Database Management Systems
Chapter 4: Logical Database Design

UNIVERSITY OF MARYLAND

ROBERT H. SMITH SCHOOL OF BUSINESS

DR. ADAM LEE

Objectives

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

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.

Relation

- A relation is a named, two-dimensional table of data.
- A table consists of rows (records) and columns (attributes or fields).
- 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.
  - Attributes (columns) in tables must have unique names.
  - The orders of columns and rows must be irrelevant.
- NOTE: All relations are in 1st Normal form (1NF).

Correspondence with ER Model

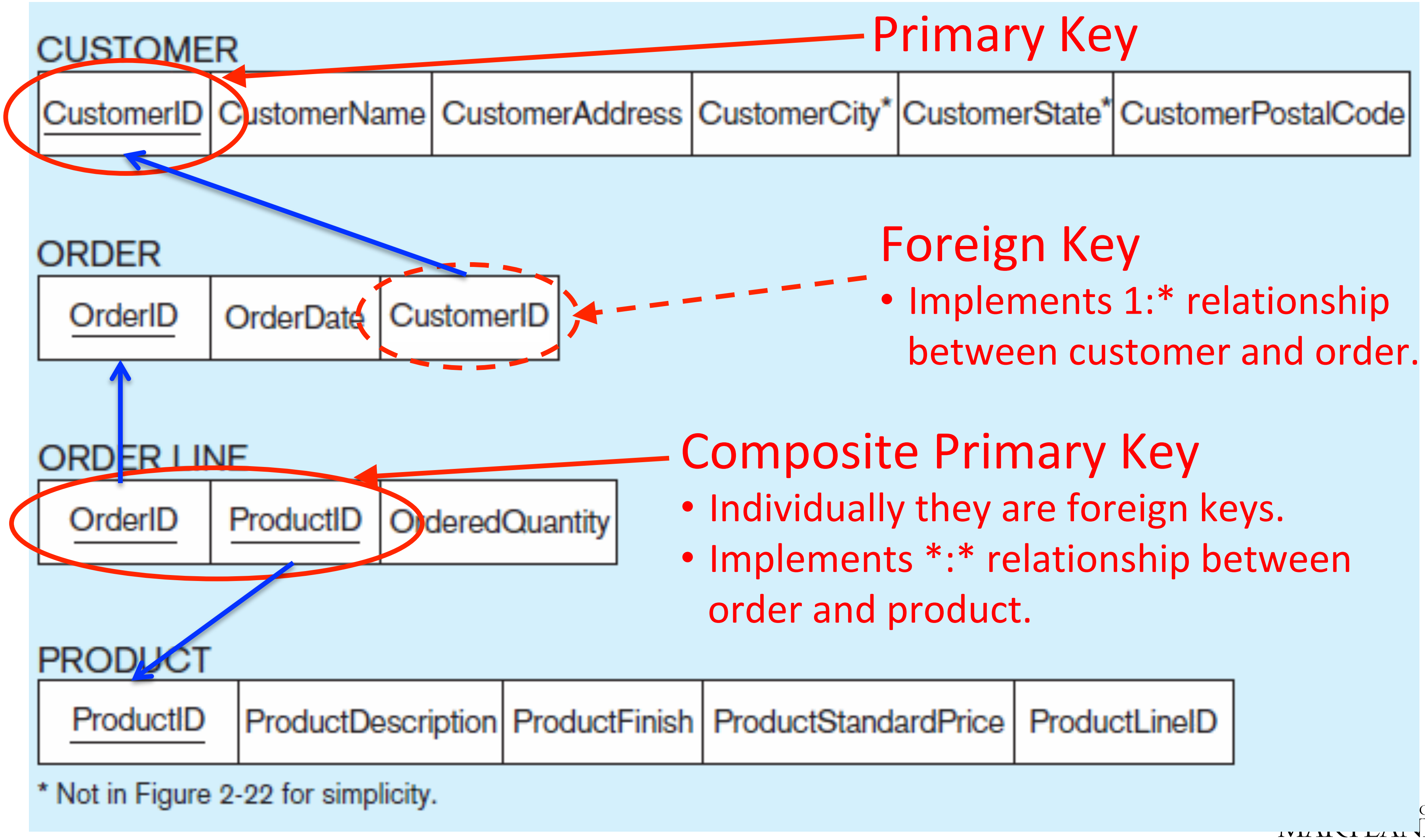
- Relations (tables) correspond with:
  - Entity types
  - Many-to-many relationship types
- Rows correspond with:
  - Entity instances
  - Many-to-many relationship instances
- Columns correspond with:
  - Attributes

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

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 attribute) or composite (a set of attributes).
- Keys usually are used as indexes to speed up the response to user queries (more on this in Chapter 5).

Figure 4-3: Schema for Four Relations



CUSTOMER

CustomerID	CustomerName	CustomerAddress	CustomerCity*	CustomerState*	CustomerPostalCode

Primary Key

ORDER

OrderID	OrderDate	CustomerID

Foreign Key

- Implements 1:* relationship between customer and order.

ORDER LINE

OrderID	ProductID	OrderedQuantity

Composite Primary Key

- Individually they are foreign keys.
- Implements *:* relationship between order and product.

PRODUCT

ProductID	ProductDescription	ProductFinish	ProductStandardPrice	ProductLineID

* Not in Figure 2-22 for simplicity.

Integrity Constraints

- Atomic Constraint.
- **Domain Constraints**:
  - Allowable format and size for an attribute.
  - Allowable values for an attribute.
- Entity Integrity.
- Referential Integrity.

Table 4-1: Domain Definitions Enforce Domain Integrity Constraints

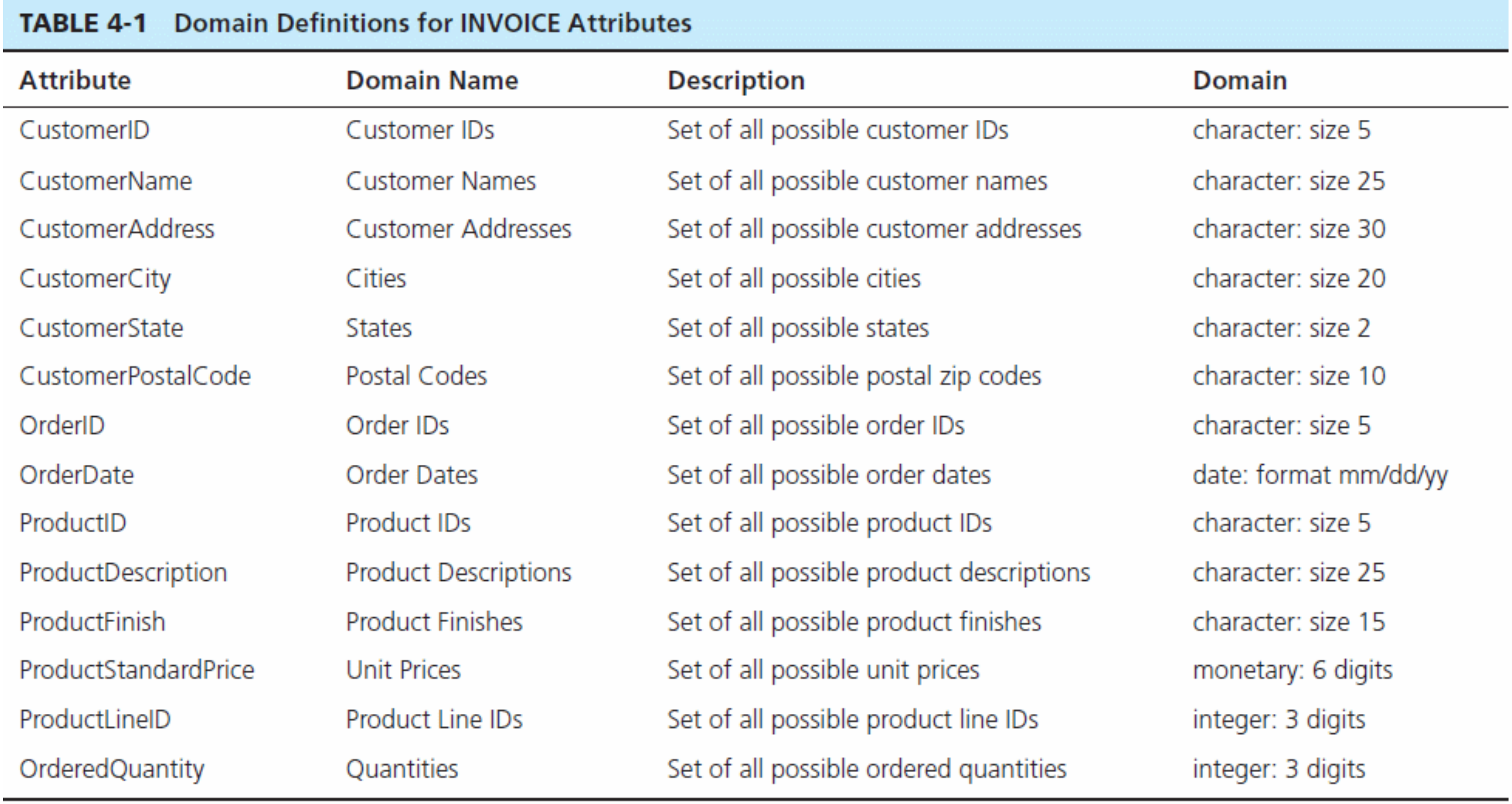


TABLE 4-1 Domain Definitions for INVOICE Attributes

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

Integrity Constraints

- Atomic Constraint.
- Domain Constraints.

- **Entity Integrity**:
  - Every relation has a primary key.
  - Every primary key attribute is NON-null.
  - All primary key attributes MUST have valid data.

- Referential Integrity.

Integrity Constraints

- Atomic Constraint.
- Domain Constraints.
- Entity Integrity.

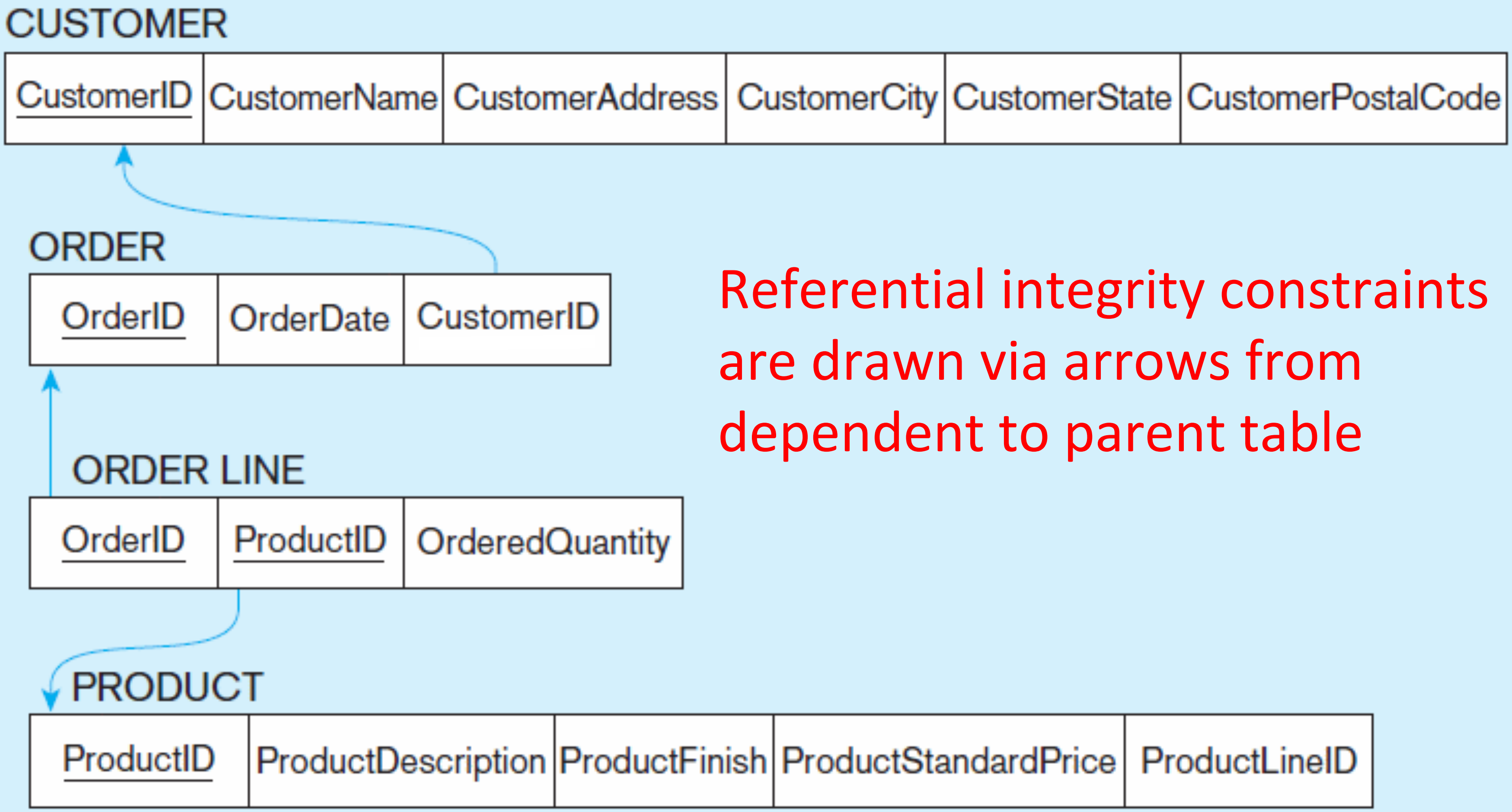
- **Referential Integrity**:
  - To maintain consistency among the rows of two relations.
  - If there is a foreign key in one relation, either each foreign key value MUST match a primary key value in one relation, or the foreign key value MUST be null.

