# Database Design

(Dealing with logical and physical schemas)

- Chapter 4
  Logical Database Design and the Relational Model
- Chapter 5
  Physical Database Design and Performance

# MDBM-Chapter 4

# Logical Database Design and the Relational Model

# Objectives

- Understand concept and properties of **relations**
- Learn properties of candidate keys
- Define first, second, and third **normal forms**
- Describe problems from merging relations
- Transform E-R and EER diagrams to relations
- Create tables with entity and relational integrity constraints
- Use **normalization** to convert anomalous tables to well-structured relations

# Logical Database Design

- It is a process of transforming the conceptual data model (E-R model) into logical data model.
- It creates stable database structures by expressing the requirements in a technical language.
- The *relational data model* is a form of logical data model, the one that is most commonly used in the contemporary database applications.

# Three components of the Relational Data Model

- **Data structure:** Data are organized in the form of tables, or relations.
- **Data manipulation:** Data are manipulated by the SQL language.
- **Data integrity:** Having mechanisms to maintain data integrity when manipulated.

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

# An Example of a Relation

EMPLOYEE1

| EmpID | Name | DeptName | Salary |
|---|---|---|---|
| 100 | Margaret Simpson | Marketing | 48,000 |
| 140 | Allen Beeton | Accounting | 52,000 |
| 110 | Chris Lucero | Info Systems | 43,000 |
| 190 | Lorenzo Davis | Finance | 55,000 |
| 150 | Susan Martin | Marketing | 42,000 |

# 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.

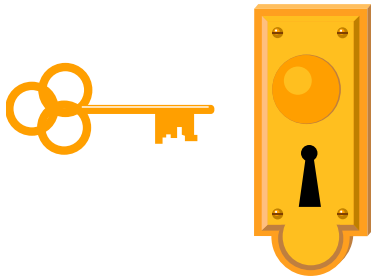
- NOTE: The word ***relation*** (in relational database) is NOT the same as the word ***relationship*** (in E-R model).

# Relation, Schema and Database

- A (relational) database may contain any number of relations.
- The structure of a relational database is described with schemas (a collection of DB objects, e.g. relations).
- Schemas are expressed by the following methods
  - Short text statements using the format of relation name followed by its attributes in parentheses, e.g., *EMPLOYEE1(EmpID, Name, DeptName, Salary).*
  - A graphical representation using a rectangle containing the attributes of a relation, e.g.,

EMPLOYEE1

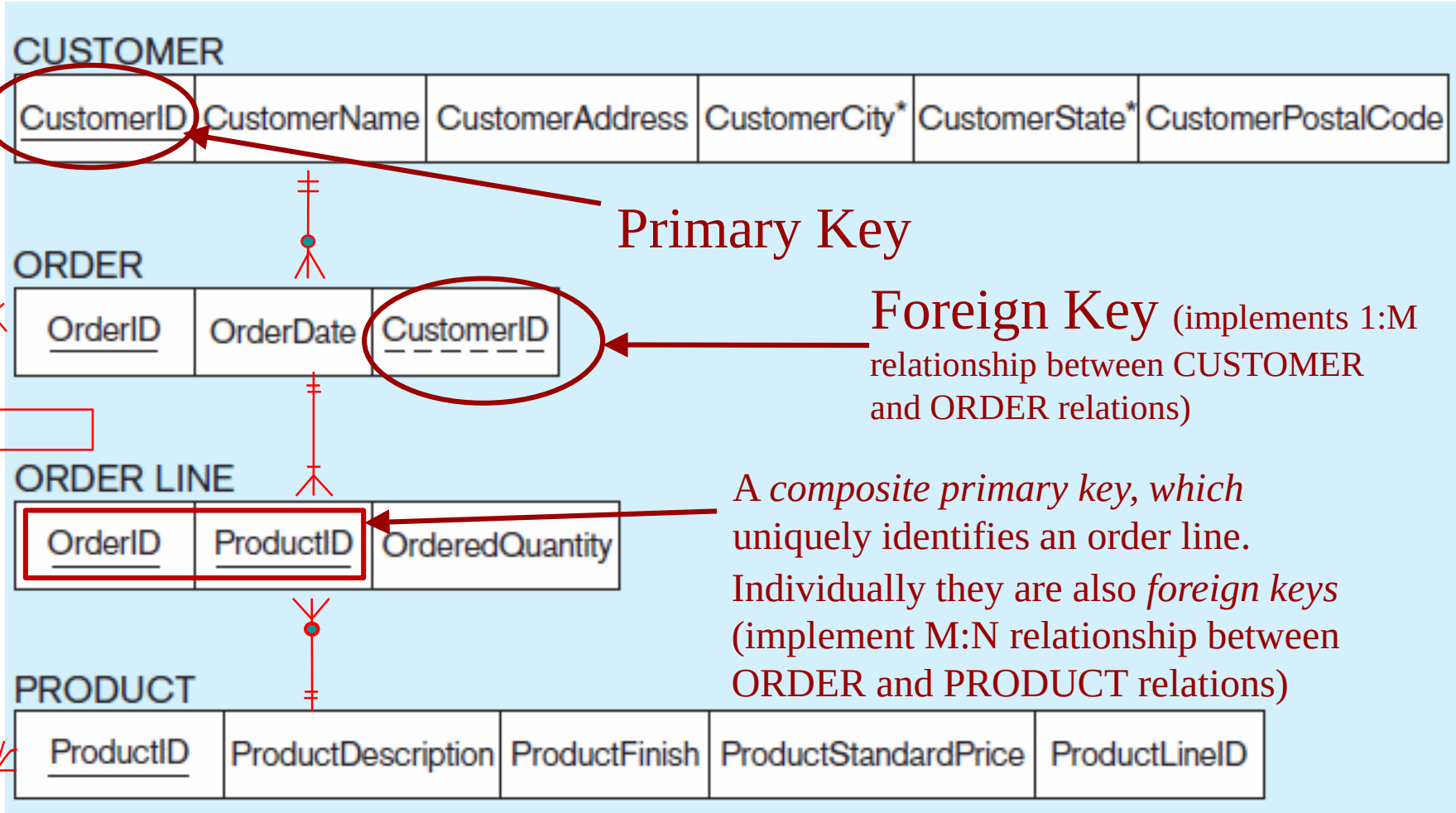
| **EmpID** | **Name** | **DeptName** | **Salary** |
|---|---|---|---|



# Key Fields

- Keys are special fields in a relation that serve two main purposes:
  - ***Primary key (PK)****:* an attribute or a combination of attributes (composite key) that uniquely identifies each row in a relation. It is designated by underlining the attribute name(s).
  - ***Foreign key (FK):*** an attribute in a relation that serves the primary key of another relation to represent the relationship of the relations, which is designated by a dashed underline in the relational schema. A foreign key enables a dependent relation to refer to its parent relation.

