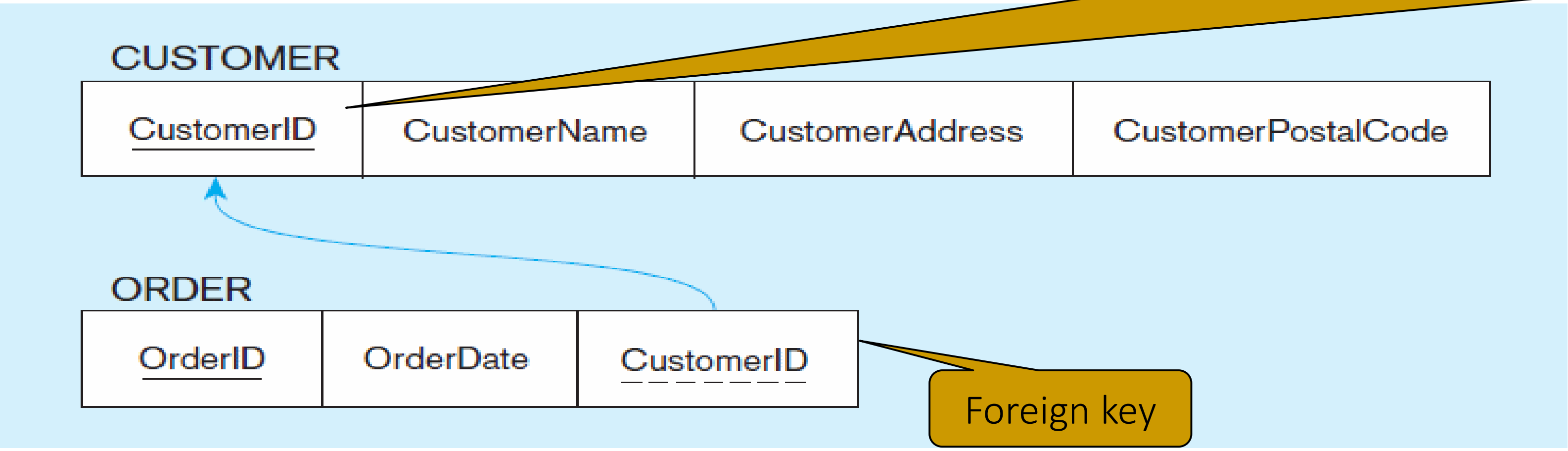
a) Relationship between customers and orders



Note the mandatory one

Question: Can CustomerID in ORDER relation has null values?