Referential Integrity

- **ON DELETE** rules
  - **RESTRICT | NO ACTION** – do not allow delete of “primary” side if related rows exist in “foreign” side
  - **CASCADE** – automatically delete “foreign” side rows that correspond with the “primary” side row to be deleted
  - **SET NULL** – set the foreign key in the “foreign” side to null if deleting from the “primary” side → not allowed for weak entities

- **ON UPDATE** rules
  - **RESTRICT | NO ACTION**
  - **CASCADE**
  - **SET NULL**

Figure 4-5: Referential Integrity Constraints



CUSTOMER

CustomerID	CustomerName	CustomerAddress	CustomerCity	CustomerState	CustomerPostalCode

ORDER

OrderID	OrderDate	CustomerID

ORDER LINE

OrderID	ProductID	OrderedQuantity

PRODUCT

ProductID	ProductDescription	ProductFinish	ProductStandardPrice	ProductLineID

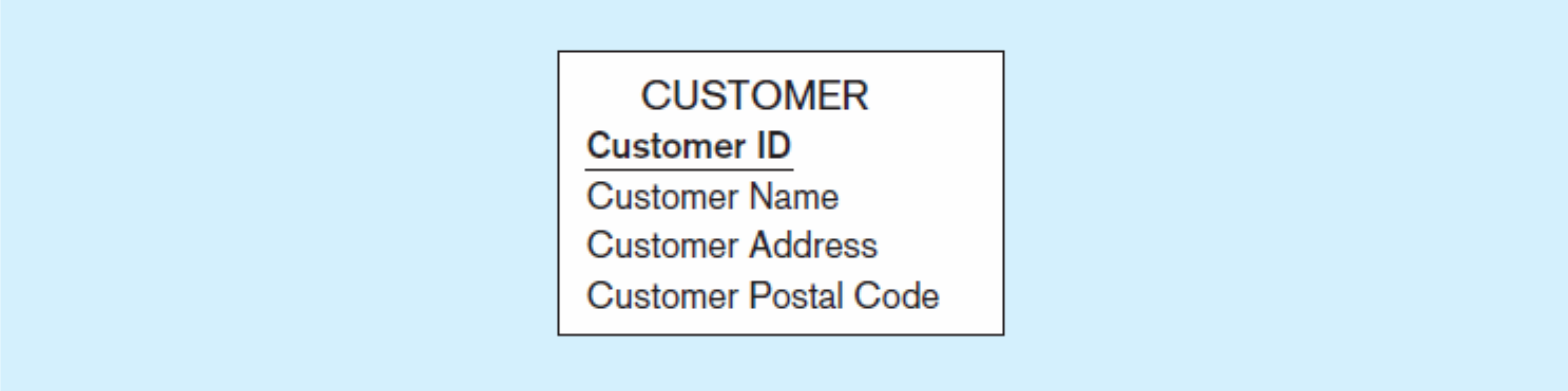
Referential integrity constraints are drawn via arrows from dependent to parent table

Transforming ER Diagrams into Relations Step 1

- Mapping regular entities to relations:
  - **Simple attributes**: ER attributes map directly onto the relation.
  - **Composite attributes**: Use only their simple, component attributes.
  - **Multivalued attribute**: Becomes a separate relation with a foreign key taken from the superior entity.

Figure 4-8 Mapping a Regular Entity

(a) CUSTOMER entity type with simple attributes



CUSTOMER

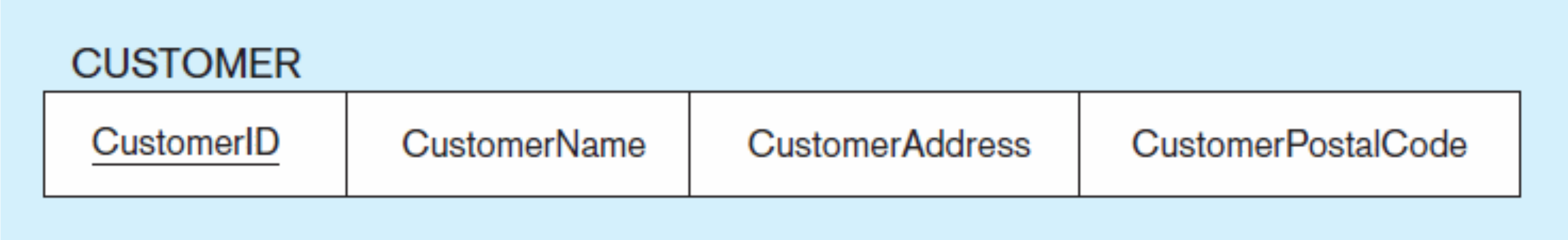
Customer ID

Customer Name

Customer Address

Customer Postal Code

(b) CUSTOMER relation

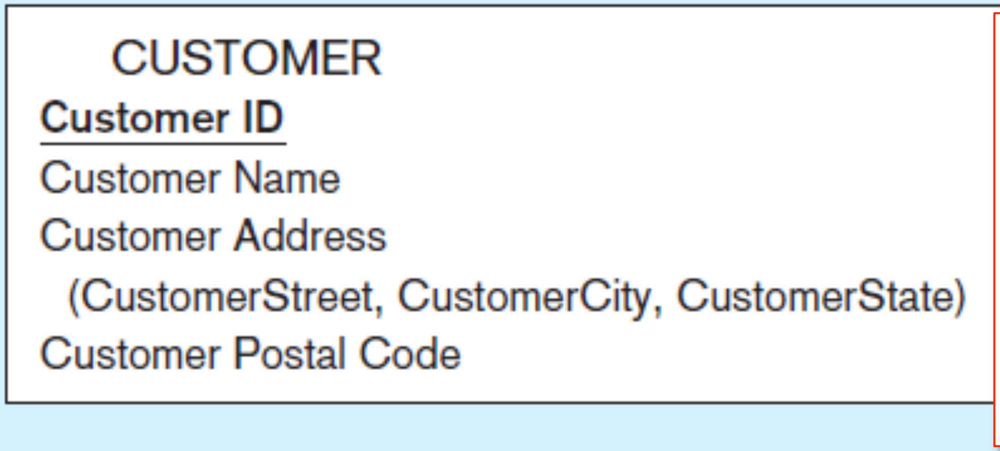


CUSTOMER

CustomerID	CustomerName	CustomerAddress	CustomerPostalCode

Figure 4-9 Mapping a Composite Attribute

(a) CUSTOMER entity type with composite attribute



CUSTOMER

Customer ID

Customer Name

Customer Address

(CustomerStreet, CustomerCity, CustomerState)

Customer Postal Code

Customer

customerId

customerName

customerAddress

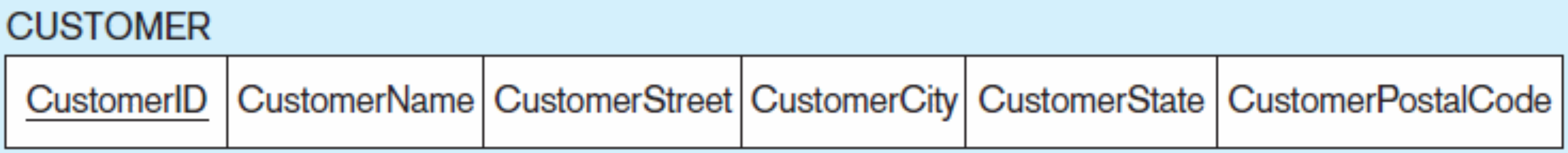
-customerStreet

-customerCity

-customerState

customerPostalCode

(b) CUSTOMER relation with address detail

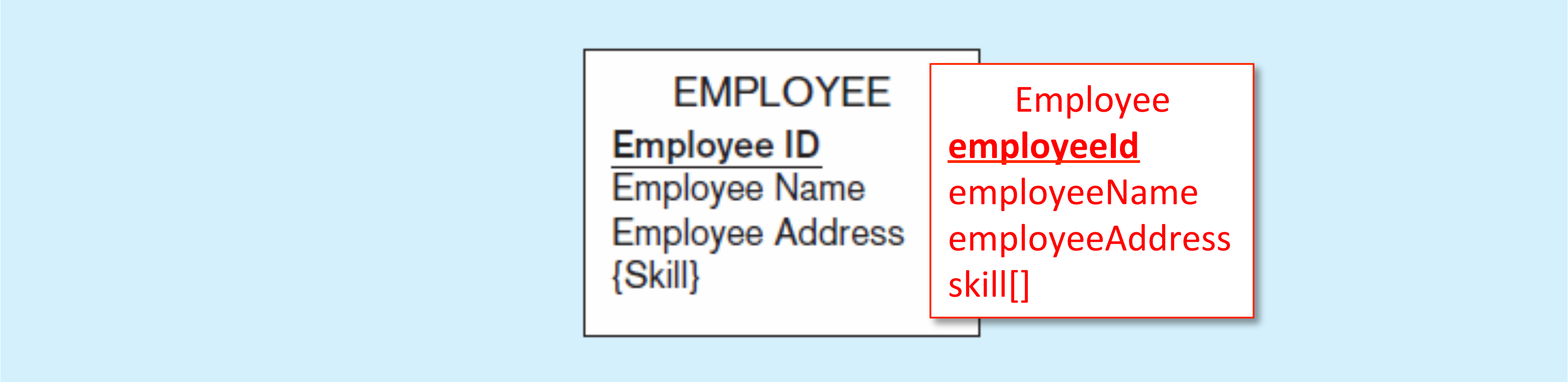


CUSTOMER

CustomerID	CustomerName	CustomerStreet	CustomerCity	CustomerState	CustomerPostalCode

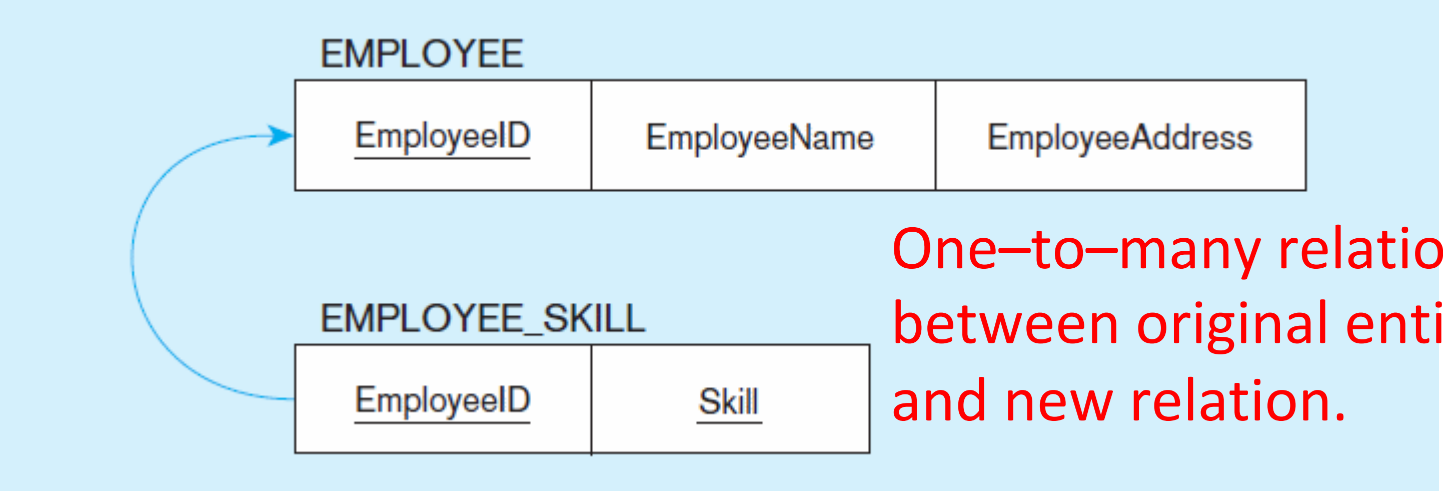
Figure 4-10 Mapping a Multivalued Attribute

(a) EMPLOYEE entity type with multivalued attribute



Employee
employeeId
employeeName
employeeAddress
skill[]

(b) Multivalued attribute becomes a separate relation with foreign key



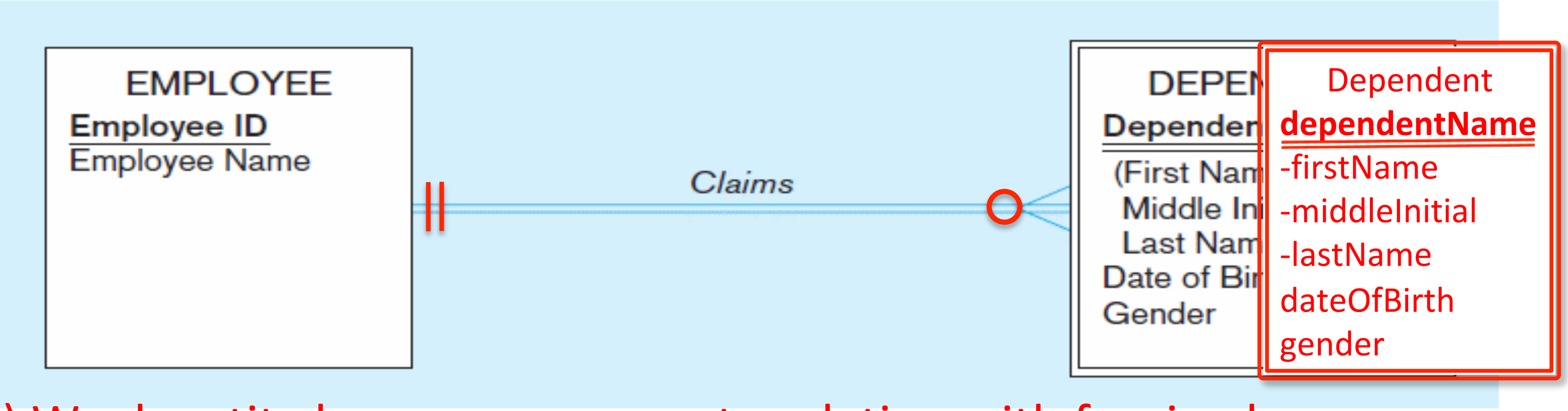
One–to–many relationship between original entity and new relation.