# Schema for four relations (PVFC)



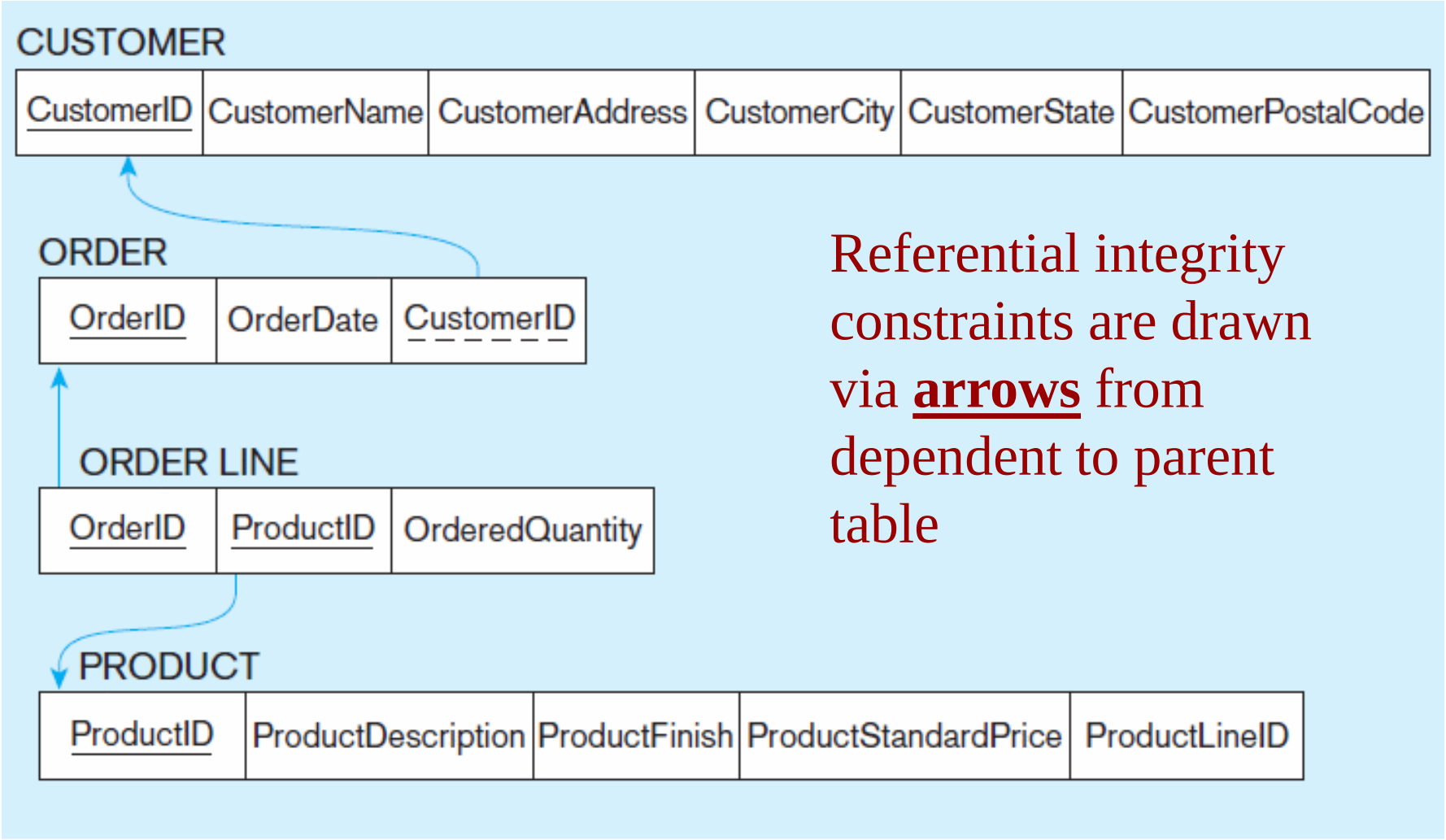
# Integrity Constraints of the Relational Data Model

- **Domain Constraints**
  - Allowable values for an attribute that are from the same domain. (See Table 4-1)
- **Entity Integrity Rule**
  - No primary key attribute (or its components) may be null, that is, all primary key fields MUST have data.
- **Referential Integrity**
  - Each foreign key value MUST match a primary key value in another relation or the foreign key value MUST be null.

# More on Referential Constraint

- **Graphical representation:** an arrow from the foreign key to the associated primary key
- **When can a FK be NULL?** only when the relationship is optional
- **Delete Rules:**
  - Prohibit delete: does not allow to delete the "parent" side (relation that the FK points to) if any related rows exist in the "dependent" side
  - Cascade delete: 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

# Referential integrity constraints (PVFC)




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

# Reinforcement of referential integrity constraint in 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

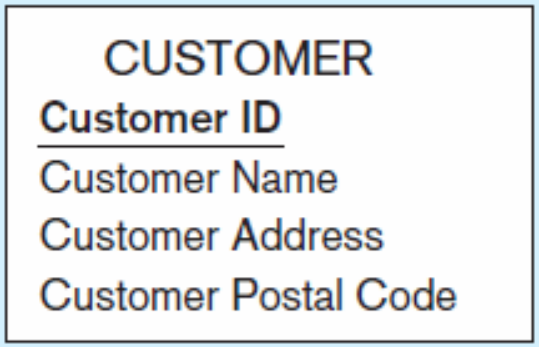
# Transforming EER Diagrams into Relations

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**



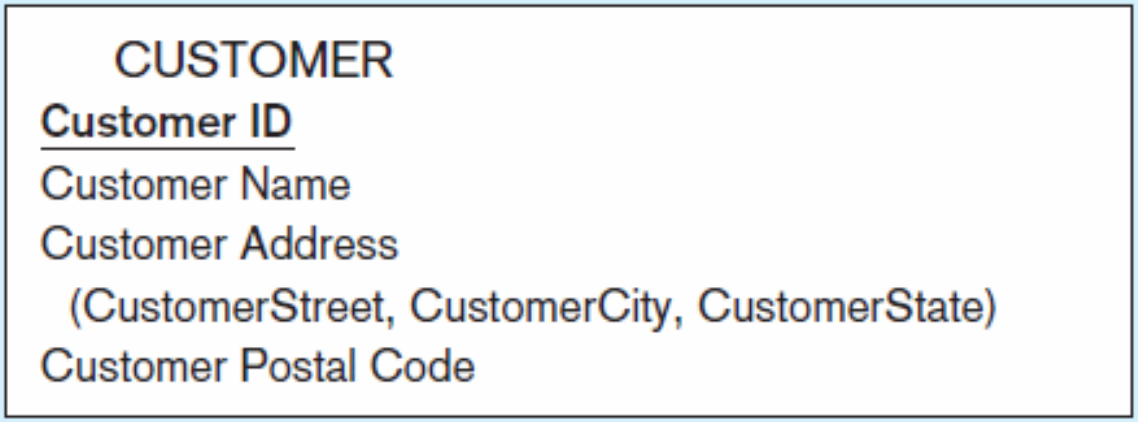

**(b) CUSTOMER relation**

CUSTOMER

| CustomerID | CustomerName | CustomerAddress | CustomerPostalCode |
|---|---|---|---|

# Mapping a composite attribute

(a) CUSTOMER entity type with composite attribute



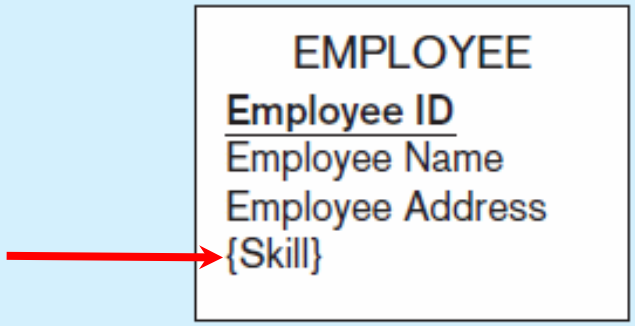

(b) CUSTOMER relation with address detail

CUSTOMER

| CustomerID | CustomerName | CustomerStreet | CustomerCity | CustomerState | CustomerPostalCode |
|---|---|---|---|---|---|

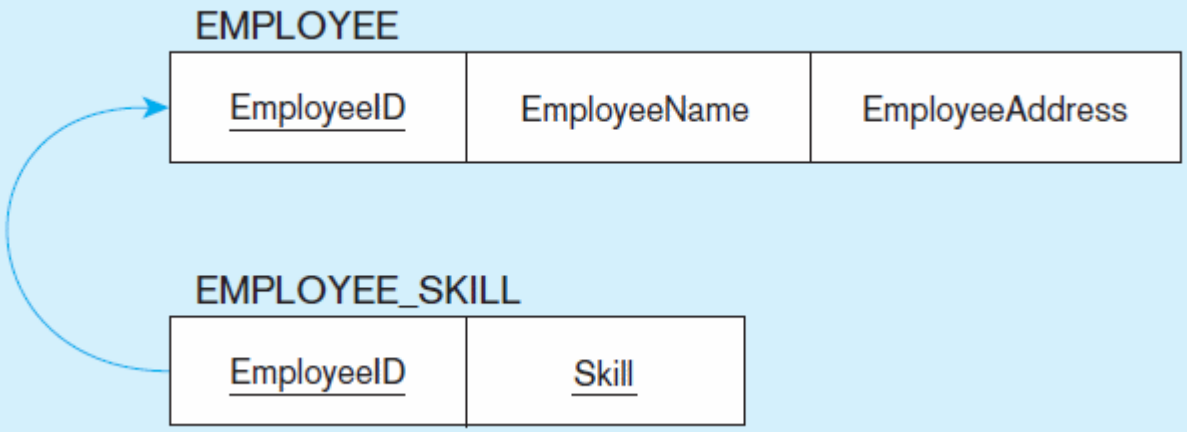
# Mapping an entity with a multivalued attribute