b) Mapping the relationship



CUSTOMER

CustomerID	CustomerName	CustomerAddress	CustomerPostalCode

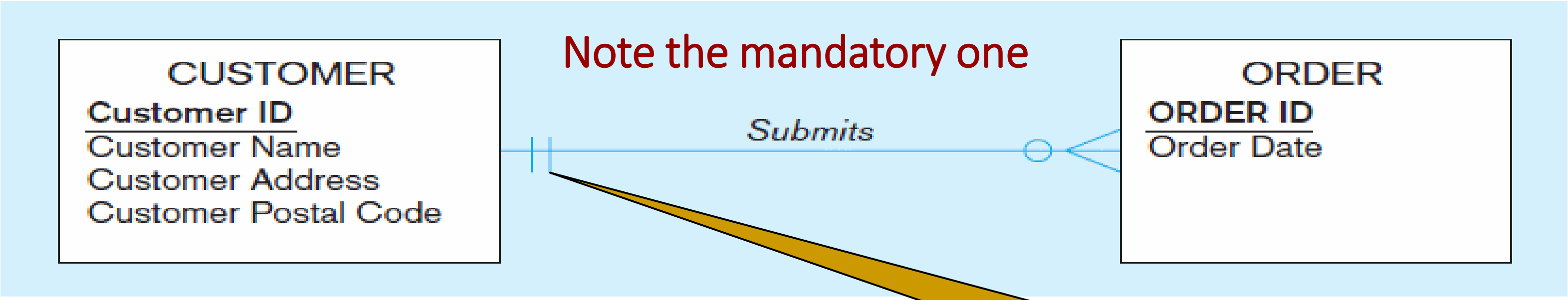
ORDER

OrderID	OrderDate	CustomerID

Foreign key

3.3.1. Mapping a 1:M relationship (Figure 4-12)

a) Relationship between customers and orders



CUSTOMER
Customer ID
Customer Name
Customer Address
Customer Postal Code

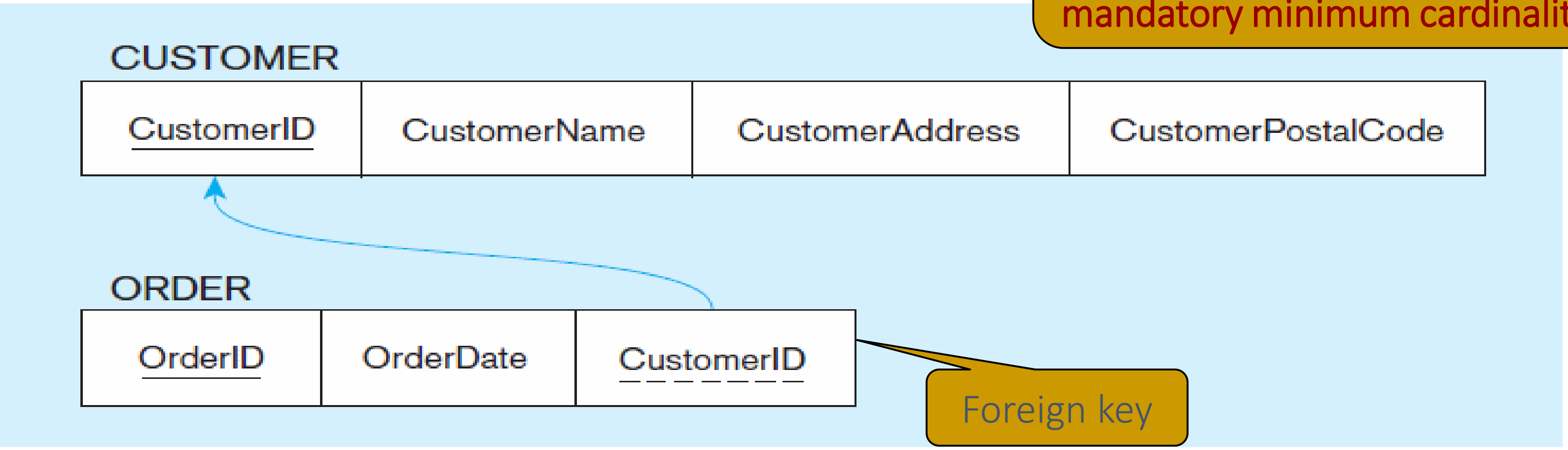
Submits

ORDER
ORDER ID
Order Date

Note the mandatory one

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

b) Mapping the relationship



CUSTOMER

CustomerID	CustomerName	CustomerAddress	CustomerPostalCode

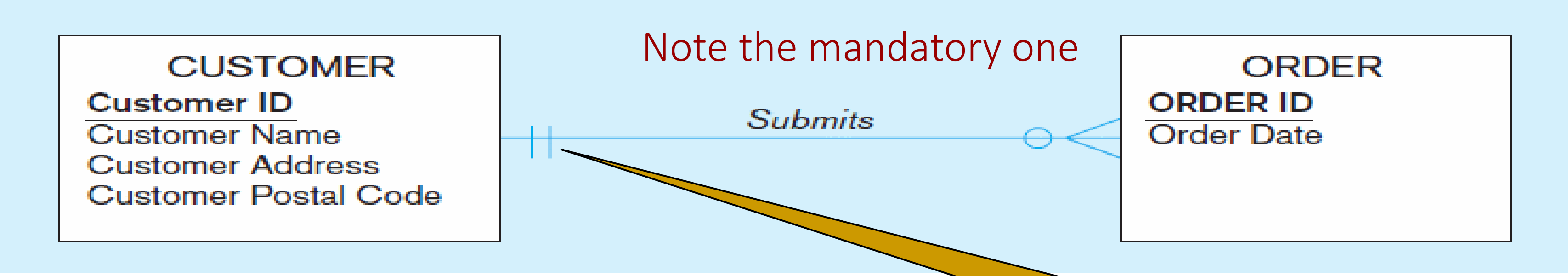
ORDER

OrderID	OrderDate	CustomerID

Foreign key

3.3.1. Mapping a 1:M relationship- Other Format

a) Relationship between customers and orders



Note the mandatory one

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

b) Mapping the relationship

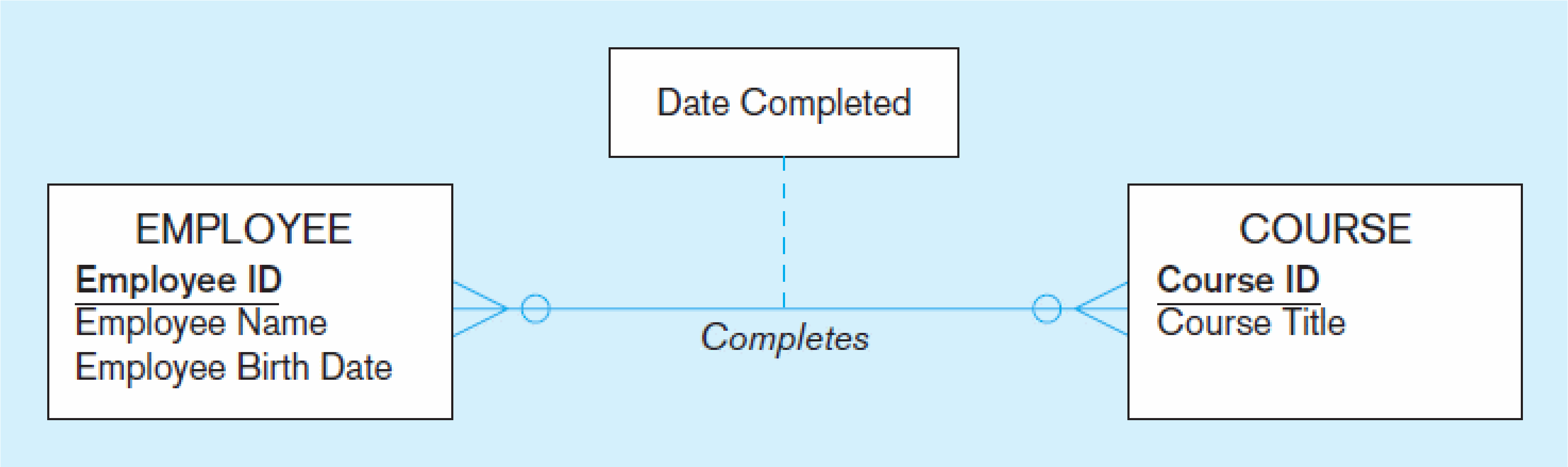
CUSTOMER (<u>CustomerID</u>, CustomerName, CustomerAddress, CustomerPostalCode)

ORDER (<u>OrderID</u>, OrderDate, CustomerID*)
FK (CustomerID) references CUSTOMER

Foreign key

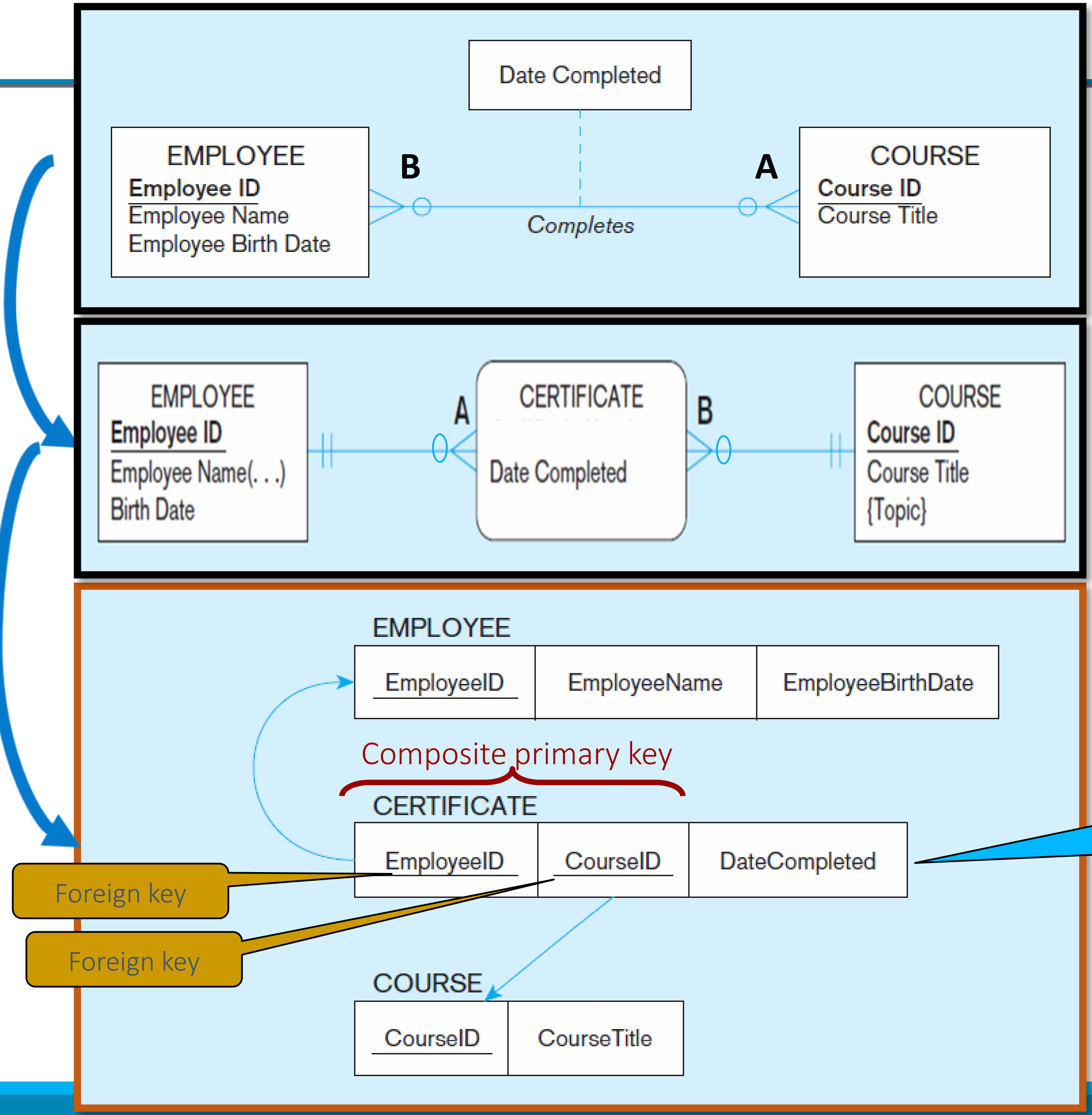
3.3.2. Mapping an M:N relationship (Figure 4-13)

a) Completes relationship (M:N)