Transforming ER Diagrams into Relations Step 2

- Mapping weak entities:
  - Becomes a separate relation with a foreign key taken from the superior entity.
  - Primary key composed of (i) partial identifier of weak entity, and (ii) primary key of identifying relation (i.e. strong entity).

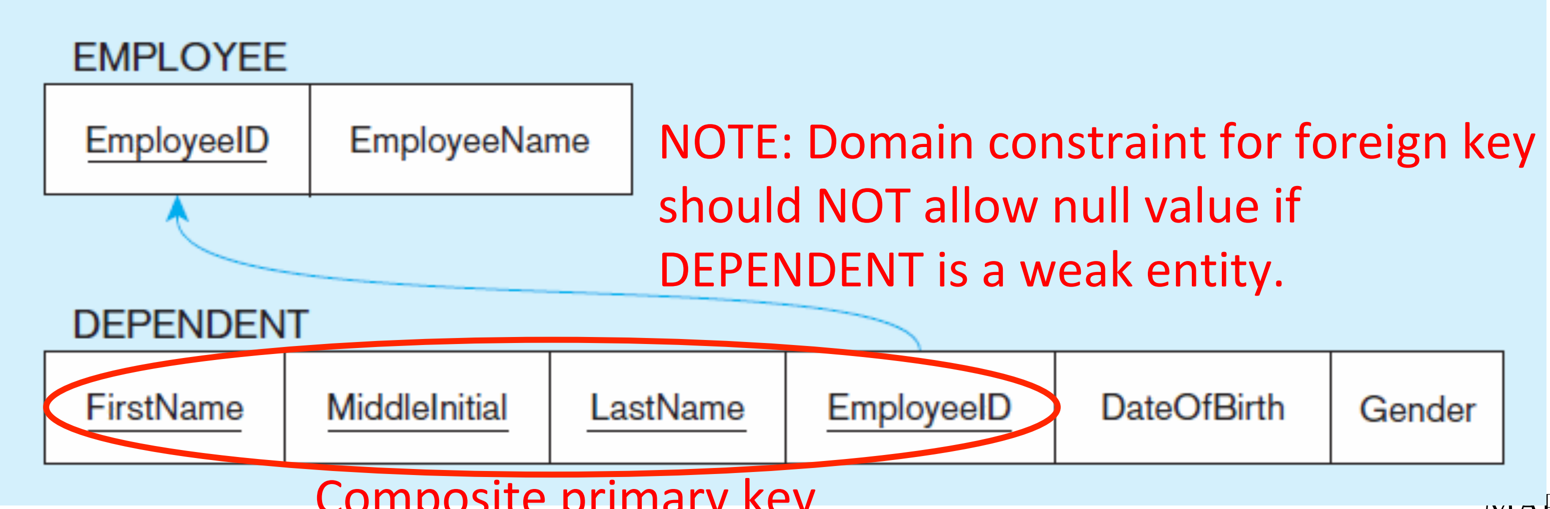
Figure 4-11 Example of Mapping a Weak Entity

(a) Weak entity DEPENDENT



Dependent
dependentName
-firstName
-middleInitial
-lastName
dateOfBirth
gender

(b) Weak entity becomes a separate relation with foreign key



NOTE: Domain constraint for foreign key should NOT allow null value if DEPENDENT is a weak entity.

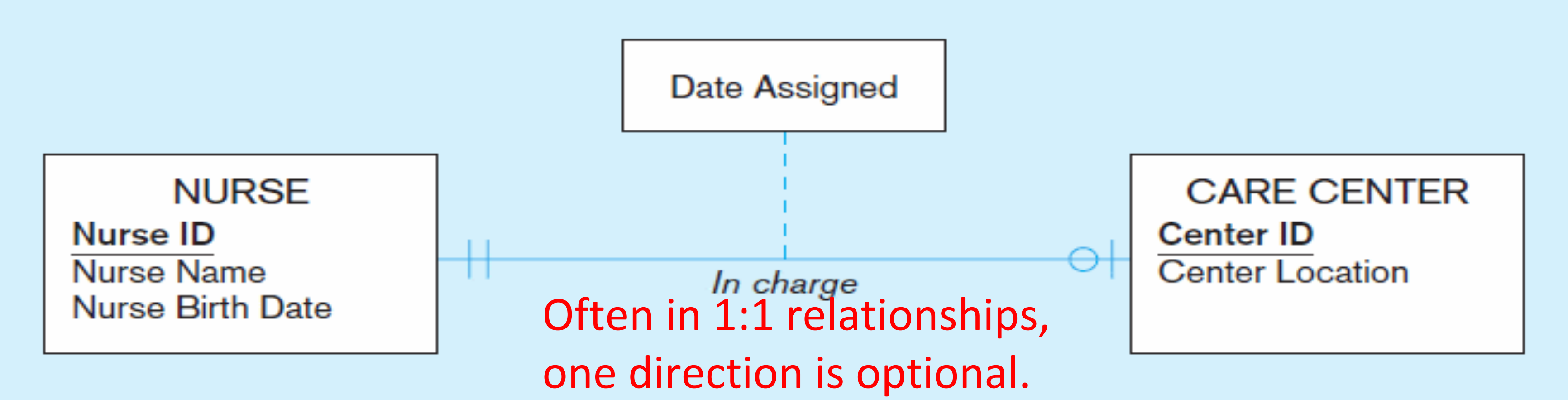
Composite primary key

Transforming ER Diagrams into Relations Step 3

- Mapping binary relationships:
  - **One-to-One** – Primary key on mandatory side becomes a foreign key on optional side.
  - **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 composite primary key.

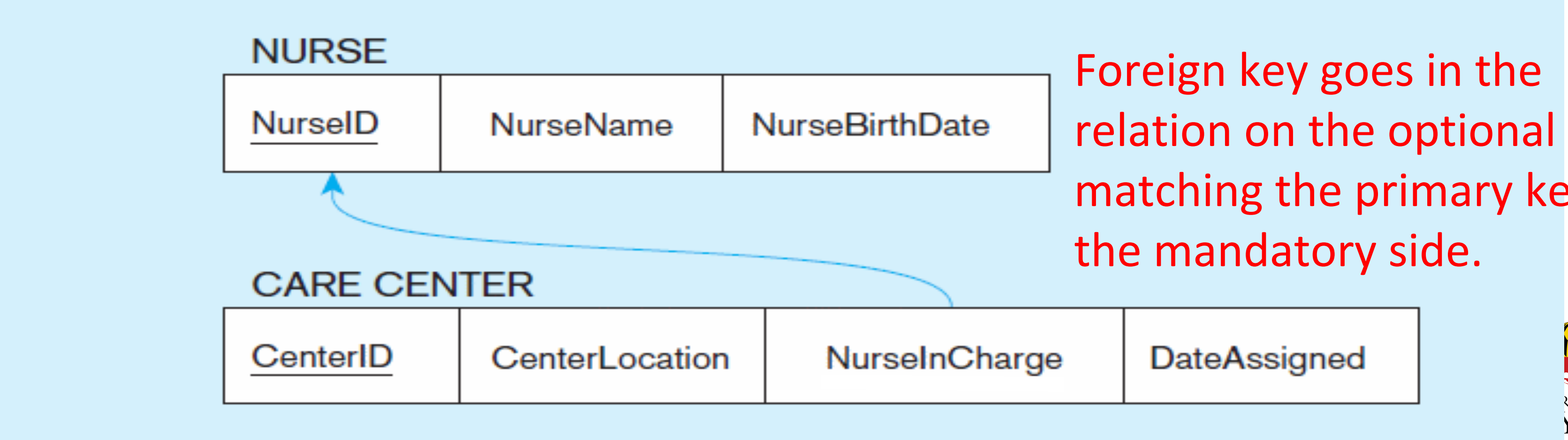
Figure 4-14: Example of Mapping a Binary 1:1 Relationship

(a) *In charge* relationship type with attribute Date Assigned



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

(b) Add foreign key and attribute to the optional entity type



NURSE

NurseID	NurseName	NurseBirthDate

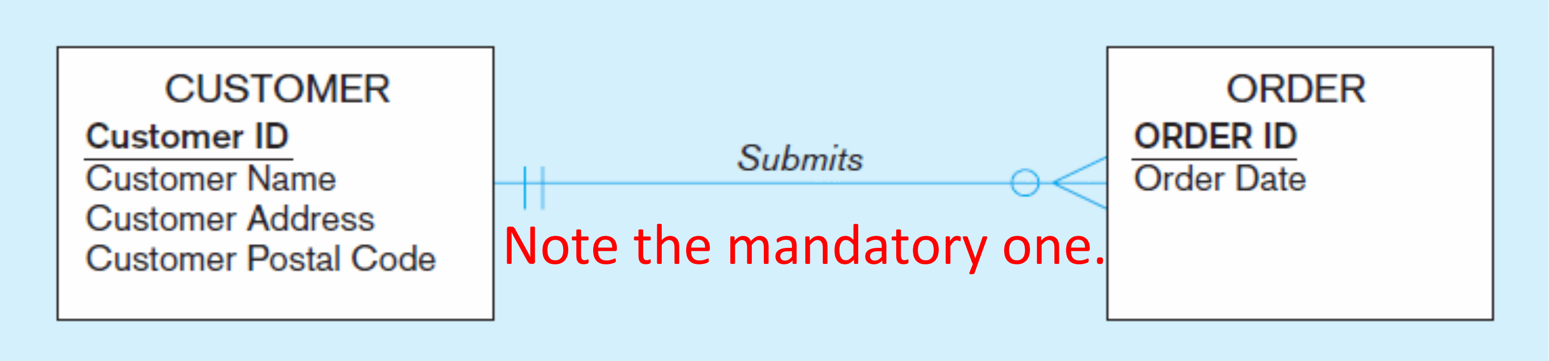
CARE CENTER

CenterID	CenterLocation	NurseInCharge	DateAssigned

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

Figure 4-12: Example of Mapping a Binary 1:M Relationship

(a) *Submits* relationship type between CUSTOMERS and ORDERS



CUSTOMER

Customer ID
Customer Name
Customer Address
Customer Postal Code

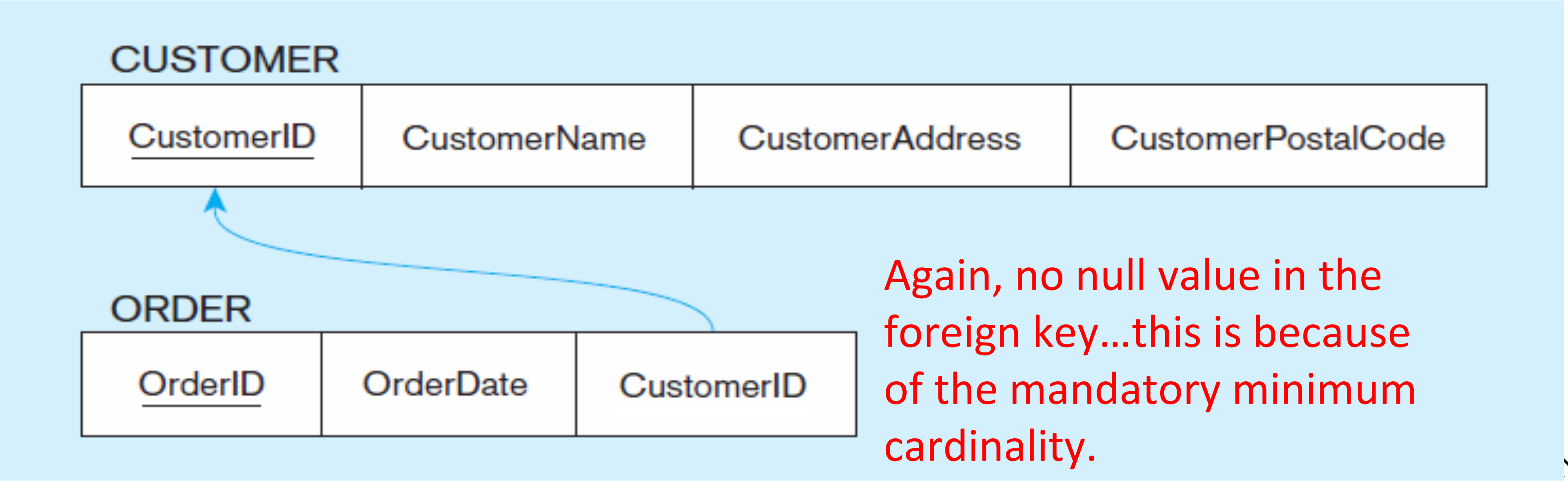
Submits

ORDER

ORDER ID
Order Date

Note the mandatory one.