(a)




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

(b)




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

# Transforming EER Diagrams into Relations (cont.)

## Mapping a **Weak Entity**

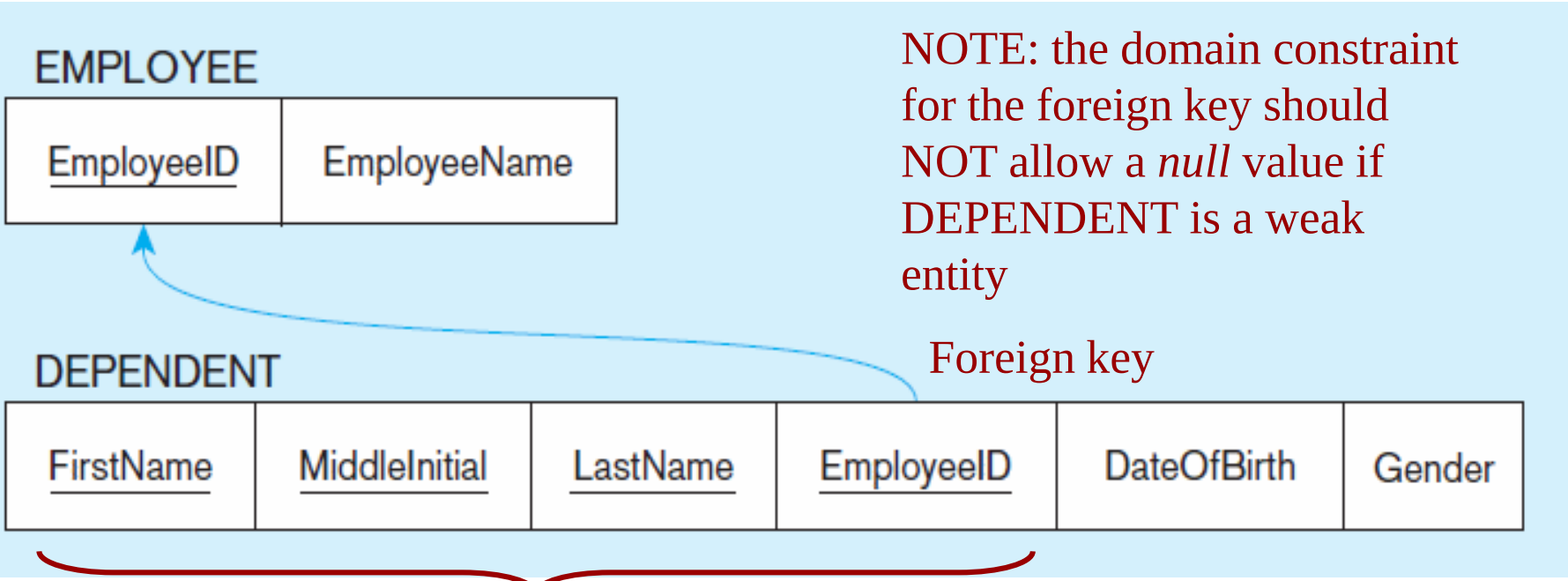
- Becomes a separate relation with a foreign key taken from the superior entity
- Primary key composed of:
  - Partial identifier of weak entity
  - Primary key of identifying relation (strong entity)

# Example of mapping a weak entity

a) Weak entity DEPENDENT

# Example of mapping a weak entity (cont.)

b) Relations resulting from a weak entity




| Dependent# | EmployeeID | FirstName | MiddleInitial | LastName | DateOfBirth | Gender |
|---|---|---|---|---|---|---|

Surrogate primary key: it is often a serial number or other system-assigned primary key to simplify the situation

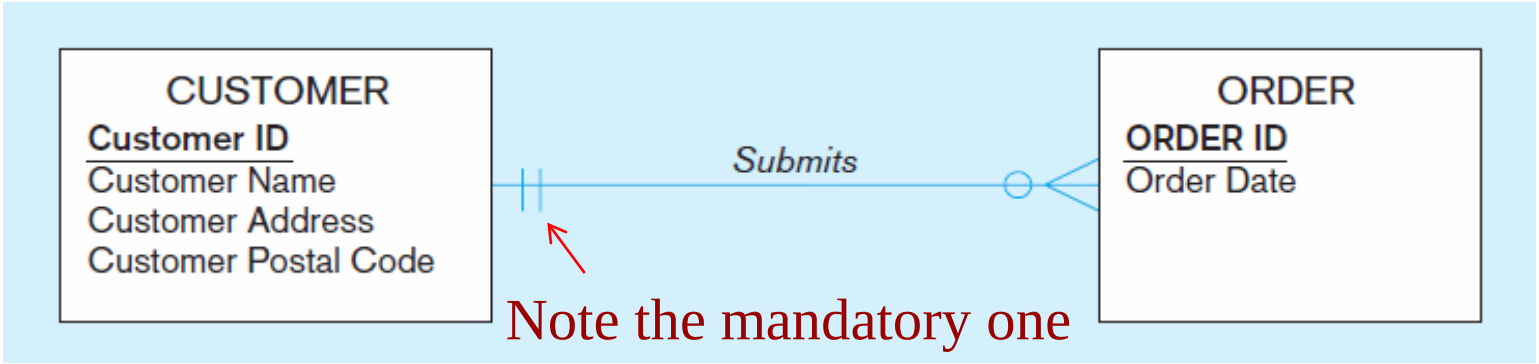
# Transforming EER Diagrams into Relations (cont.)

Mapping Binary Relationships (depending on the *degree* and *cardinality*)

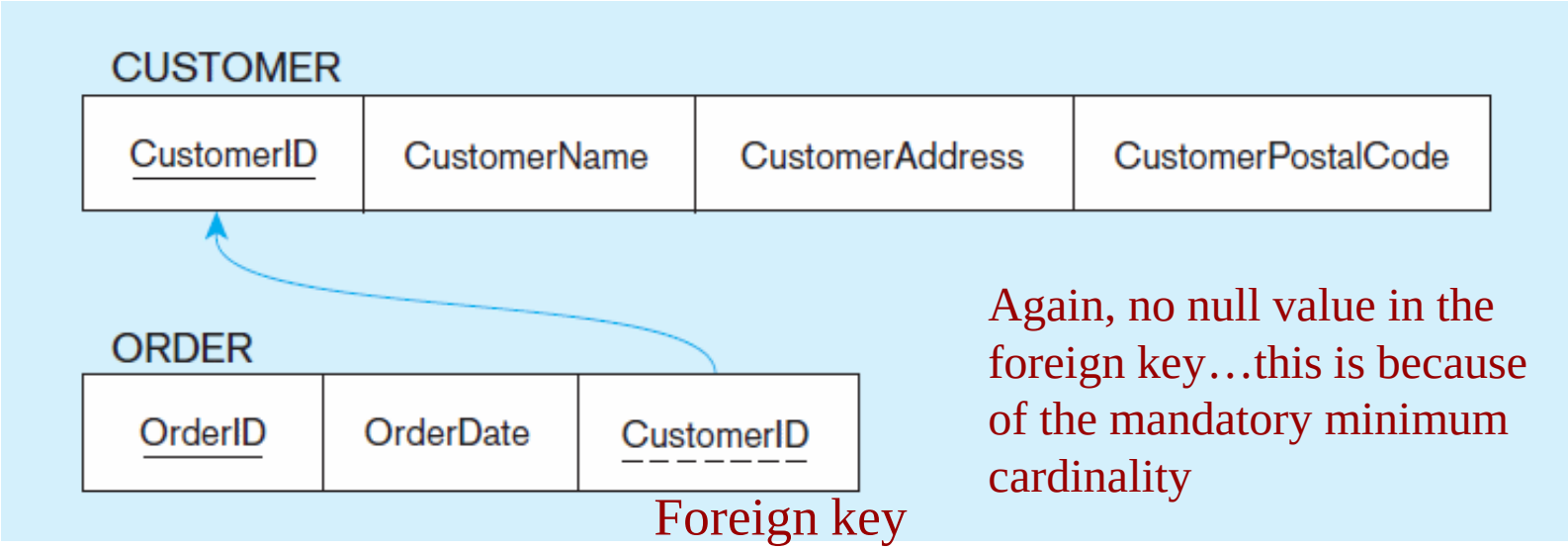
- One-to-Many: Primary key on the one side becomes a foreign key on the many side
- Many-to-Many: Create a ***new relation*** with the primary keys of the two entities as its primary key
- One-to-One: Primary key on the mandatory side becomes a foreign key on the optional side