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

3.3.2. Mapping an M:N relationship (Figure 4-13) (cont.)



b) Three resulting relations

EMPLOYEE

EmployeeID	EmployeeName	EmployeeBirthDate

Composite primary key

CERTIFICATE

EmployeeID	CourseID	DateCompleted

COURSE

CourseID	CourseTitle

Foreign key

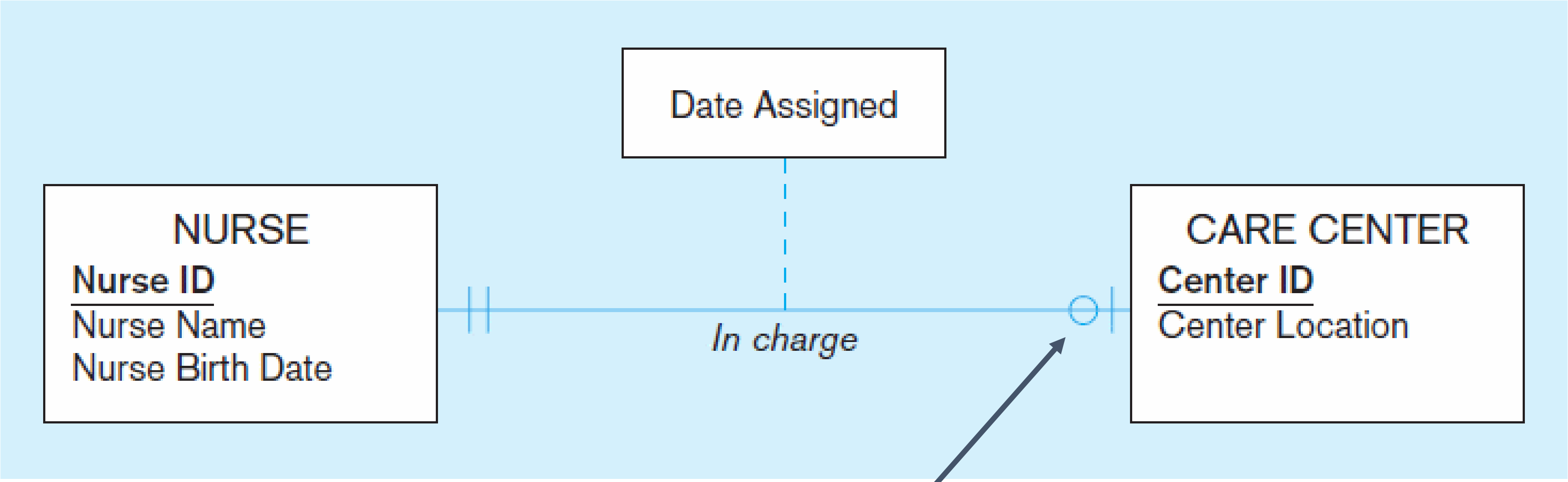
Foreign key

new intersection relation related to the associative entity

3.3.3. Mapping a binary 1:1 relationship (Figure 4-14)

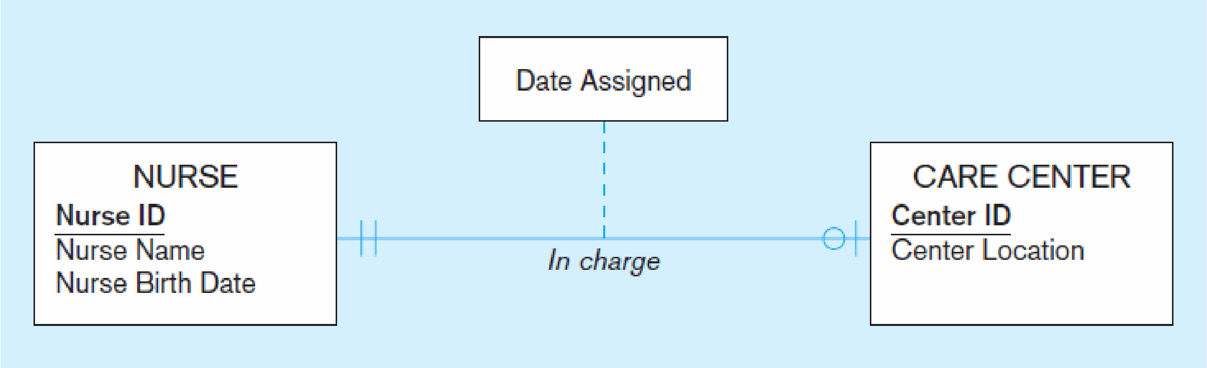
a) In charge relationship (1:1)

Rule: in 1:1 relationships, PK of the entity on the Mandatory side will be FK in the entity on the Optional side



Often in 1:1 relationships, one direction is optional

3.3.3. Mapping a binary 1:1 relationship (Figure 4-14) (cont.)



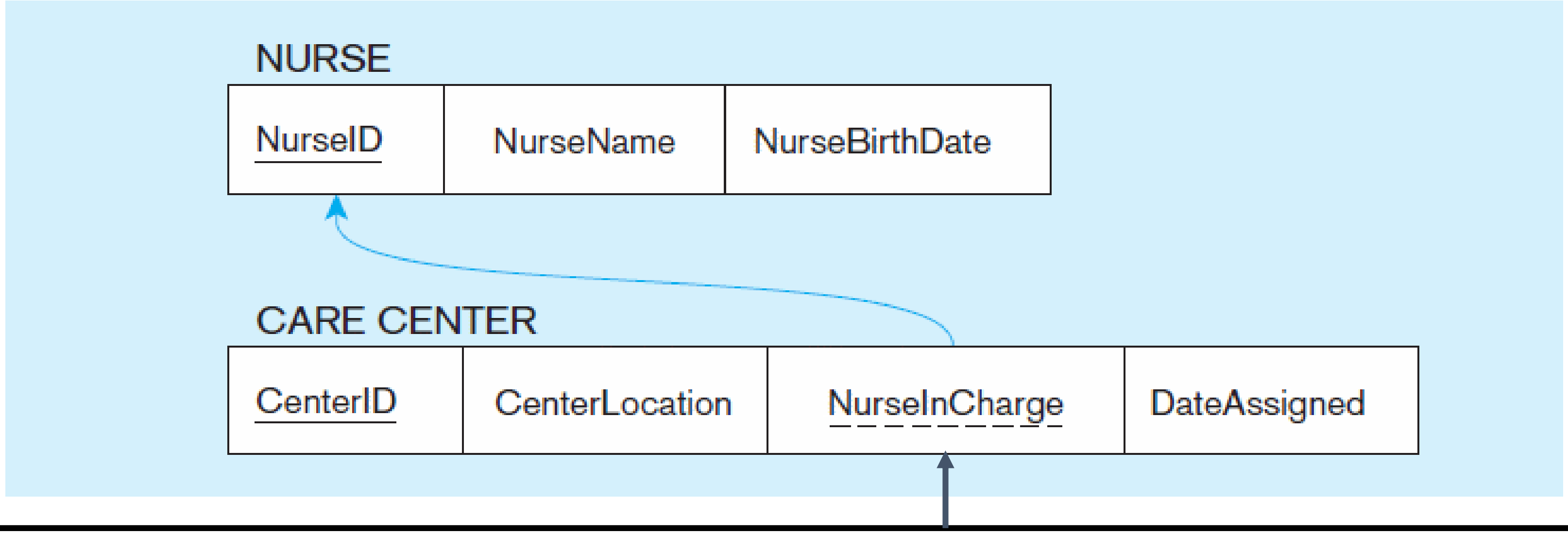
Date Assigned

NURSE
Nurse ID
Nurse Name
Nurse Birth Date

In charge

CARE CENTER
Center ID
Center Location

b) Resulting relations



NURSE

NurseID	NurseName	NurseBirthDate

CARE CENTER

CenterID	CenterLocation	NurseInCharge	DateAssigned

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

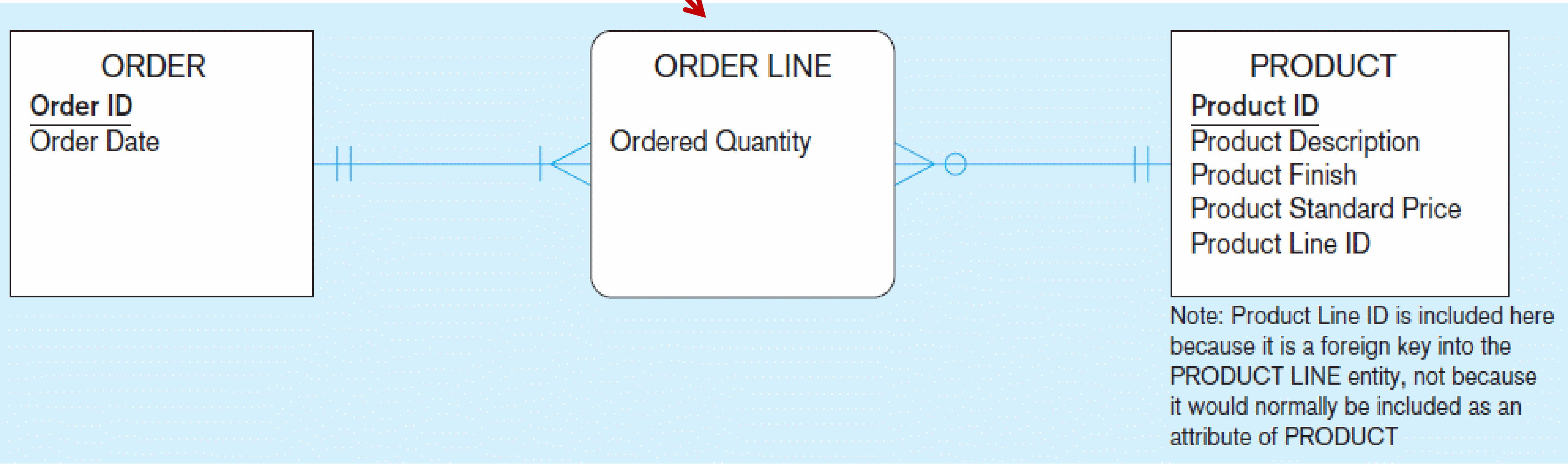
3.4. Mapping Associative Entities

- 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)

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

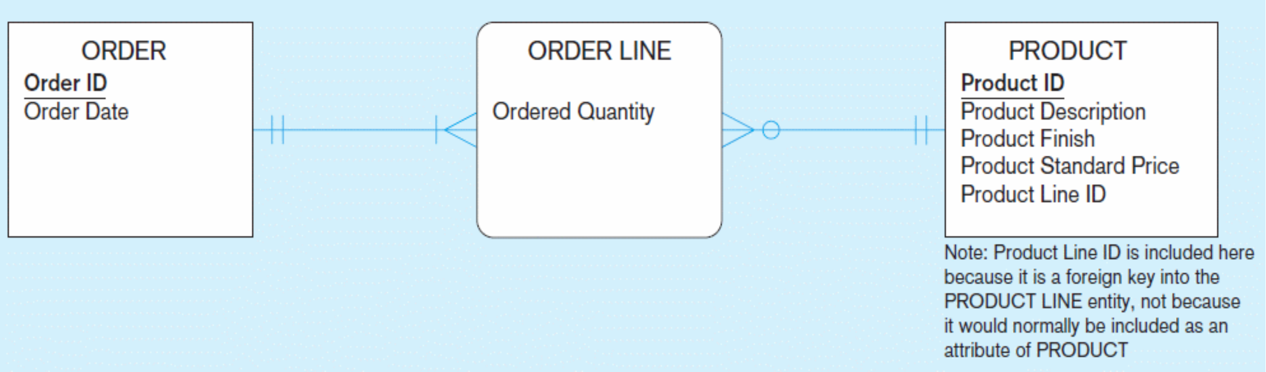
Example of mapping an associative entity (Figure 4-15)