(b) Add foreign key to the optional entity type



CUSTOMER

CustomerID	CustomerName	CustomerAddress	CustomerPostalCode

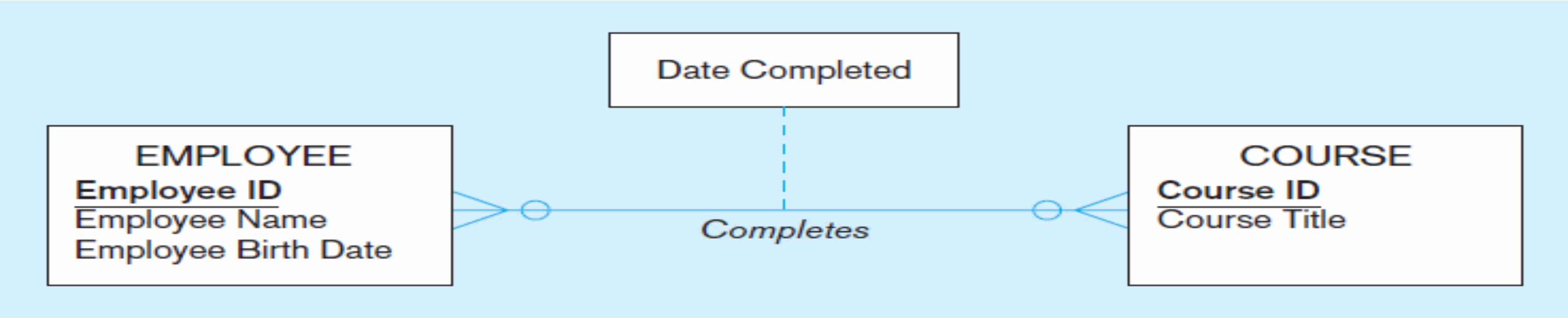
ORDER

OrderID	OrderDate	CustomerID

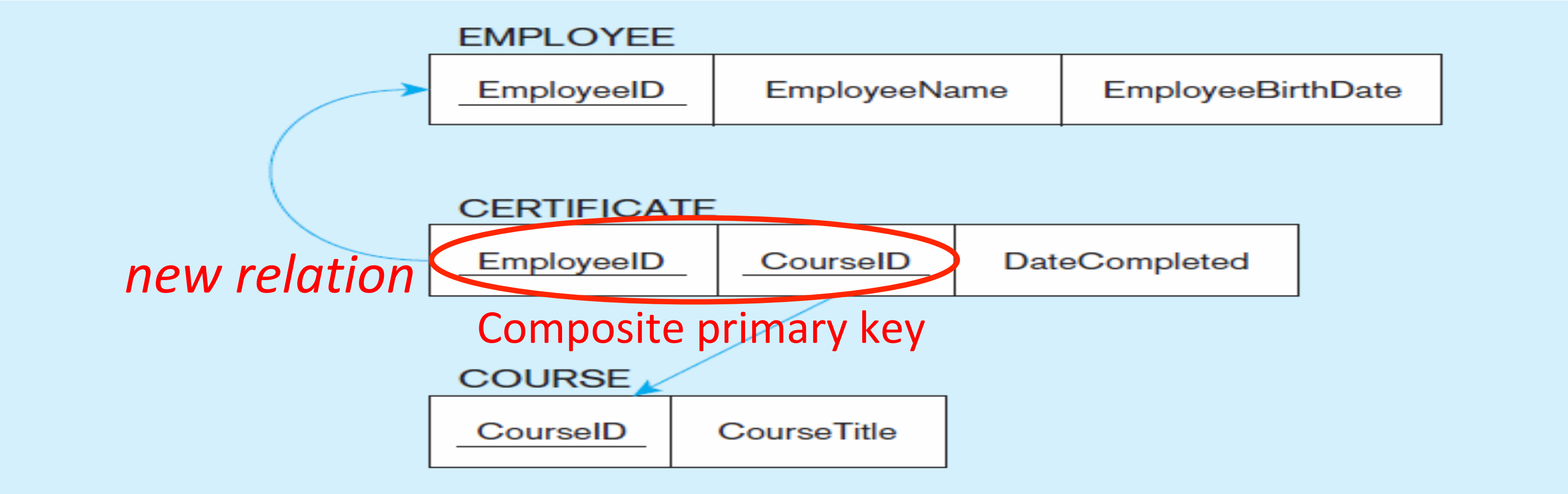
Again, no null value in the foreign key…this is because of the mandatory minimum cardinality.

Figure 4-13: Example of Mapping a Binary M:N Relationship

(a) *Completes* relationship type between EMPLOYEE and COURSE



(b) Create new relation with two foreign keys and attribute



EMPLOYEE

EmployeeID	EmployeeName	EmployeeBirthDate

CERTIFICATE

EmployeeID	CourseID	DateCompleted

COURSE

CourseID	CourseTitle

new relation

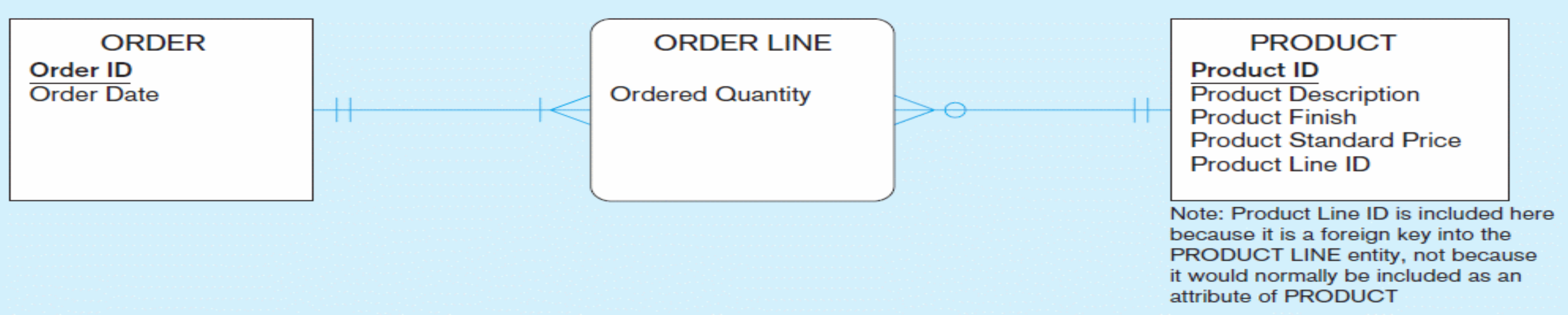
Composite primary key

Transforming ER Diagrams into Relations Step 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 *:* relationship).
  - Identifier Assigned:
    - It is natural and familiar to end-users.
    - Default identifier may not be unique.

Figure 4-15: Example of Mapping a Associative Entity

(a) Two strong entities and an associative entity



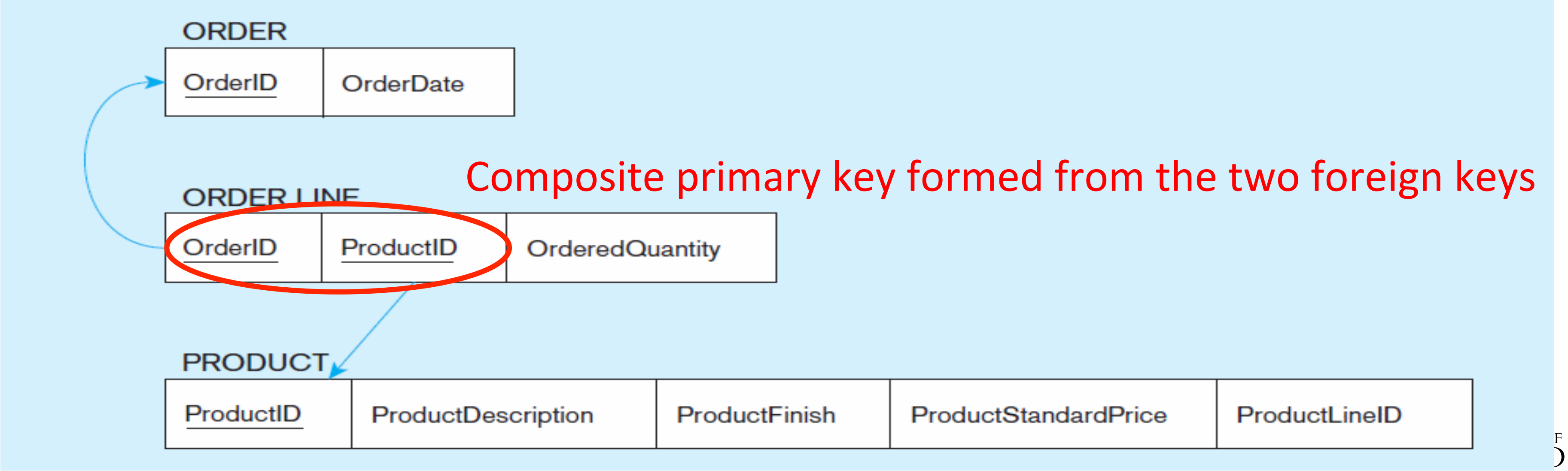
ORDER
Order ID
Order Date

ORDER LINE
Ordered Quantity

PRODUCT
Product ID
Product Description
Product Finish
Product Standard Price
Product Line ID

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



ORDER

OrderID	OrderDate

ORDER LINE

OrderID	ProductID	OrderedQuantity

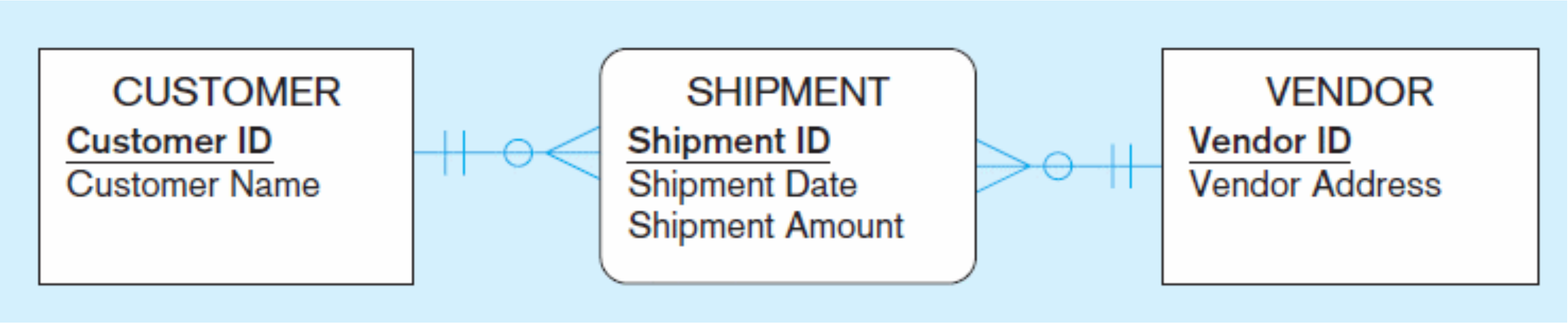
Composite primary key formed from the two foreign keys