# Example of mapping a 1:M relationship

a) Relationship between customers and orders

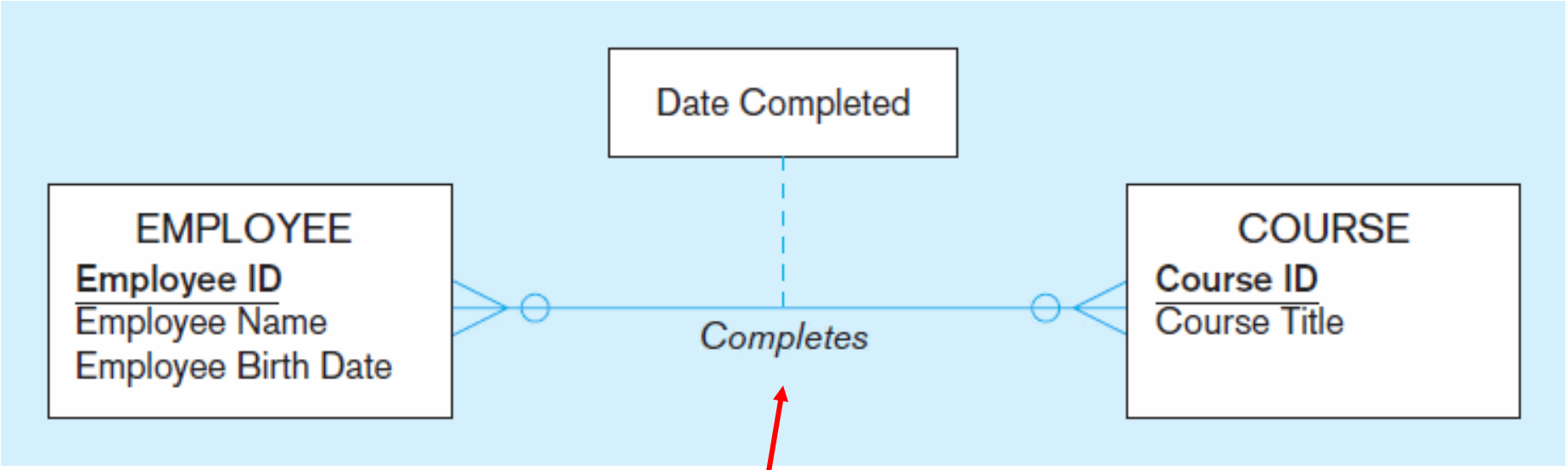



b) Mapping the relationship



# Example of mapping an M:N relationship

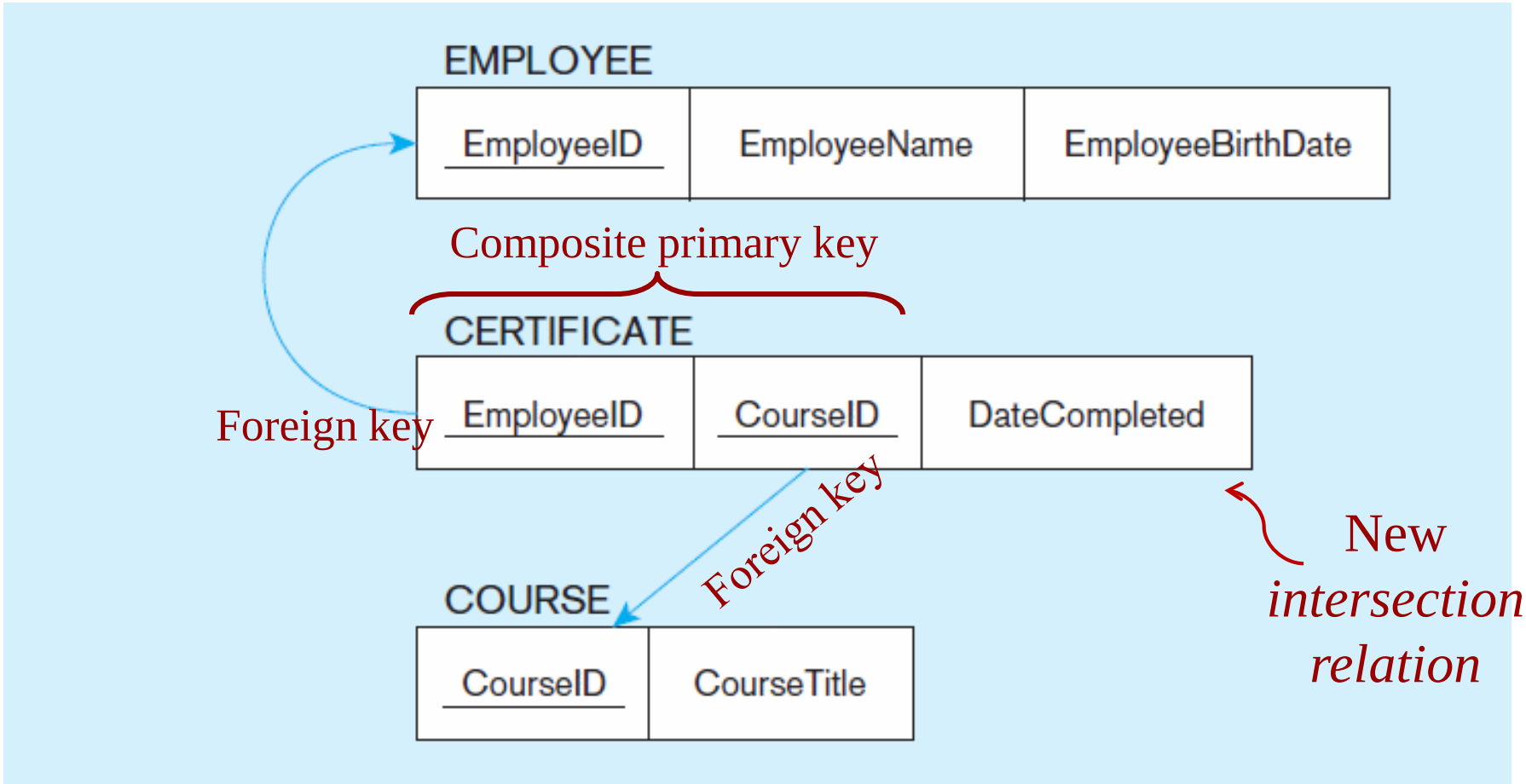
a) The "Completes" relationship (M:N)