a) An associative entity



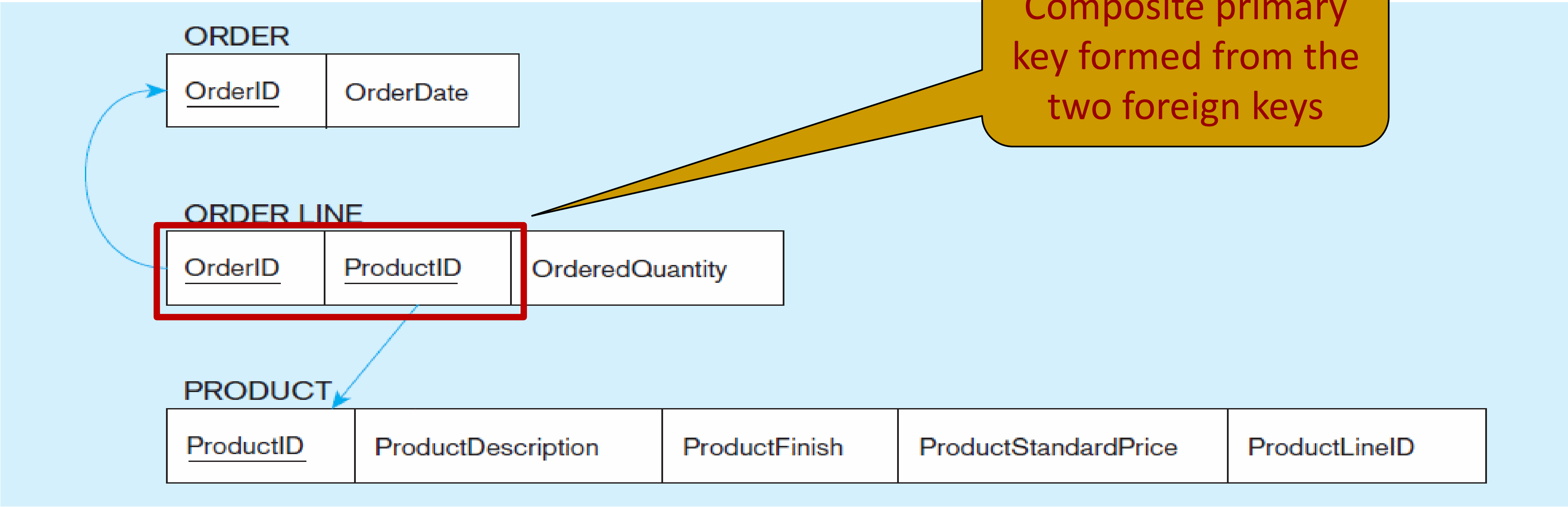
Note: Product Line ID is included here because it is a foreign key into the PRODUCT LINE entity, not because it would normally be included as an attribute of PRODUCT

Example of mapping an associative entity (Figure 4-1) (cont.)



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

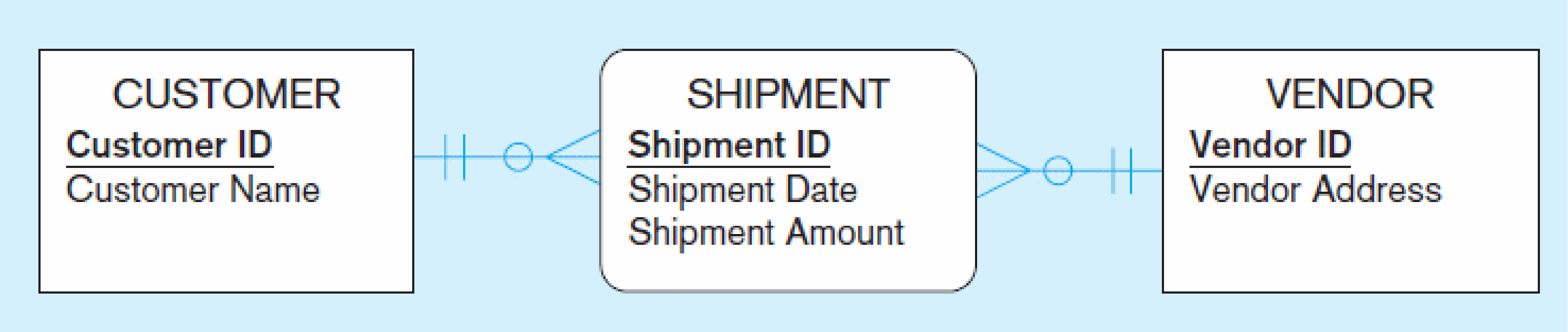
b) Three resulting relations



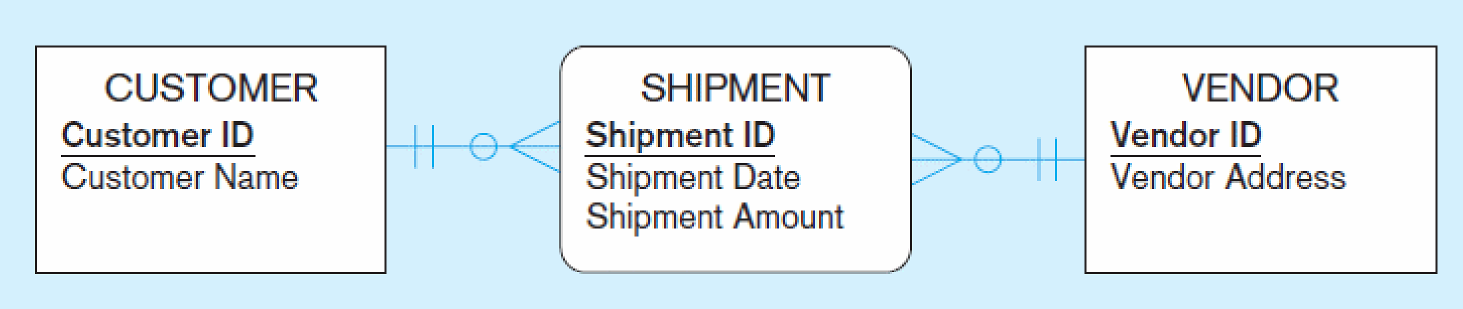
Composite primary key formed from the two foreign keys