PRODUCT

ProductID	ProductDescription	ProductFinish	ProductStandardPrice	ProductLineID

Figure 4-16: Example of Mapping a Associative Entity with an Identifier

(a) Two strong entities and an associative entity

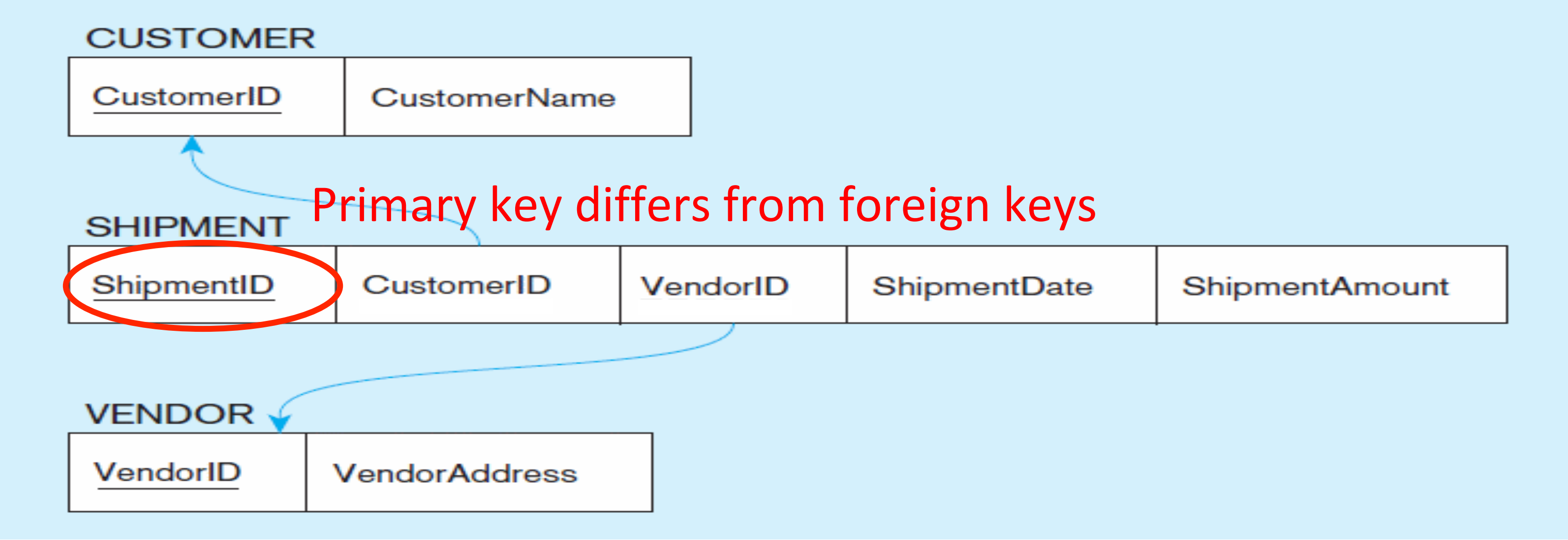


CUSTOMER
Customer ID
Customer Name

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

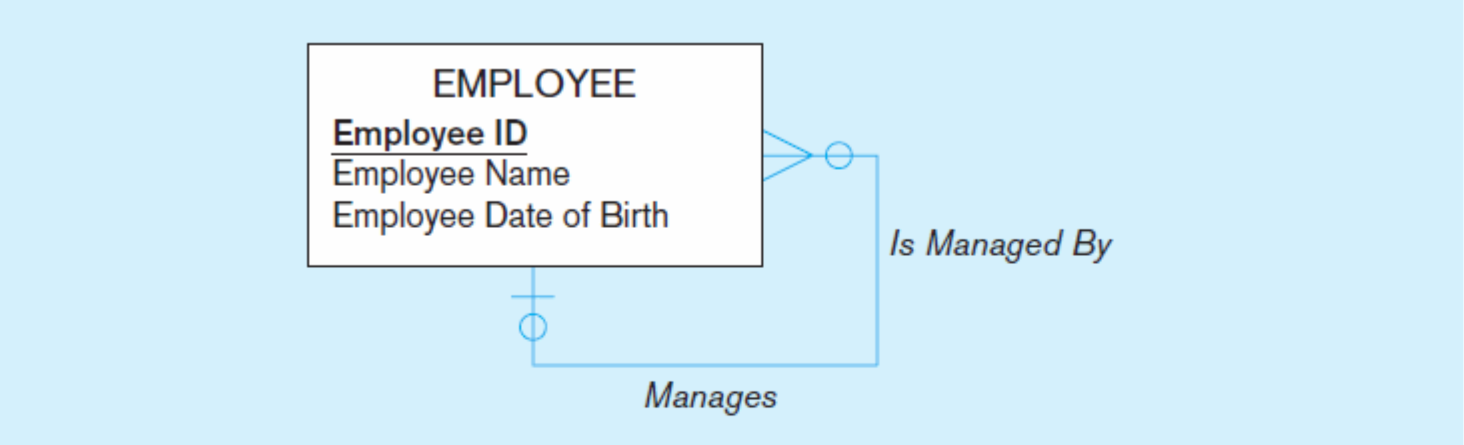
Primary key differs from foreign keys

Transforming ER Diagrams into Relations Step 5

- 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:N Relationship

(a) EMPLOYEE entity with unary relationship



EMPLOYEE

Employee ID

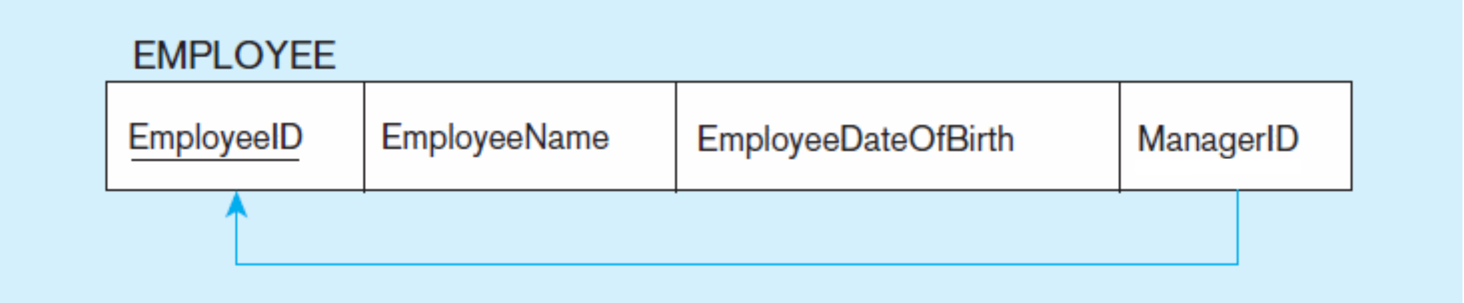
Employee Name

Employee Date of Birth

Is Managed By

Manages

(b) EMPLOYEE relation with recursive foreign key

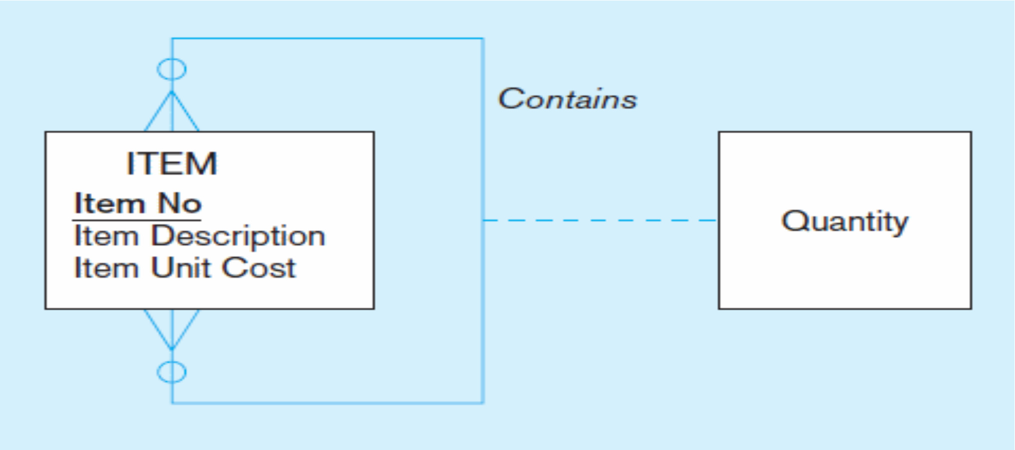


EMPLOYEE

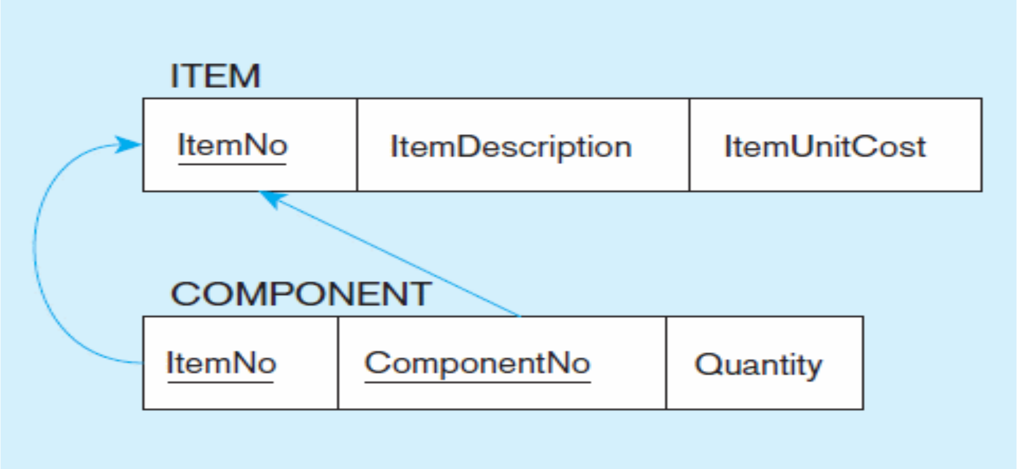
EmployeeID	EmployeeName	EmployeeDateOfBirth	ManagerID

Figure 4-18: Mapping a Unary M:N Relationship

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



(b) ITEM and COMPONENT relations



ITEM

ItemNo	ItemDescription	ItemUnitCost

COMPONENT

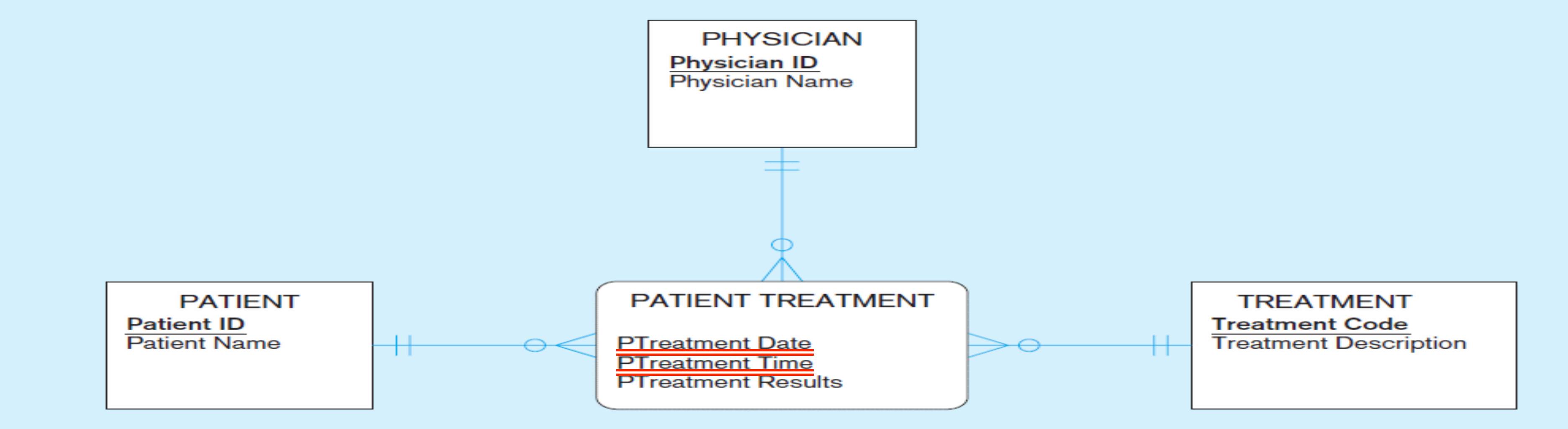
ItemNo	ComponentNo	Quantity

Transforming ER Diagrams into Relations Step 6

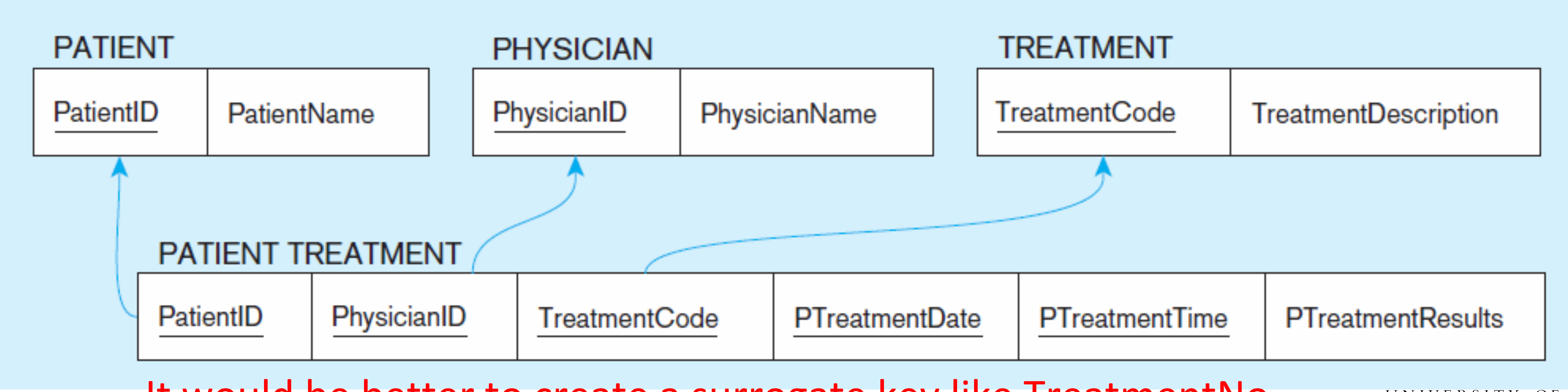
- Mapping ternary (and *n*-ary) relationships:
  - One relation for each entity and one for the associative entity.
  - Associative entity has foreign keys to each entity in the relationship.