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

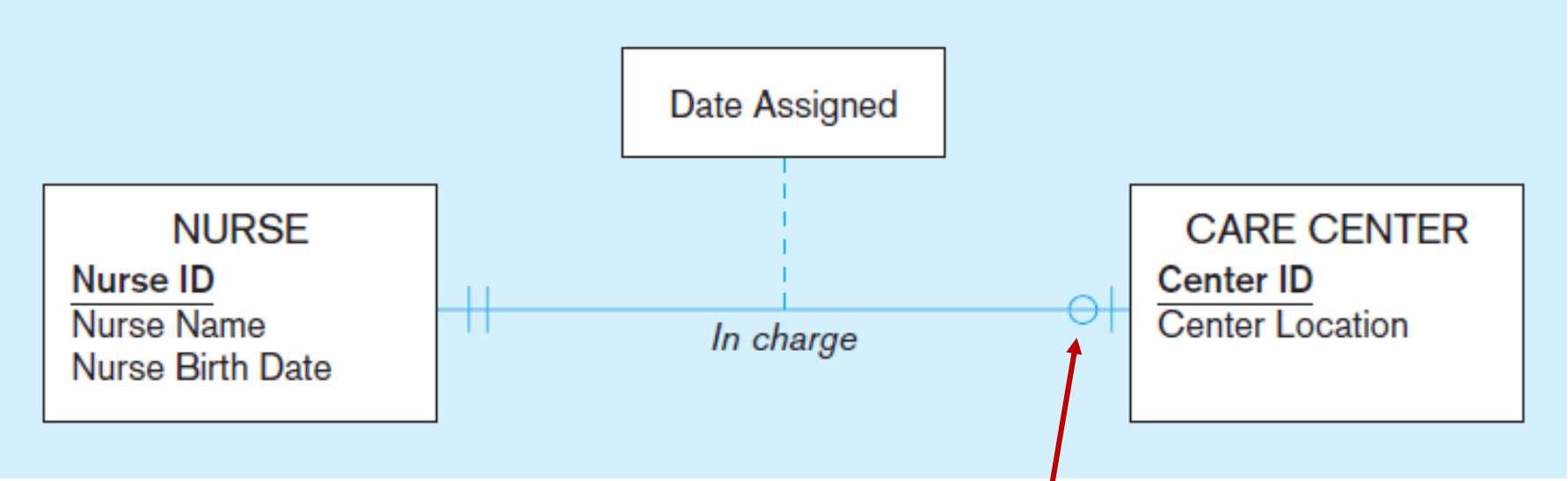
# Example of mapping an M:N relationship (cont.)

b) The new "CERTIFICATE" relation



# Example of mapping a binary 1:1 relationship

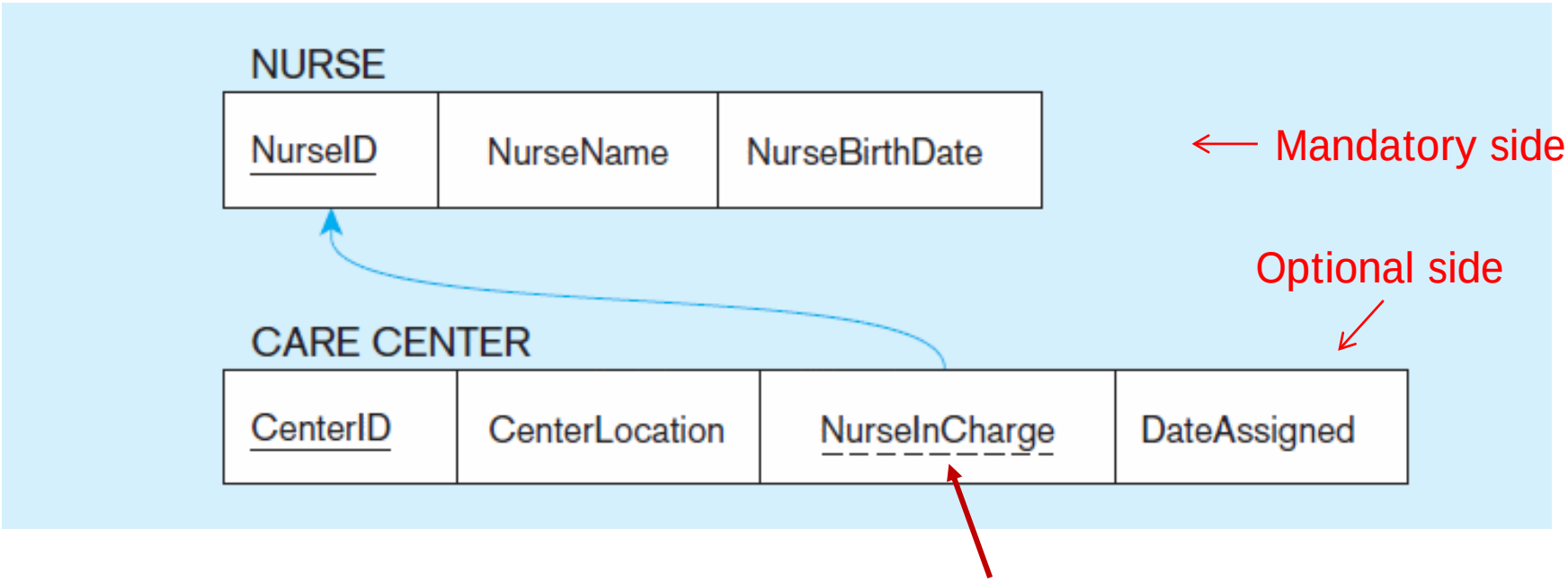
a) The “In charge” relationship (1:1)




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

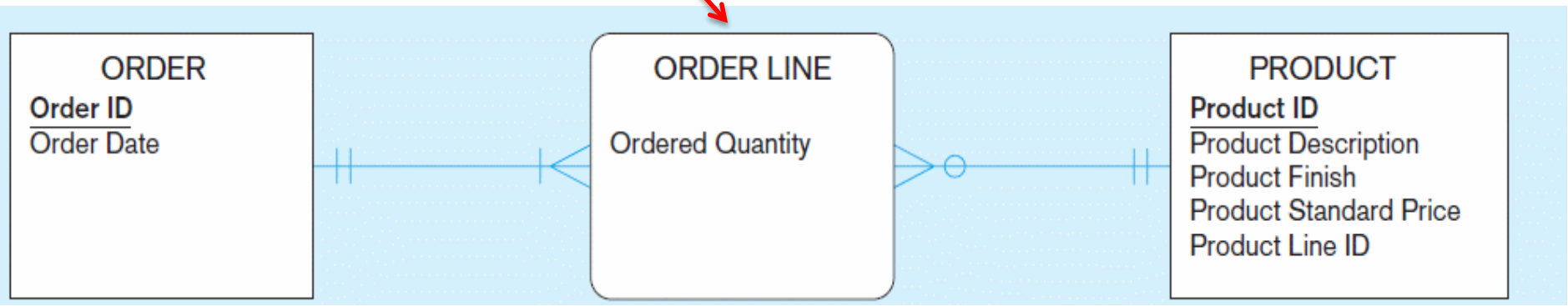
# Transforming EER Diagrams into Relations (cont.)

## Mapping **Associative Entities (AE)**

- Identifier Not assigned in AE
  - Default primary key for the associative relation is composed of the primary keys of the other two entities (as in M:N relationship)
- Identifier assigned in AE
  - The identifier is used as the primary key of the associative relation.
  - The primary keys of the other two entities (as in M:N relationship) become foreign keys of the associative relation.

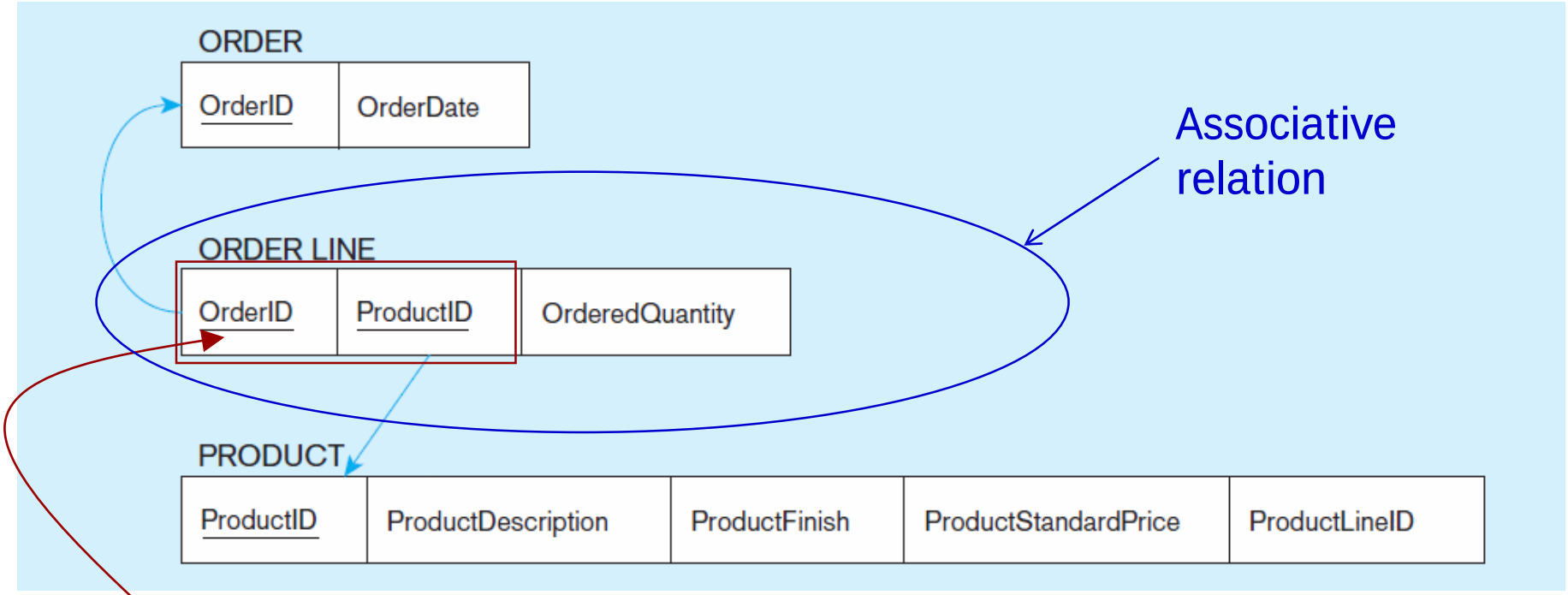
# Example of mapping an associative entity