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

a) SHIPMENT associative entity



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

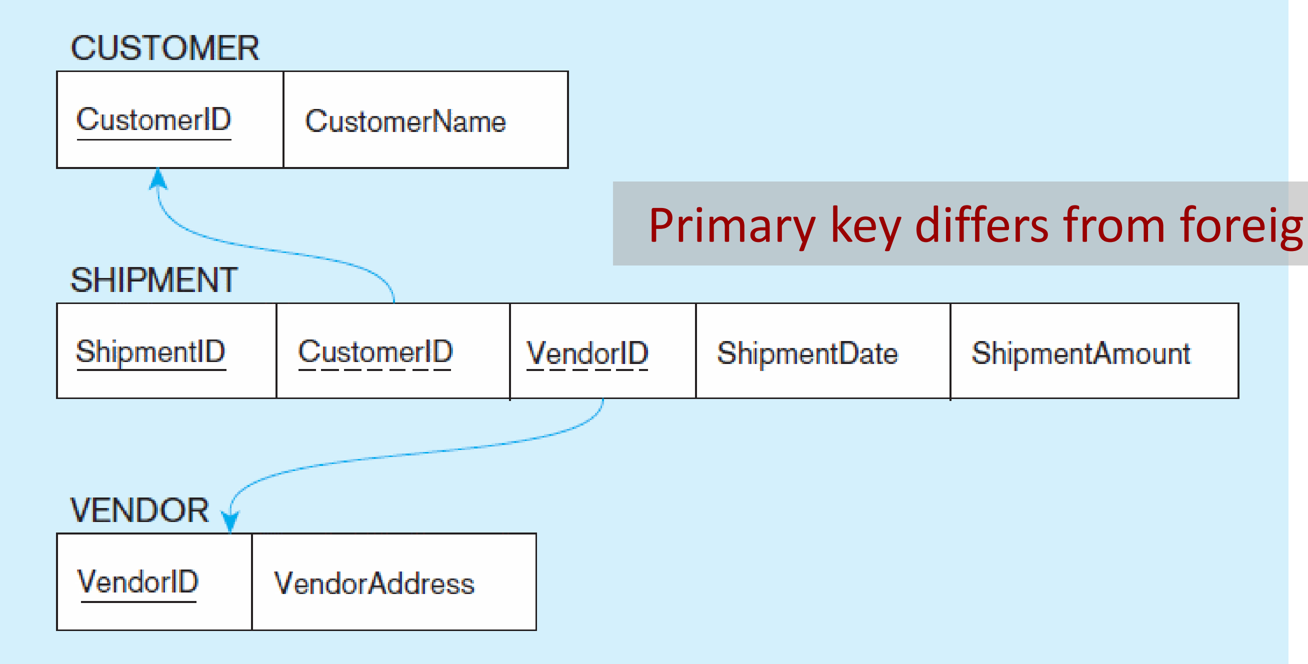


CUSTOMER
Customer ID
Customer Name

SHIPMENT
Shipment ID
Shipment Date
Shipment Amount

VENDOR
Vendor ID
Vendor Address

b) Three resulting relations



CUSTOMER
CustomerID | CustomerName

SHIPMENT
ShipmentID | CustomerID | VendorID | ShipmentDate | ShipmentAmount

VENDOR
VendorID | VendorAddress

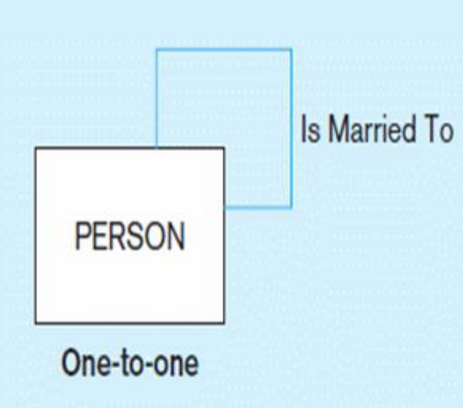
Primary key differs from foreign keys

3.5. Mapping Unary Relationships

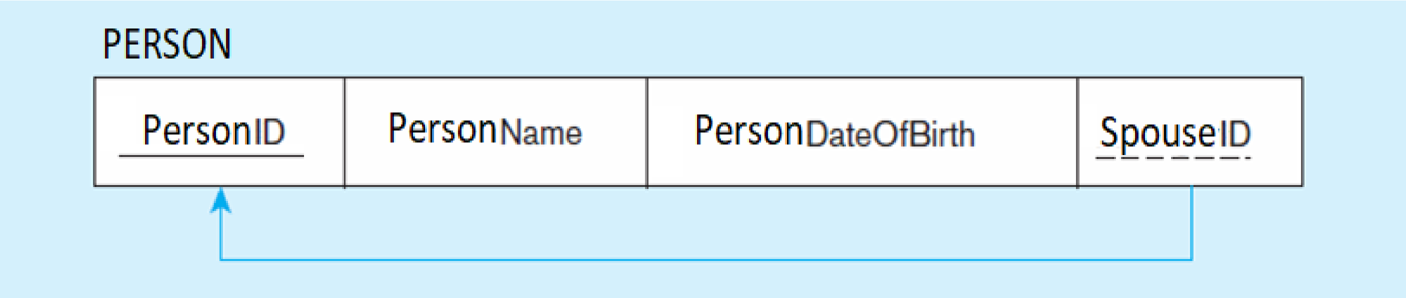
- **One-to-One and One-to-Many**–Recursive foreign key in the same relation
- **Many-to-Many**–Two relations:
  - One for the entity type
  - One for an associative relation in which the primary key has two attributes, both taken from the primary key of the entity

Mapping a unary 1:1 relationship

(a) **PERSON** entity with unary relationship



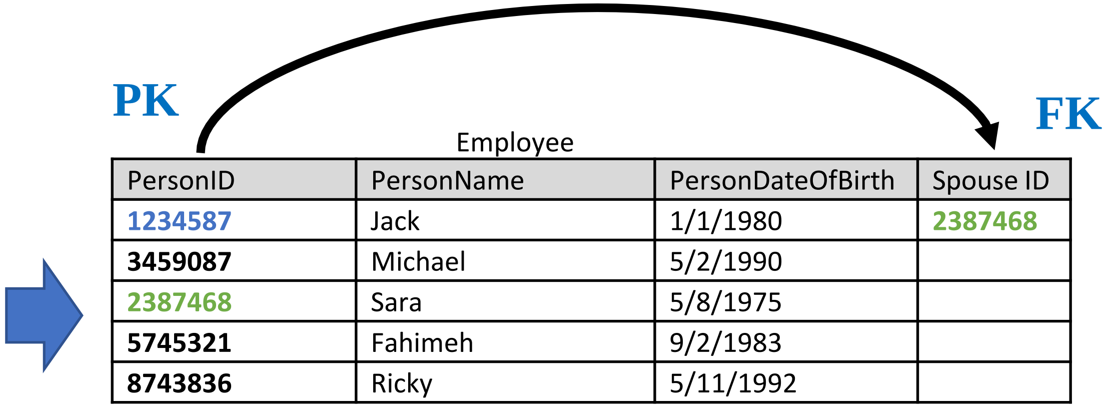
(b) PERSON relation with recursive foreign key



PERSON

PersonID	PersonName	PersonDateOfBirth	SpouseID

PERSON (PersonID, PersonName, PersonDateOfBirth, SpouseID*)
FK (SpouseID) references PERSON

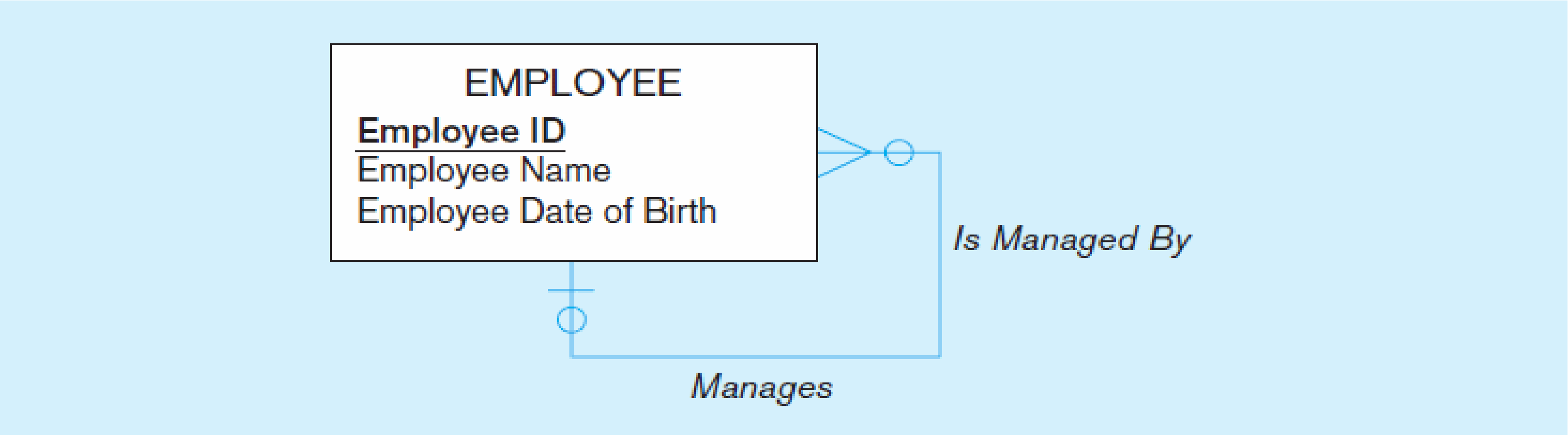


Employee

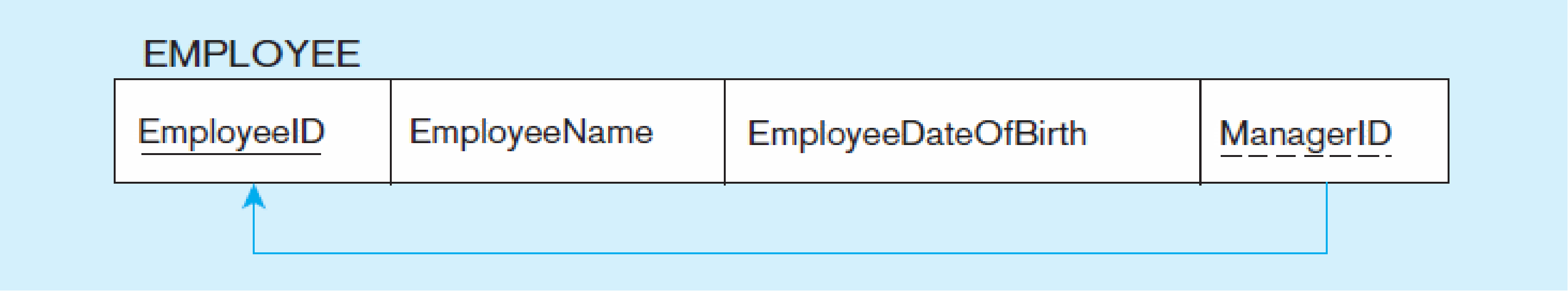
PersonID	PersonName	PersonDateOfBirth	Spouse ID
1234587	Jack	1/1/1980	2387468
3459087	Michael	5/2/1990	
2387468	Sara	5/8/1975	
5745321	Fahimeh	9/2/1983	
8743836	Ricky	5/11/1992	

Mapping a unary 1:N relationship (Figure 4-17)