Figure 4-19: Mapping a Ternary Relationship

(a) PATIENT TREATMENT ternary relationship with associative entity



(b) PATIENT TREATMENT relation with three foreign keys



PATIENT

PatientID	PatientName

PHYSICIAN

PhysicianID	PhysicianName

TREATMENT

TreatmentCode	TreatmentDescription

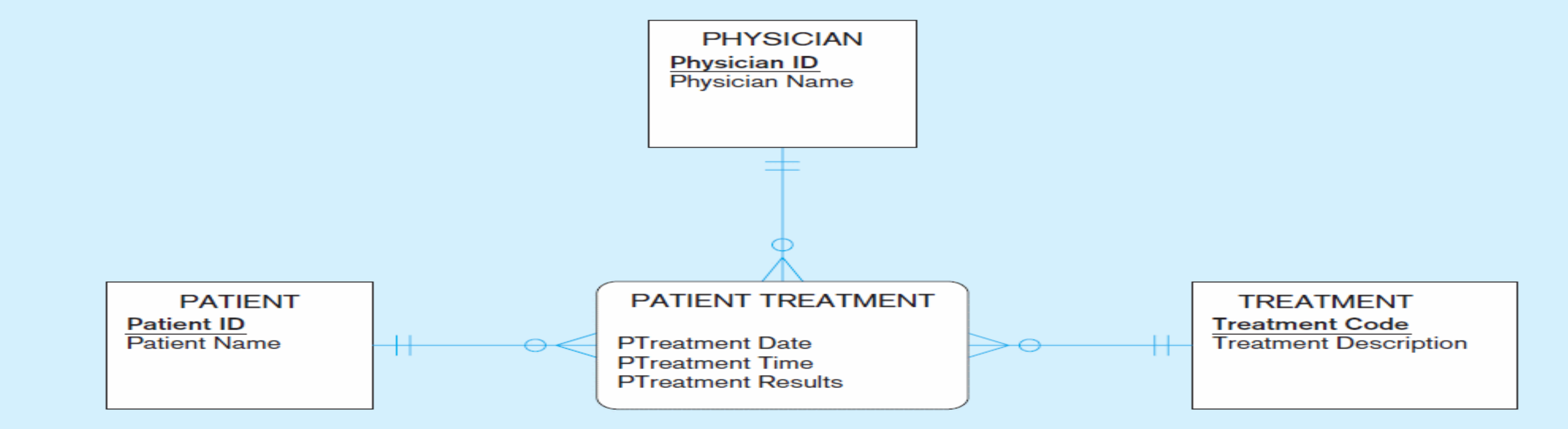
PATIENT TREATMENT

PatientID	PhysicianID	TreatmentCode	PTreatmentDate	PTreatmentTime	PTreatmentResults

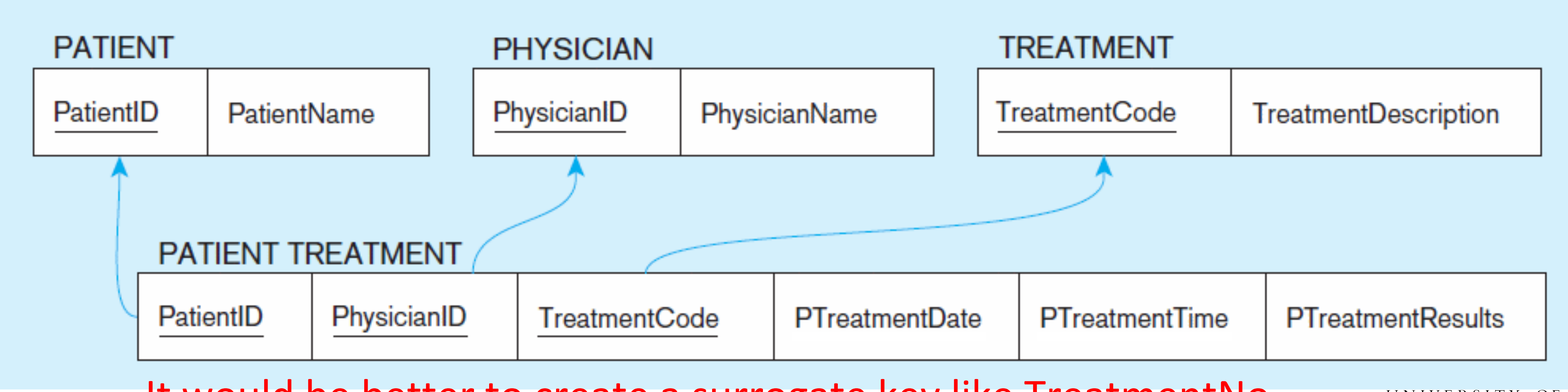
It would be better to create a surrogate key like TreatmentNo.

Figure 4-19: Mapping a Ternary Relationship

(a) PATIENT TREATMENT ternary relationship with associative entity



(b) PATIENT TREATMENT relation with three foreign keys



PATIENT

PatientID	PatientName

PHYSICIAN

PhysicianID	PhysicianName

TREATMENT

TreatmentCode	TreatmentDescription

PATIENT TREATMENT

PatientID	PhysicianID	TreatmentCode	PTreatmentDate	PTreatmentTime	PTreatmentResults

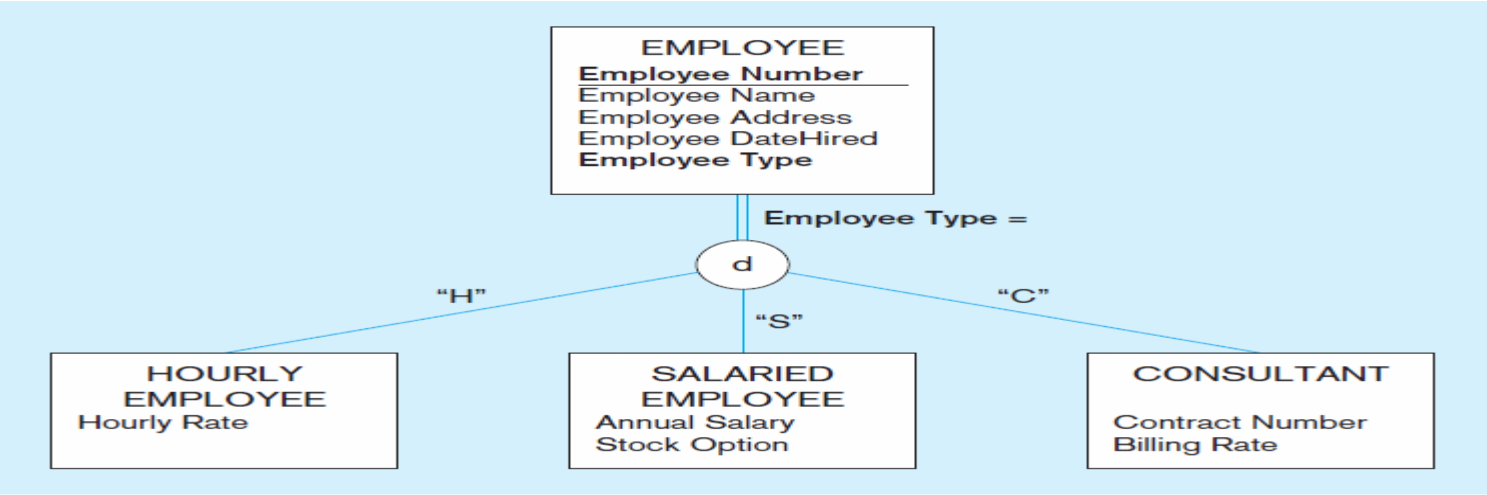
It would be better to create a surrogate key like TreatmentNo.

Transforming ER Diagrams into Relations Step 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.

Figure 4-20: Supertype/Subtype Relationships

(a) EMPLOYEE with three subtypes



EMPLOYEE
Employee Number
Employee Name
Employee Address
Employee DateHired
Employee Type

Employee Type =

d

"H"

"S"

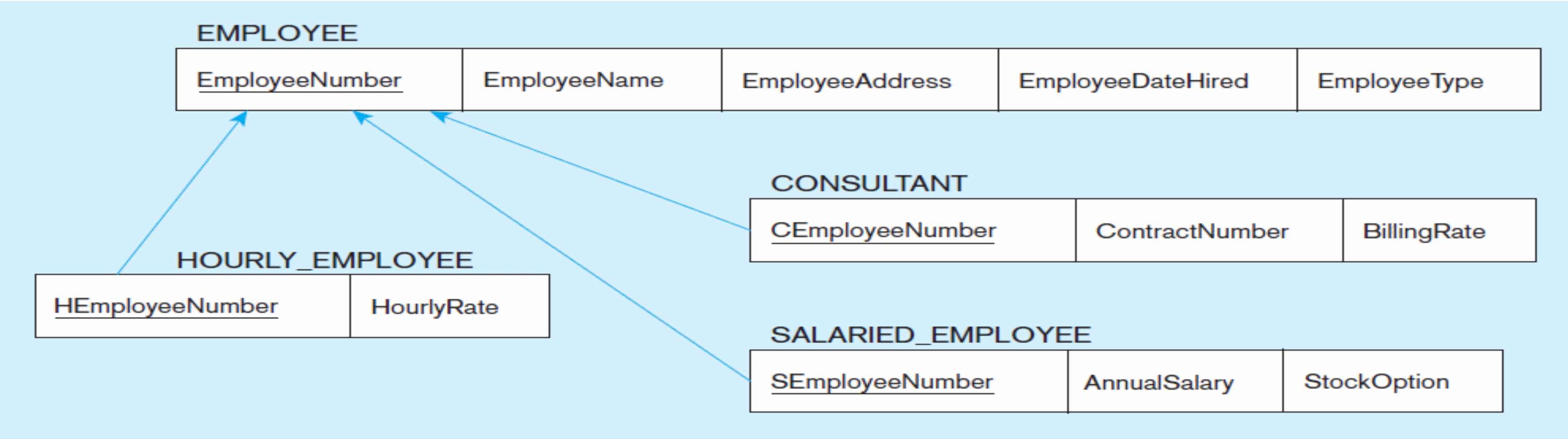
"C"

HOURLY EMPLOYEE
Hourly Rate

SALARIED EMPLOYEE
Annual Salary
Stock Option

CONSULTANT
Contract Number
Billing Rate

(b) These are implemented as one-to-one relationships



EMPLOYEE

EmployeeNumber	EmployeeName	EmployeeAddress	EmployeeDateHired	EmployeeType

CONSULTANT

CEmployeeNumber	ContractNumber	BillingRate

HOURLY_EMPLOYEE

HEmployeeNumber	HourlyRate

SALARIED_EMPLOYEE

SEmployeeNumber	AnnualSalary	StockOption

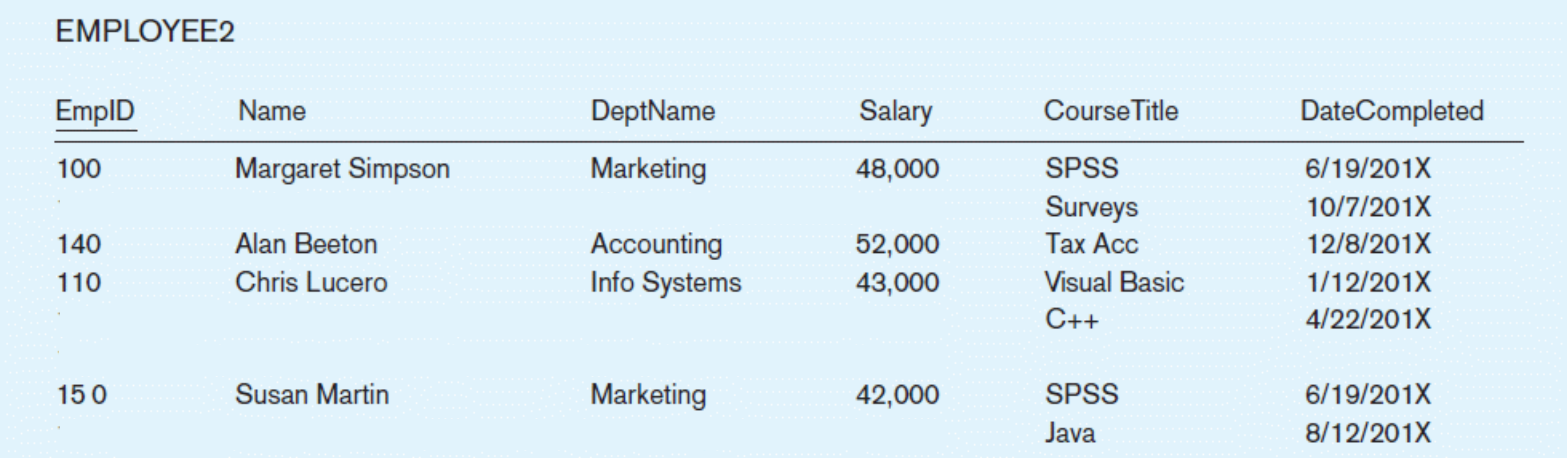
Data Normalization

- 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.

Figure 4-2: EMPLOYEE2



EMPLOYEE2

EmpID	Name	DeptName	Salary	CourseTitle	DateCompleted
100	Margaret Simpson	Marketing	48,000	SPSS	6/19/201X
				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
				C++	4/22/201X
15 0	Susan Martin	Marketing	42,000	SPSS	6/19/201X
				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: EmpID, CourseTitle?

Anomalies in This Table

- **Insertion** – Cannot enter a new employee without having the employee take a class (or at least empty fields of class information).
- **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.