## a) An associative entity w/o an identifier



# Example of mapping an associative entity (cont.)

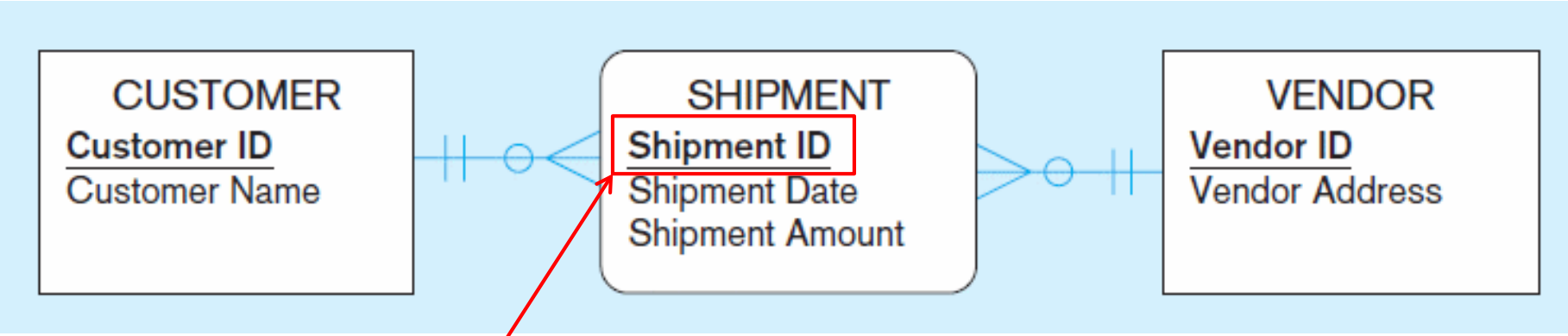
b) Three resulting relations




Composite primary key formed from the two foreign keys

# Example of mapping an associative entity with an identifier

a) SHIPMENT associative entity

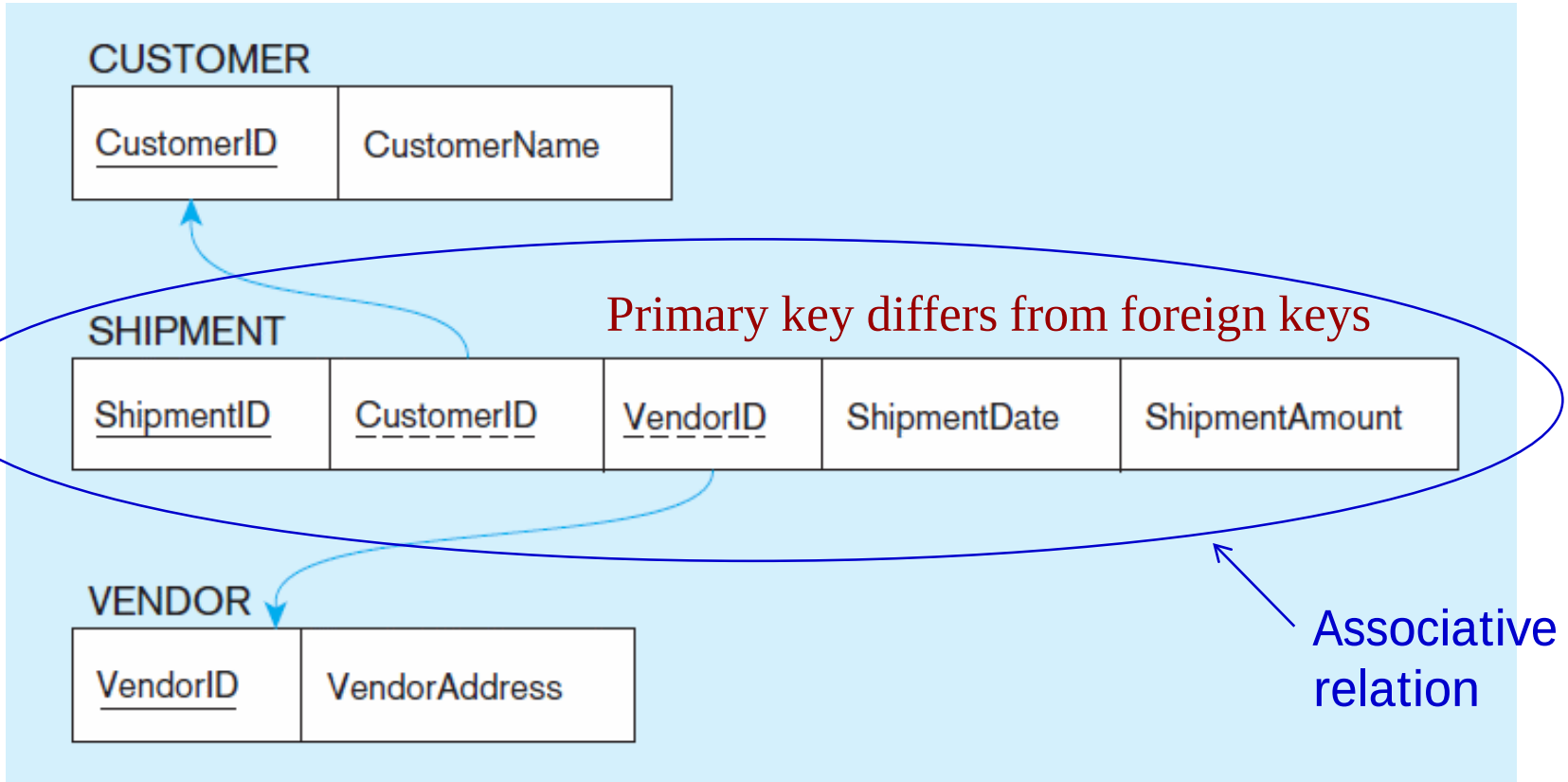



Two reasons to be assigned an identifier:

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

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

b) Three resulting relations



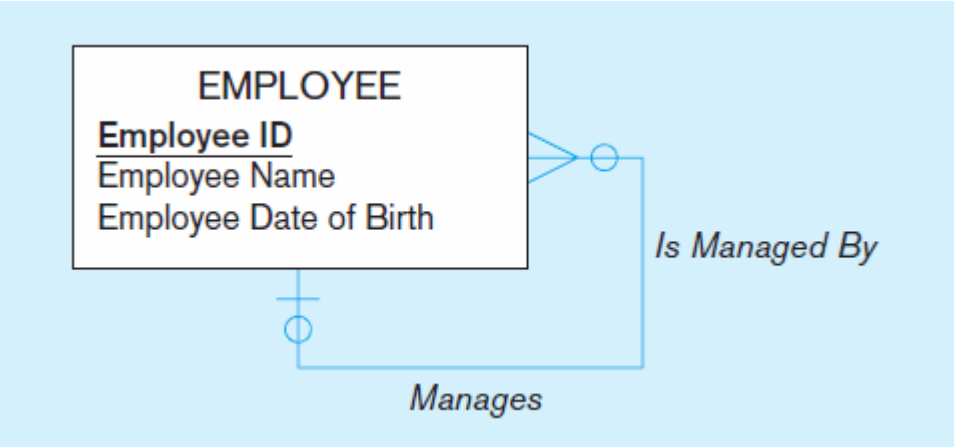
# Transforming EER Diagrams into Relations (cont.)