(a) **EMPLOYEE** entity with unary relationship



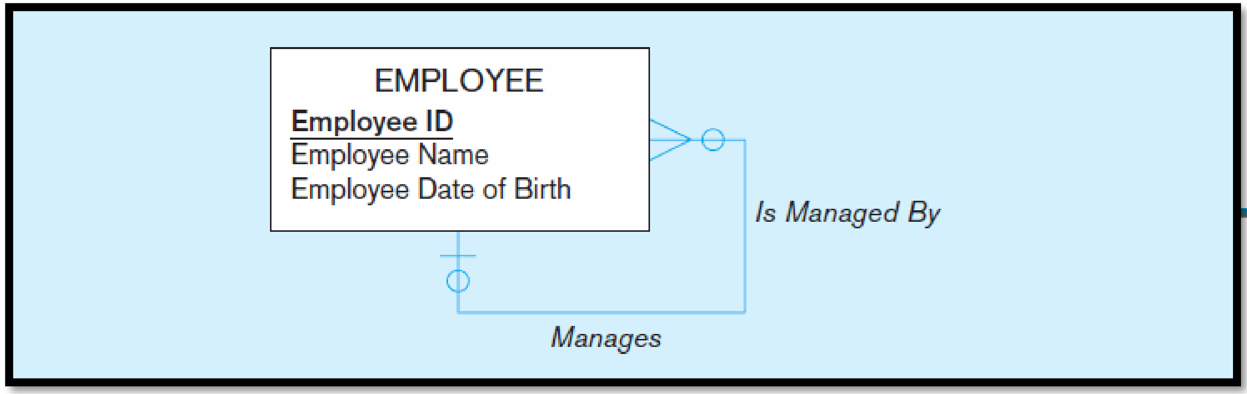
(b) EMPLOYEE relation with recursive foreign key

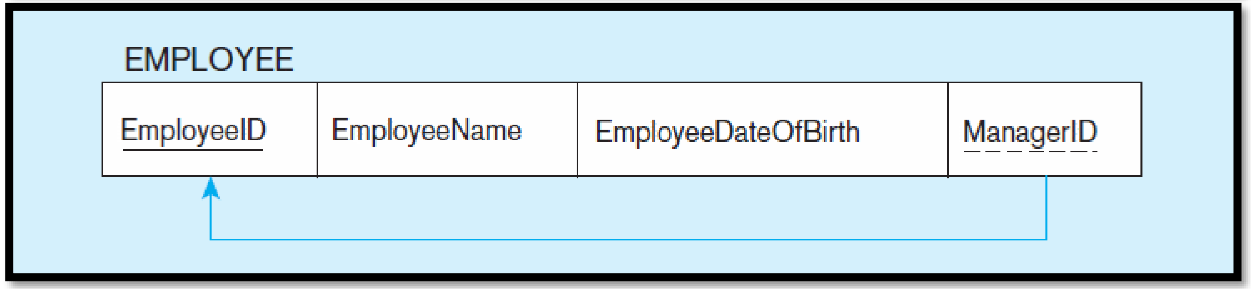


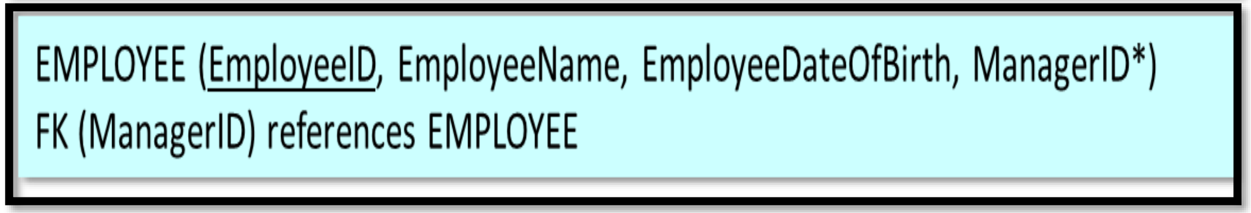
EMPLOYEE

EmployeeID	EmployeeName	EmployeeDateOfBirth	ManagerID

EMPLOYEE (<u>EmployeeID</u>, EmployeeName, EmployeeDateOfBirth, ManagerID*)
FK (ManagerID) references EMPLOYEE







EMPLOYEE (EmployeeID, EmployeeName, EmployeeDateOfBirth, ManagerID*)
FK (ManagerID) references EMPLOYEE

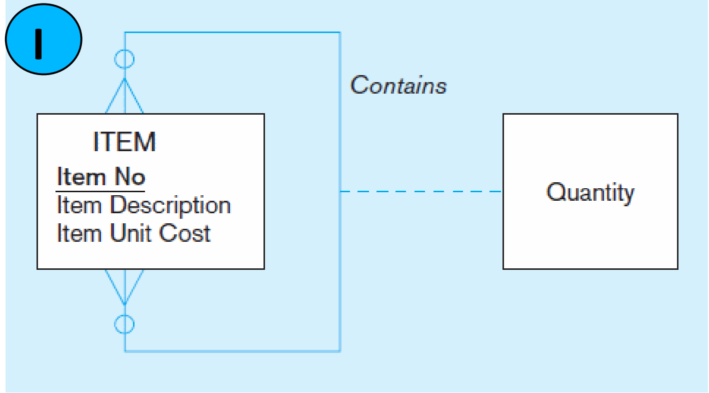
PK

FK

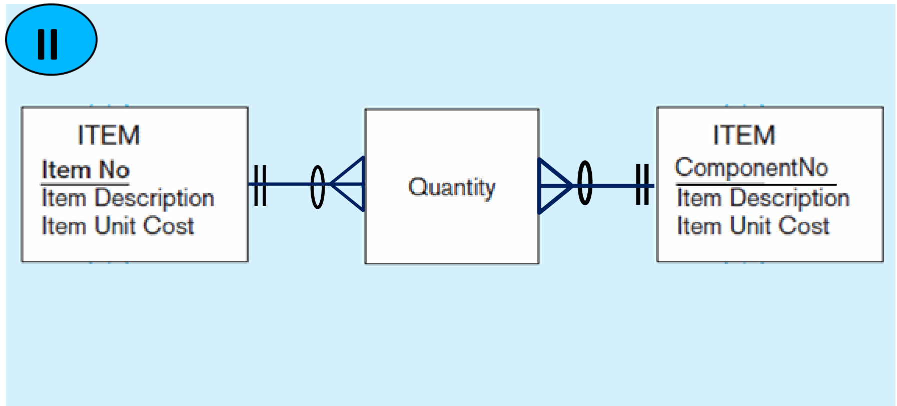
Employee_ID	Employee_Name	Employee_DateOfBirth	Manager_ID
1123	Sara	1.1.2000	7892
1456	Jake	1.1.2000	7892
7892	Fahimeh	1.1.1970	1245
1245	Julia	1.1.1980	…
…	…	…	…

Mapping a unary M:N relationship (Figure 4-18)

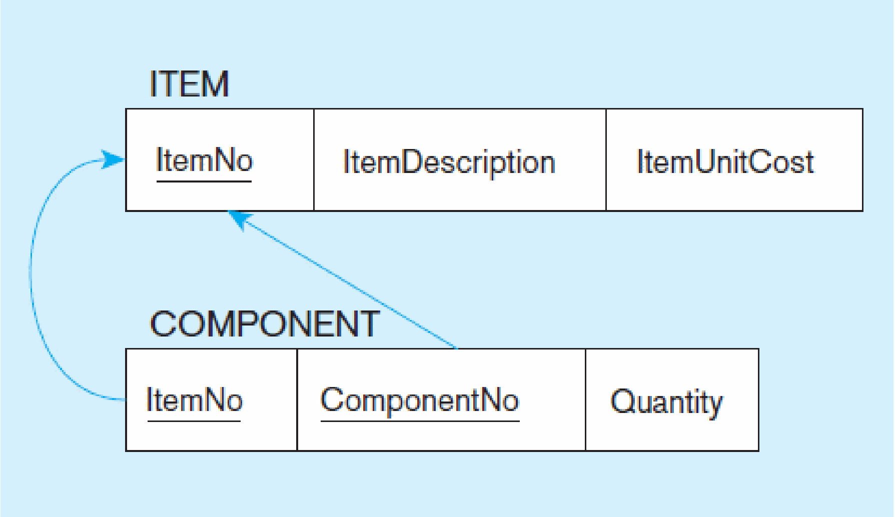
- Bill-of-materials relationships (M:N)



ERD II is just provided to clarify ERD I.
ERD II is not a standard ERD



- ITEM and COMPONENT relations



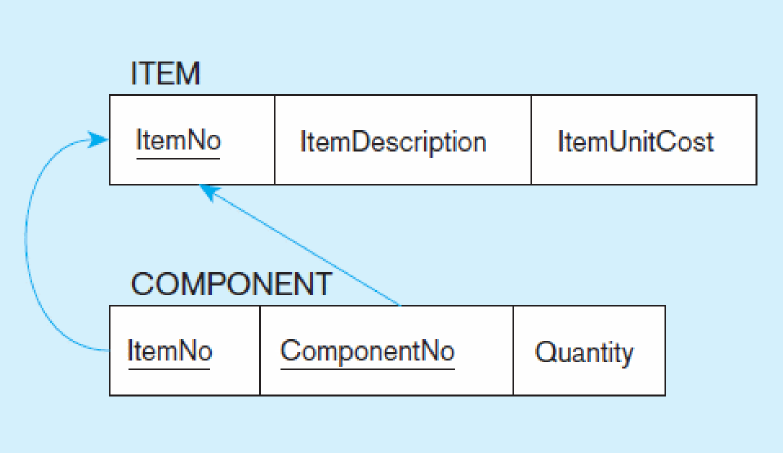
ITEM

ItemNo	ItemDescription	ItemUnitCost

COMPONENT

ItemNo	ComponentNo	Quantity

Example



ITEM

ItemNo	ItemDescription	ItemUnitCost

COMPONENT

ItemNo	ComponentNo	Quantity

Item_No	Item_Description	Item_Unit_Cost
12	Wheel	50
13	Spoke	0.5
14	Rim	30
15	Valve	5