Figure 4.22: Steps in Normalization

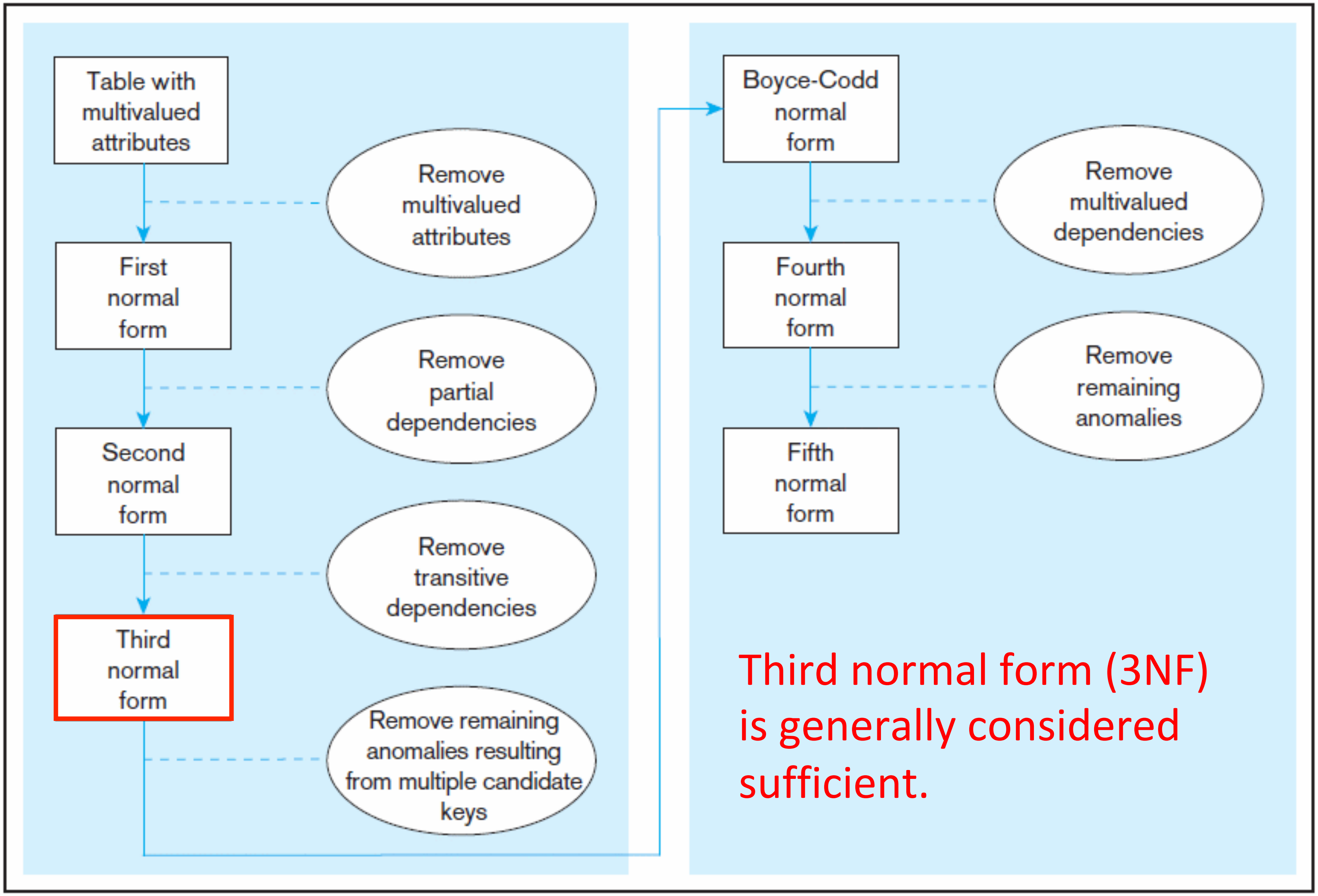


Table with multivalued attributes
Remove multivalued attributes
First normal form
Remove partial dependencies
Second normal form
Remove transitive dependencies
Third normal form
Remove remaining anomalies resulting from multiple candidate keys
Boyce-Codd normal form
Remove multivalued dependencies
Fourth normal form
Remove remaining anomalies
Fifth normal form

Third normal form (3NF) is generally considered sufficient.

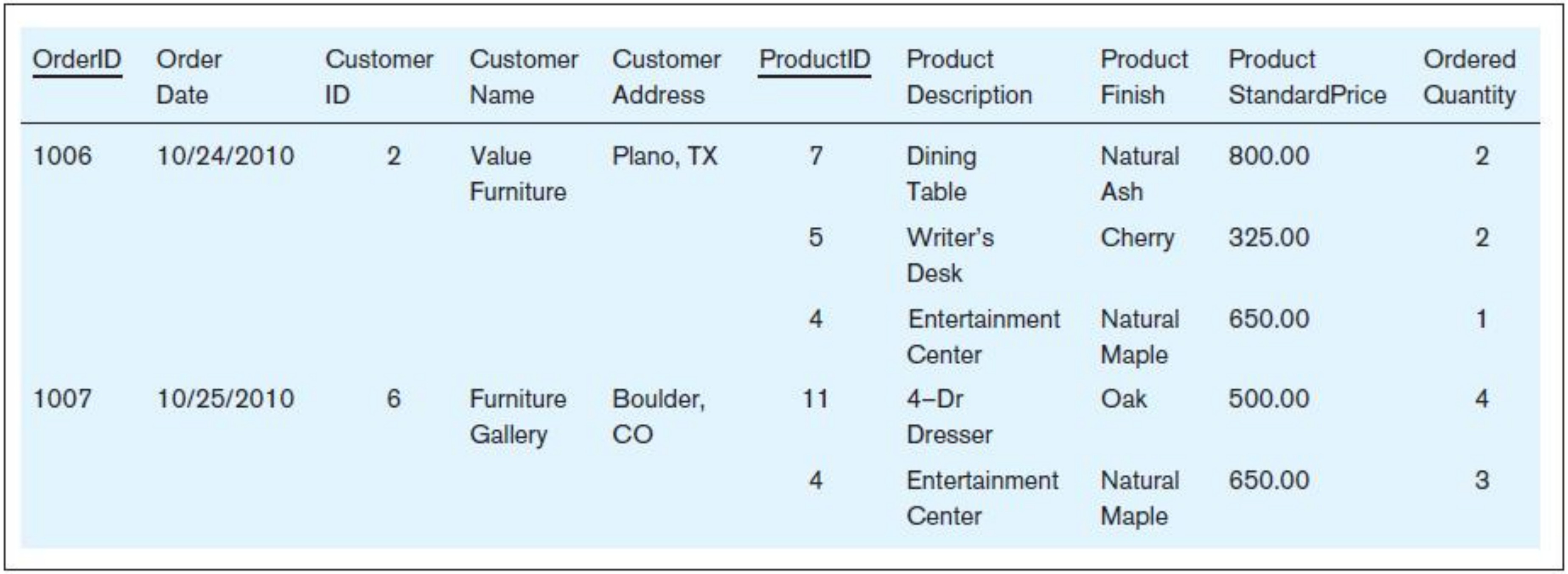
Functional Dependencies and Keys

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

First Normal Form (1NF)

- No multivalued attributes.
- Every attribute value is atomic.
- Fig. 4-25 is not in 1NF (multivalued attributes).
  - i.e. It is not a relation.
- Fig. 4-26 is in 1NF.
- All relations are in 1NF.

Figure 4-25: Table with Multivalued Attributes, not in 1NF

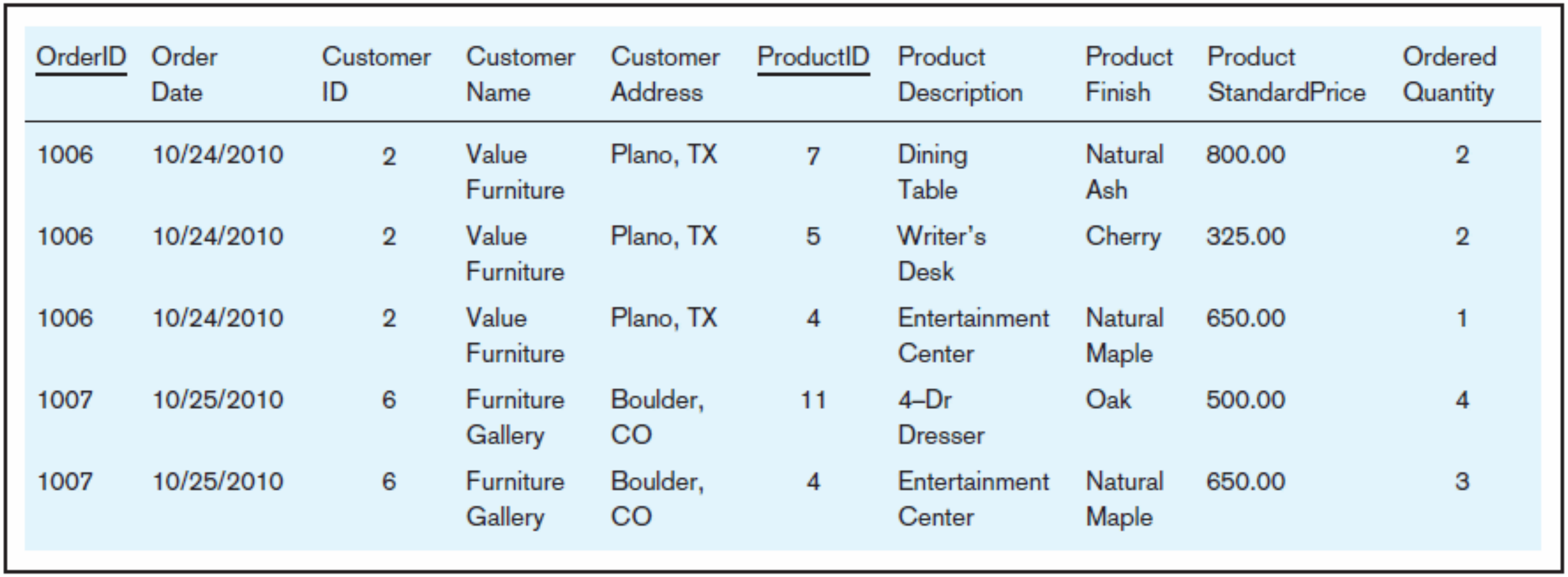


OrderID	Order Date	Customer ID	Customer Name	Customer Address	ProductID	Product Description	Product Finish	Product StandardPrice	Ordered Quantity
1006	10/24/2010	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
					5	Writer's Desk	Cherry	325.00	2
					4	Entertainment Center	Natural Maple	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

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

NOTE: This is NOT a relation.

Figure 4-26: Table with No Multivalued Attributes and Unique Rows, in 1NF



OrderID	Order Date	Customer ID	Customer Name	Customer Address	ProductID	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

FIGURE 4-26 **INVOICE relation (1NF) (Pine Valley Furniture Company)**

NOTE: This is a relation, but not a well-structured one.

Anomalies in This Table

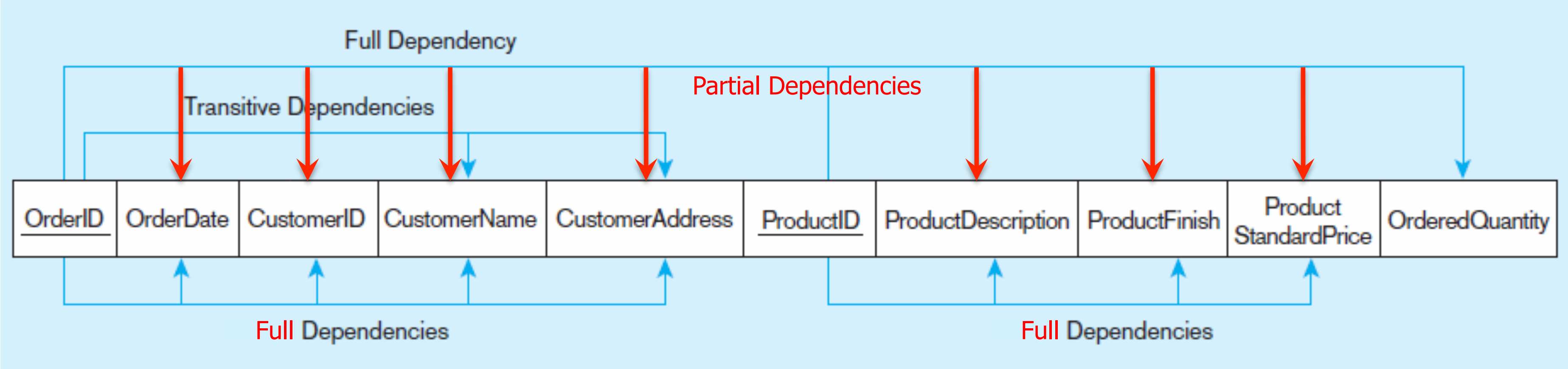
- **Insertion** – If new product is ordered for order 1007 of existing customer, customer data must be re-entered, causing duplication.
- **Deletion** – If we delete the Dining Table from Order 1006, we lose information concerning this item's finish and price.
- **Modification** – Changing the price of product ID 4 requires update in multiple 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 (2NF)

- 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.