## Mapping Unary Relationships

- One-to-Many: A *recursive foreign key* is added to reference the primary key in the same relation; the recursive foreign key must have the same domain as the primary key.
- Many-to-Many: Two relations are created
  - 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

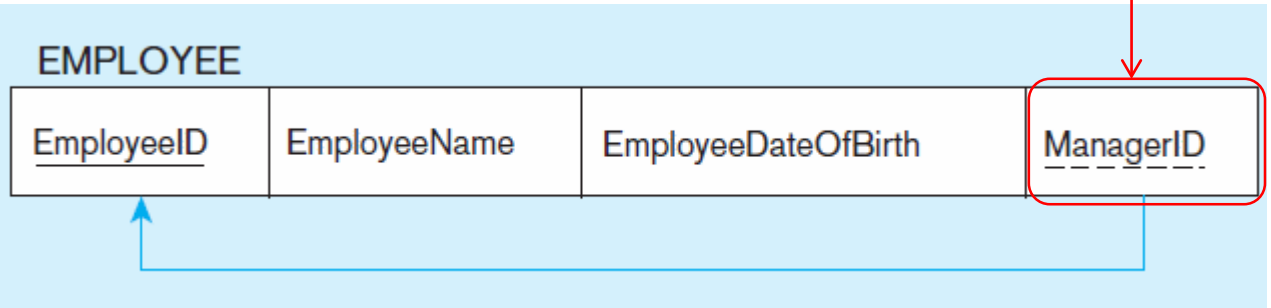
# Mapping a unary 1:M relationship

(a) EMPLOYEE entity with unary relationship

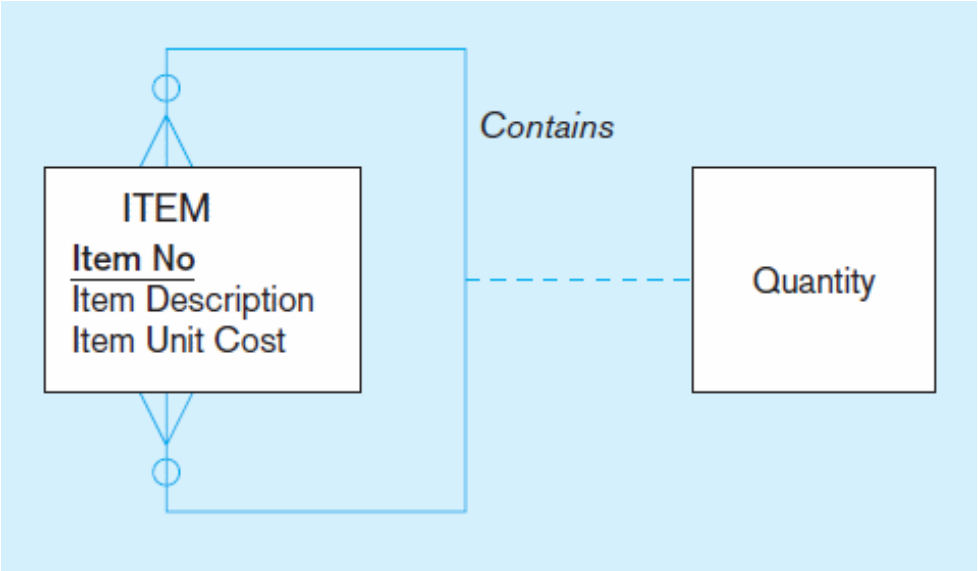



Newly added recursive foreign key, which is also an EmployeeID

(b) EMPLOYEE relation with recursive foreign key

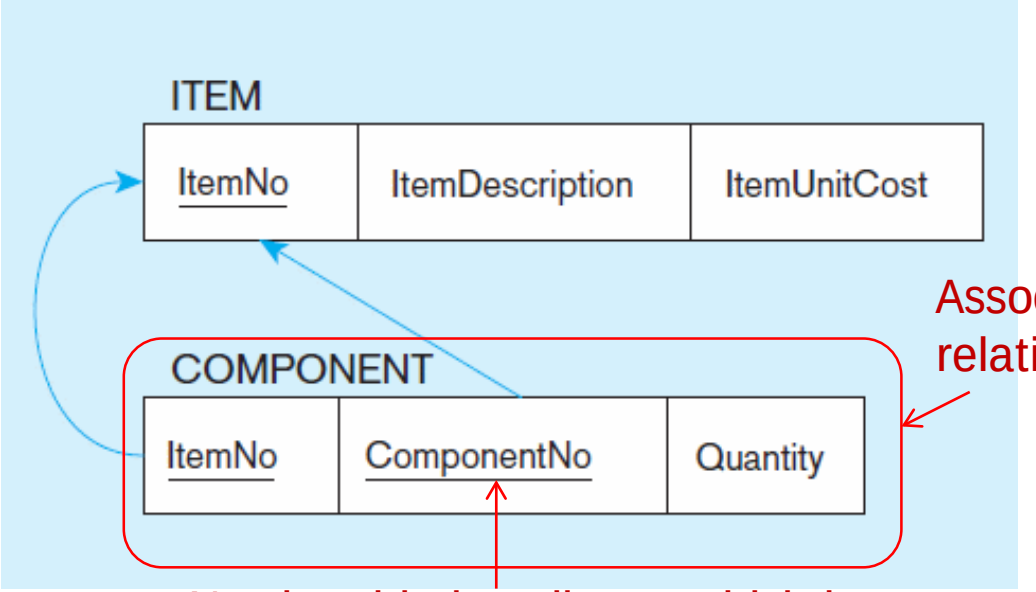


# Mapping a unary M:N relationship




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

(b) ITEM and COMPONENT relations



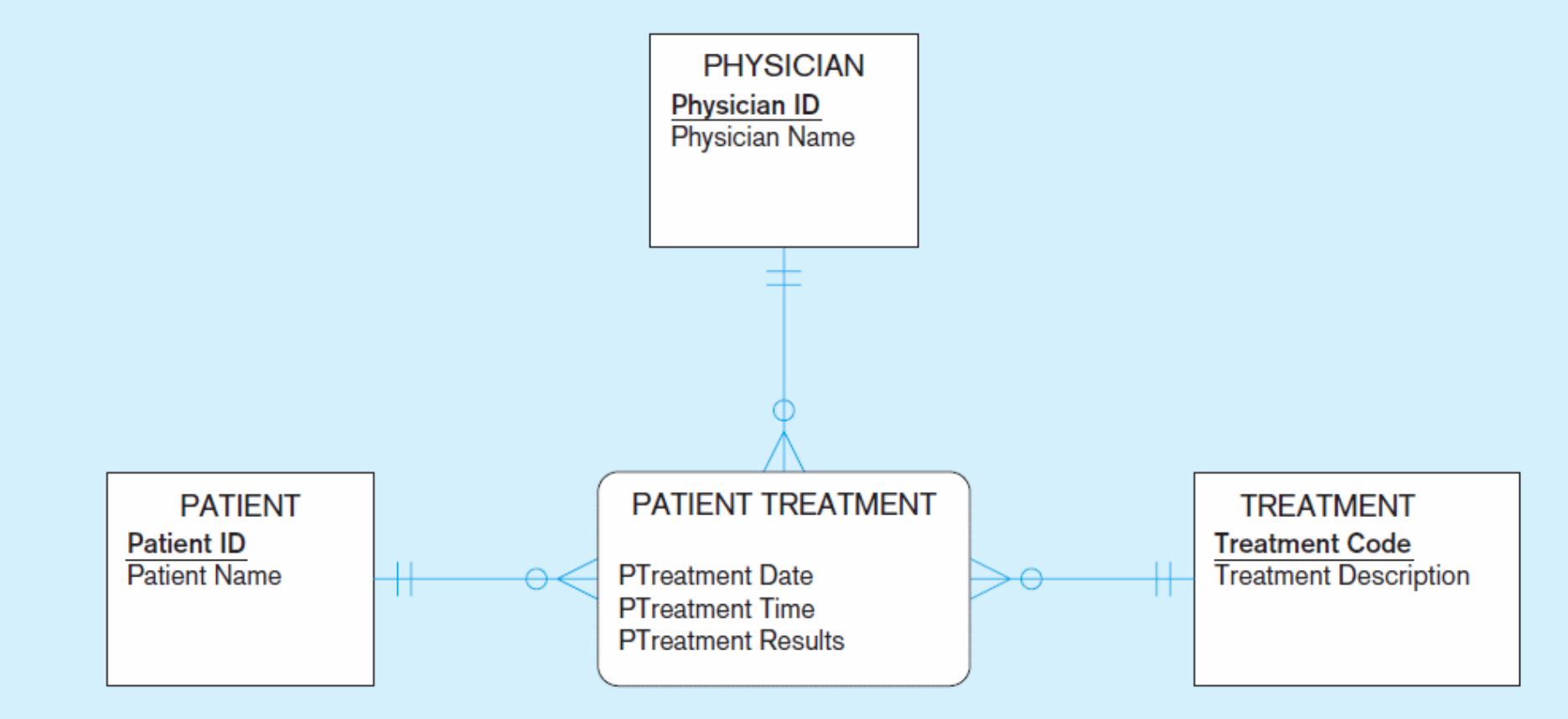
# Transforming EER Diagrams into Relations (cont.)