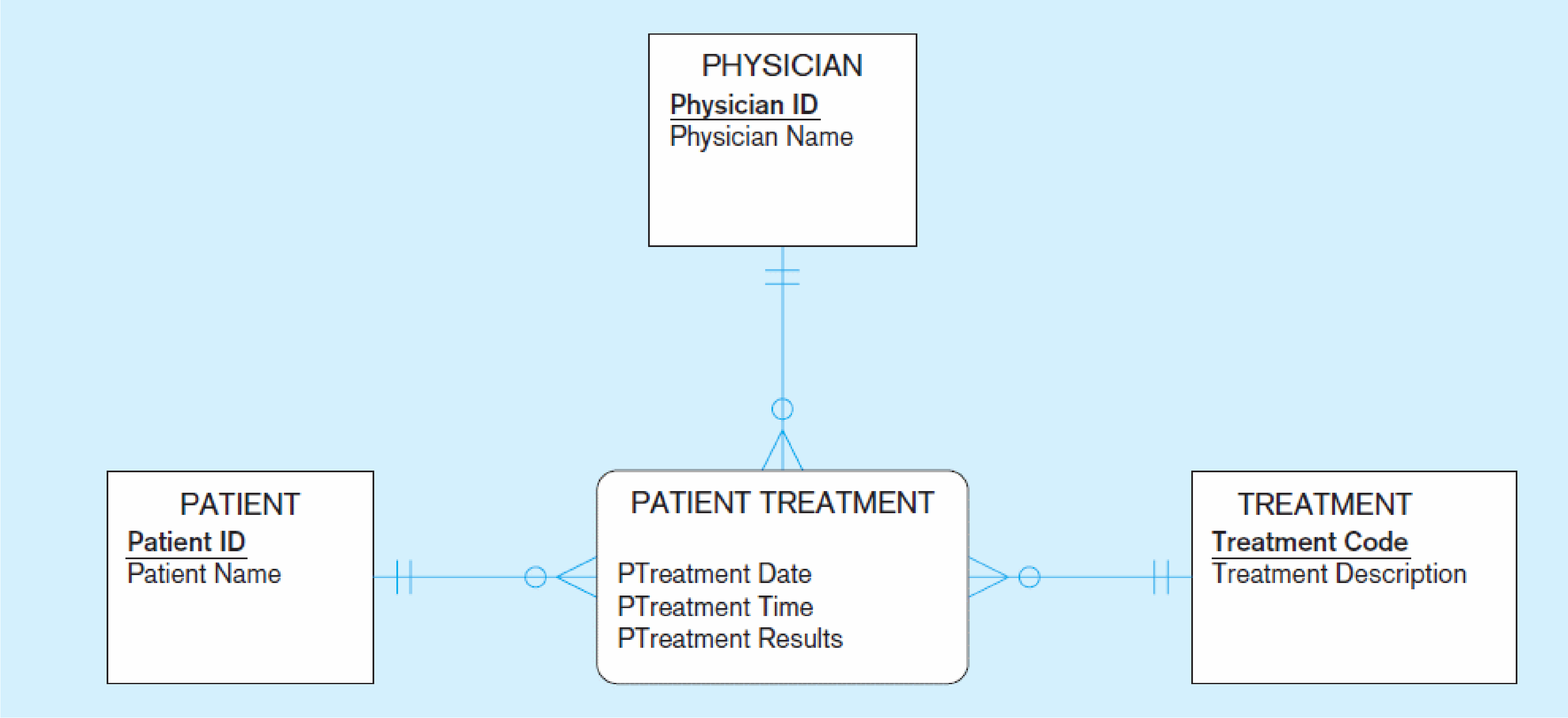
Item_No	Component_No	Quantity
12	13	30
12	14	1
12	15	1

3.6. Mapping Ternary (and n-ary) Relationships

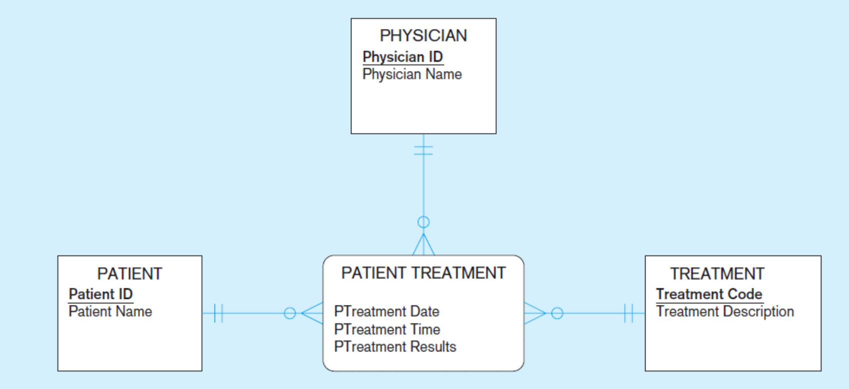
- **One relation for each entity and one for the associative entity**
- **Associative entity has foreign keys to each entity in the relationship**

Mapping a ternary relationship (Figure 4-19)

a) PATIENT TREATMENT Ternary relationship with associative entity



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



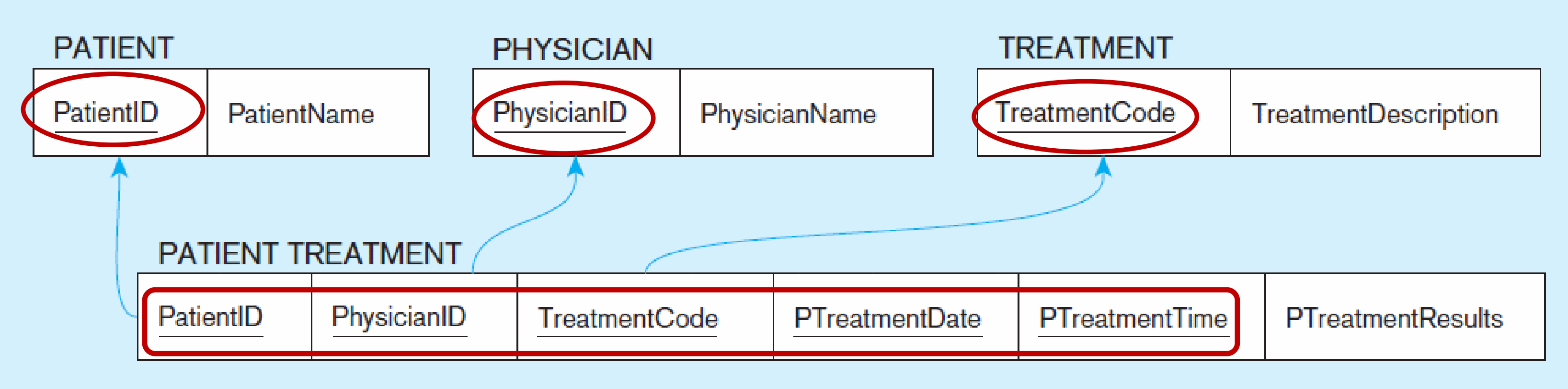
PHYSICIAN
Physician ID
Physician Name

PATIENT
Patient ID
Patient Name

PATIENT TREATMENT
PTreatment Date
PTreatment Time
PTreatment Results

TREATMENT
Treatment Code
Treatment Description

b) Mapping the ternary relationship PATIENT TREATMENT



PATIENT

PatientID	PatientName

PHYSICIAN

PhysicianID	PhysicianName

TREATMENT

TreatmentCode	TreatmentDescription

PATIENT TREATMENT

PatientID	PhysicianID	TreatmentCode	PTreatmentDate	PTreatmentTime	PTreatmentResults

Remember that the primary key MUST be unique.