Figure 4-27: Functional Dependency Diagram

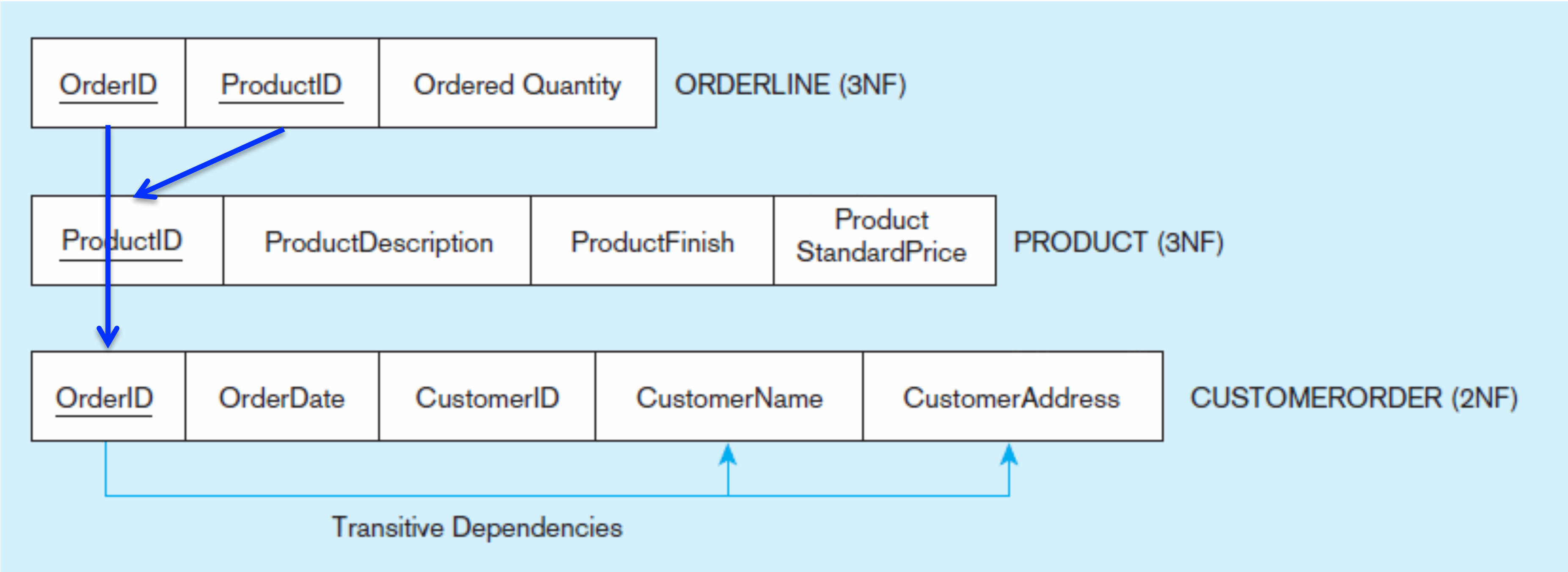


- OrderID → OrderDate, CustomerID, CustomerName, CustomerAddress
- CustomerID → CustomerName, CustomerAddress
- ProductID → ProductDescription, ProductFinish, ProductStandardPrice
- OrderID, ProductID → OrderQuantity, OrderDate, CustomerID, CustomerName, CustomerAddress, ProductDescription, ProductFinish, ProductStandardPrice

Therefore, NOT in 2NF

UNIVERSITY OF MARYLAND

Figure 4-28: Removing Partial Dependencies



OrderID | ProductID | Ordered Quantity — ORDERLINE (3NF)

ProductID | ProductDescription | ProductFinish | Product StandardPrice — PRODUCT (3NF)

OrderID | OrderDate | CustomerID | CustomerName | CustomerAddress — CUSTOMERORDER (2NF)

Transitive Dependencies

- OrderID, ProductID → OrderQuantity
- ProductID → ProductDescription, ProductFinish, ProductStandardPrice
- OrderID → OrderDate, CustomerID, CustomerName, CustomerAddress

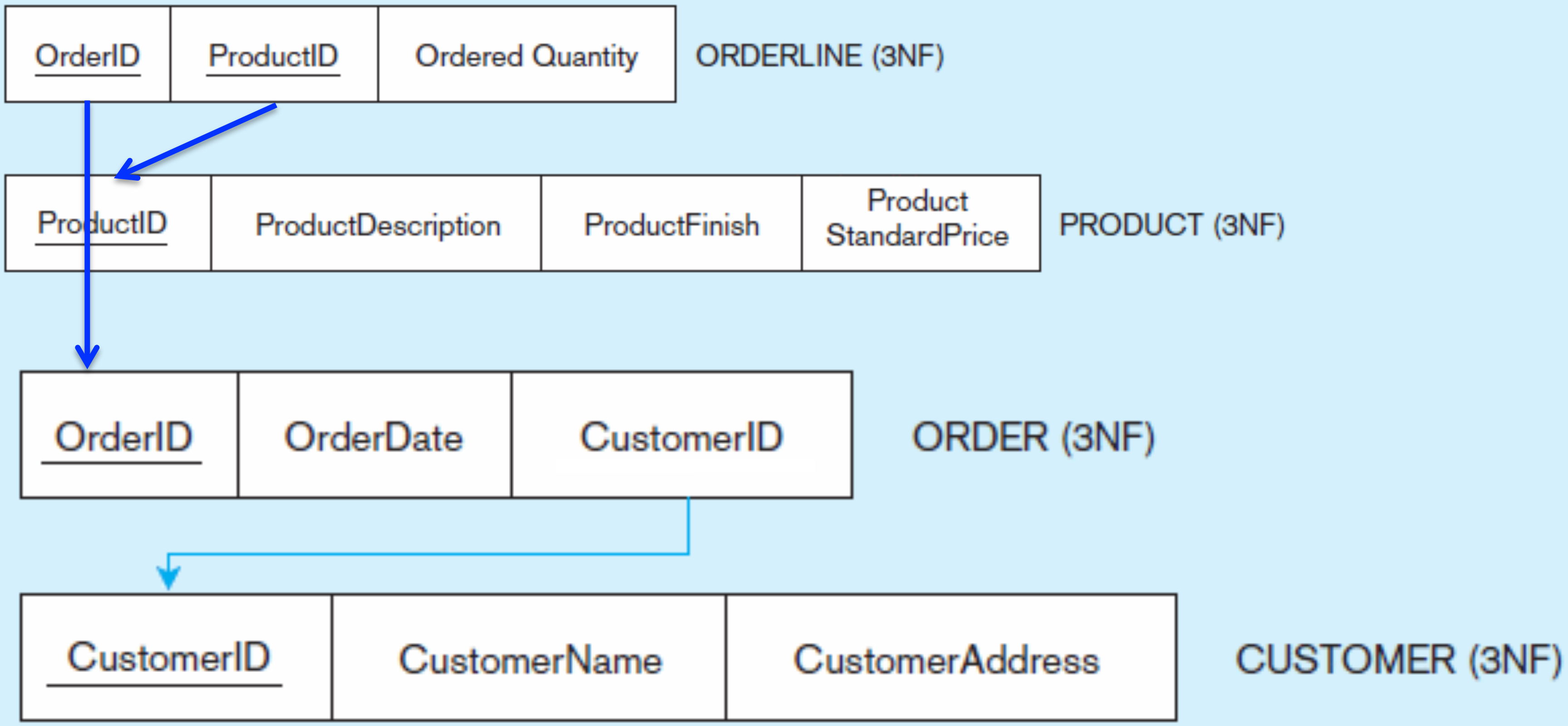
But there are still transitive dependencies.

- CustomerID → CustomerName, CustomerAddress

Third Normal Form (3NF)

- 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.

Figure 4-29: Removing Transitive Dependencies



OrderID	ProductID	Ordered Quantity

ORDERLINE (3NF)

ProductID	ProductDescription	ProductFinish	Product StandardPrice

PRODUCT (3NF)

OrderID	OrderDate	CustomerID

ORDER (3NF)

CustomerID	CustomerName	CustomerAddress

CUSTOMER (3NF)

OrderID, ProductID → OrderQuantity

ProductID → ProductDescription, ProductFinish, ProductStandardPrice

OrderID → OrderDate, CustomerID

CustomerID → CustomerName, CustomerAddress

Merging Relations

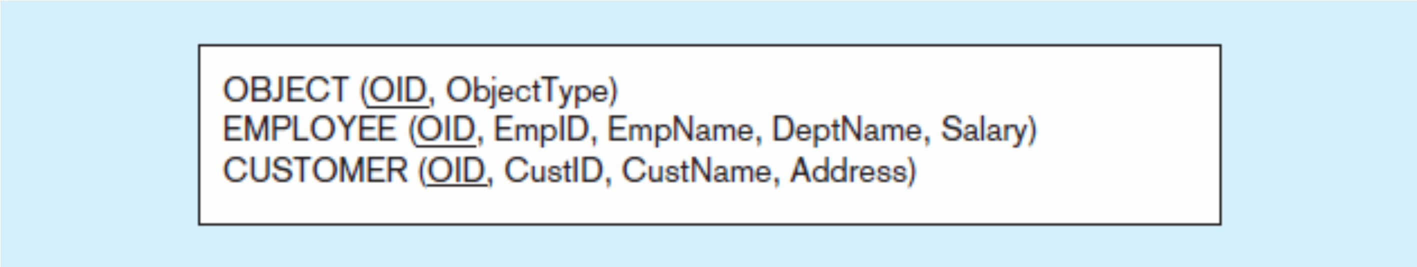
- **View Integration** – Combining entities from multiple ER models into common relations.
- Issues to watch out for when merging entities from different ER models:
  - **Synonyms** – Two or more attributes with different names but same meaning.
  - **Homonyms** – Attributes with same name but different 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.

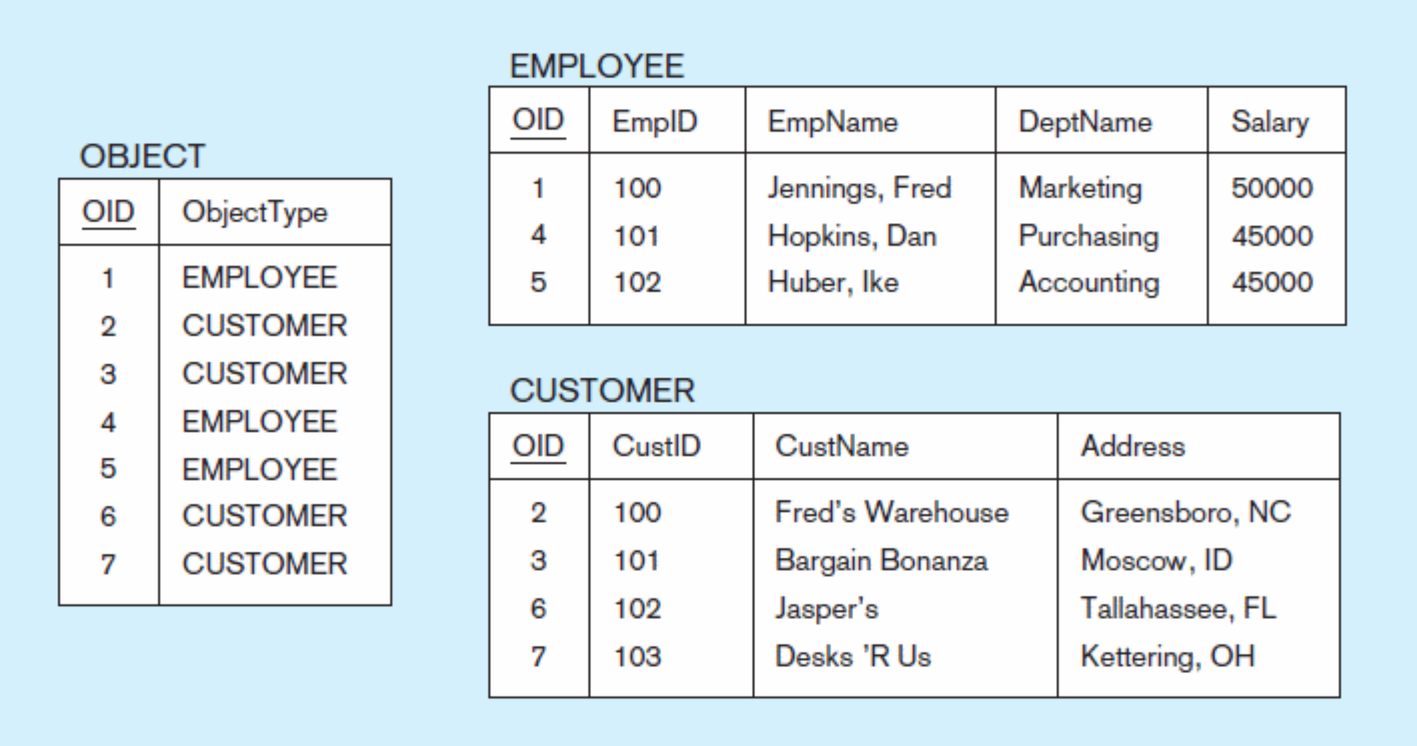
Figure 4-31: Enterprise Keys

(a) Relations with enterprise key



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

(b) Sample data with enterprise key



OBJECT

OID	ObjectType
1	EMPLOYEE
2	CUSTOMER
3	CUSTOMER
4	EMPLOYEE
5	EMPLOYEE
6	CUSTOMER
7	CUSTOMER

EMPLOYEE

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

CUSTOMER

OID	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