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

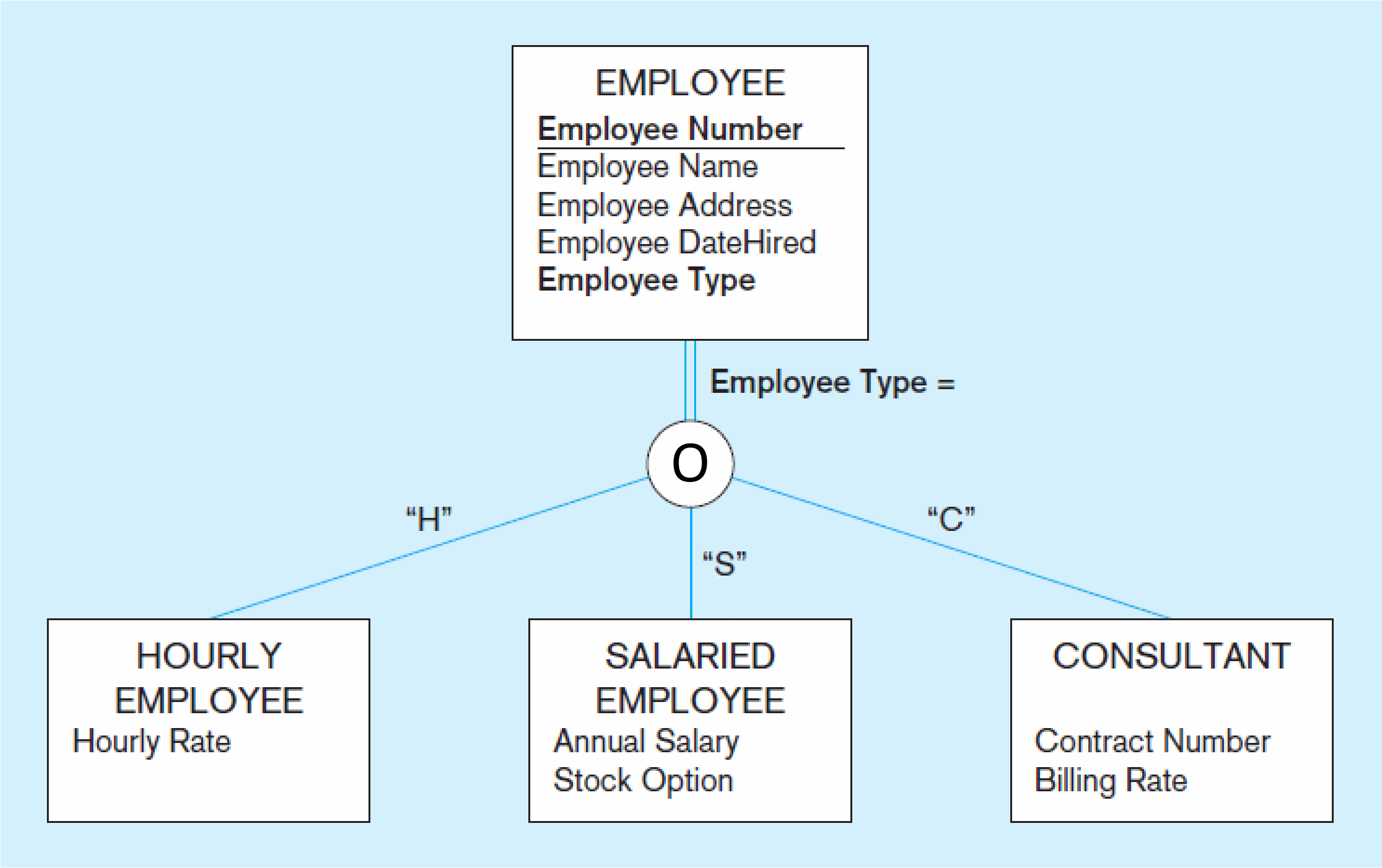
But this makes a very cumbersome key…

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

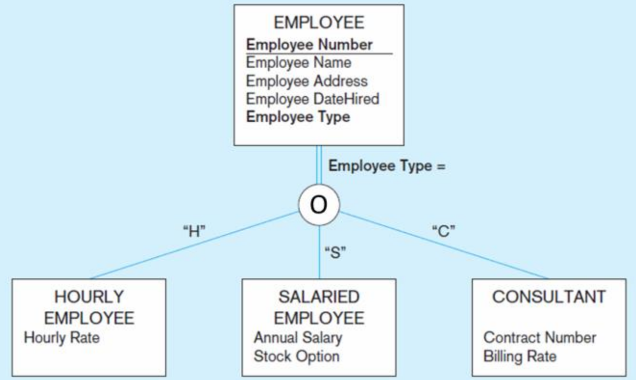
3.7. Mapping Supertype/Subtype Relationships

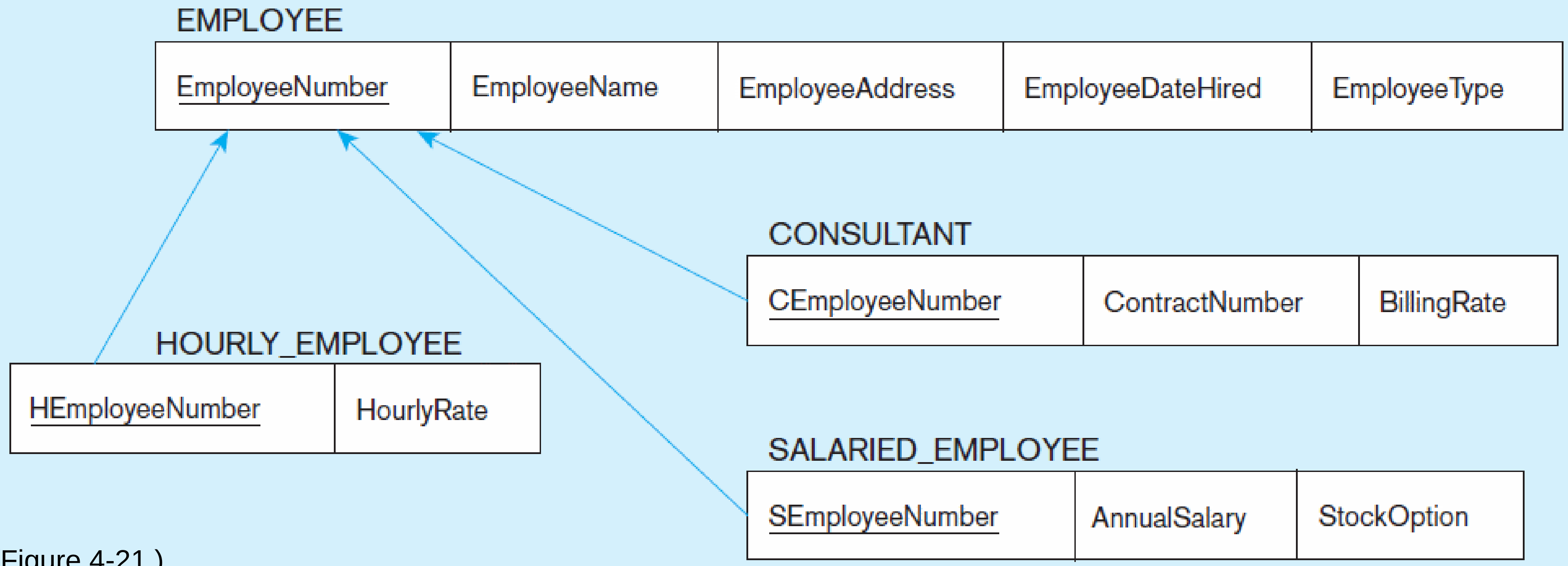
- One relation for supertype and for each subtype
- Supertype attributes (including identifier and subtype discriminator) go into supertype relation
- Subtype attributes go into each subtype; primary key of supertype relation also becomes primary key of subtype relation
- 1:1 relationship established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

Supertype/subtype relationships (Figure 4-20)



Mapping supertype/subtype relationships to relations





(Figure 4-21)

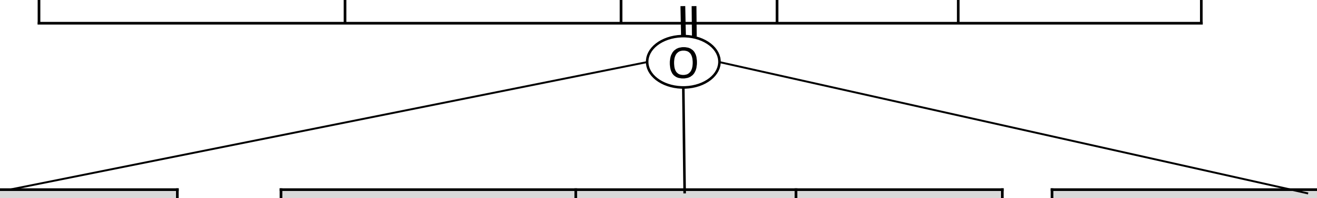
These are implemented as one-to-one relationships.

Note: This is the best method to map supertype/subtypes to relations. There are other two methods that will be discussed in the related tutorial.

Introducing a subtype discriminator (overlap rule)

Employee_Number	Employee_Name	Address	Date_Hire	Employe_type (H? S? C?)
1123	Sara	UTS	1/1/2014	YNY
1456	Jake	32/50 …	5/8/2013	NYN
7892	Fahimeh	12/97 …	2/3/2013	NNY

Composite Attribute



HEmployeeNumber	HourlyRate
1123	80

SEmployeeNumber	AnnualSalary	StockOption
1456	70000	0.2

CEmployeeNumber	ContractNumber	BillingRate
7892	9856	50
1123	9812	30

Summary

- List properties of relations
- Transform E-R and EER diagrams to relations
- Create tables with entity and relational integrity constraints

Next Lecture…

1. **Terms to know to Do Normalization**
    1.1. Functional Dependencies
    1.2. Keys: Super-key, Candidate key and Primary Key
    1.3. Determining Candidate Keys from FDs
    1.4. Partial Functional Dependencies
    1.5. Transitive Functional Dependencies
2. **Data Normalization and Well-Structured Relations**
3. **Steps in normalization**
4. **First Normal Form**
5. **Second Normal Form**
6. **Third Normal Form**