## Mapping Ternary (and n-ary) Relationships

- Create one relation for each entity and one for the associative entity.
- Assign the *default primary key* to the associative entity using the three primary keys of the participating entity types, which are also the foreign keys of the associative entity; in some cases, additional attributes are required to form a unique PK.

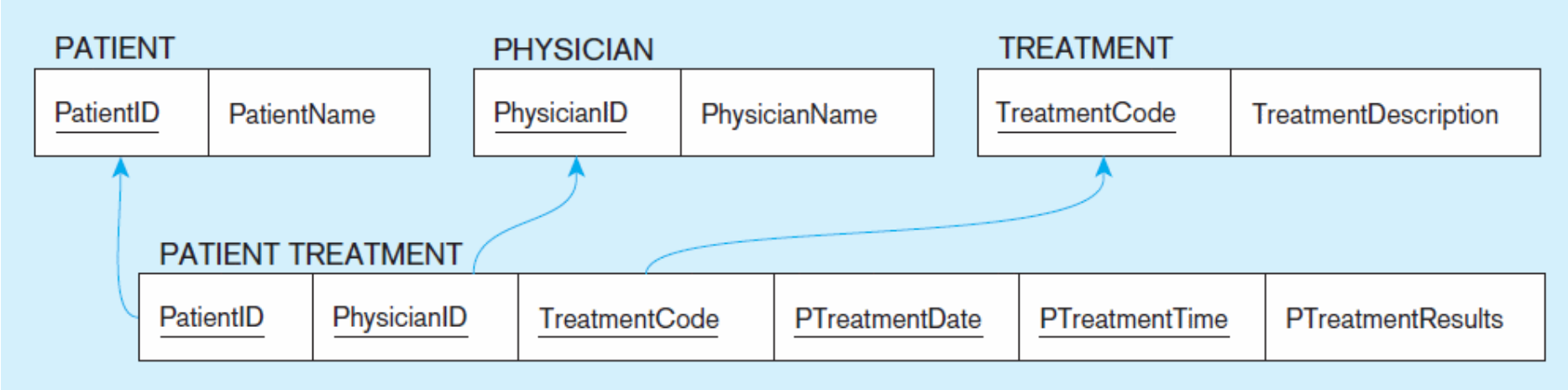
# Mapping a ternary relationship

a) PATIENT TREATMENT Ternary relationship with associative entity



# Mapping a ternary relationship (cont.)

b) Mapping the ternary relationship PATIENT TREATMENT



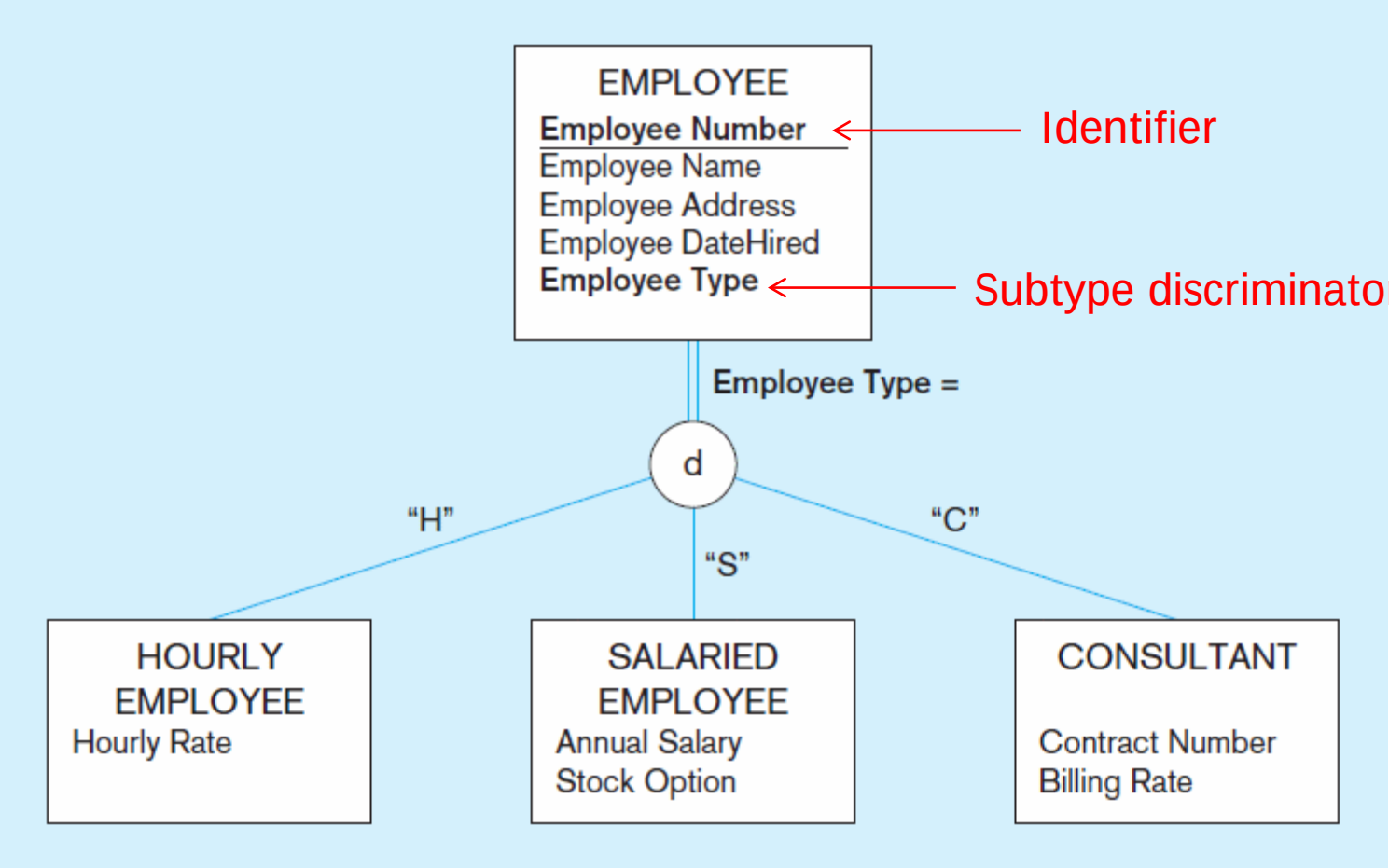

A primary key MUST be unique. In this case, the default primary key consisting of the three FKs is still not unique since the same physician may give a patient the same treatment more than once on a same day. So, two more attributes, PTreatmentDate and PTreatmentTime, are needed to make it unique. This is a pretty cumbersome PK; it would be better to create a surrogate key like Treatment#.

# Transforming EER Diagrams into Relations (cont.)

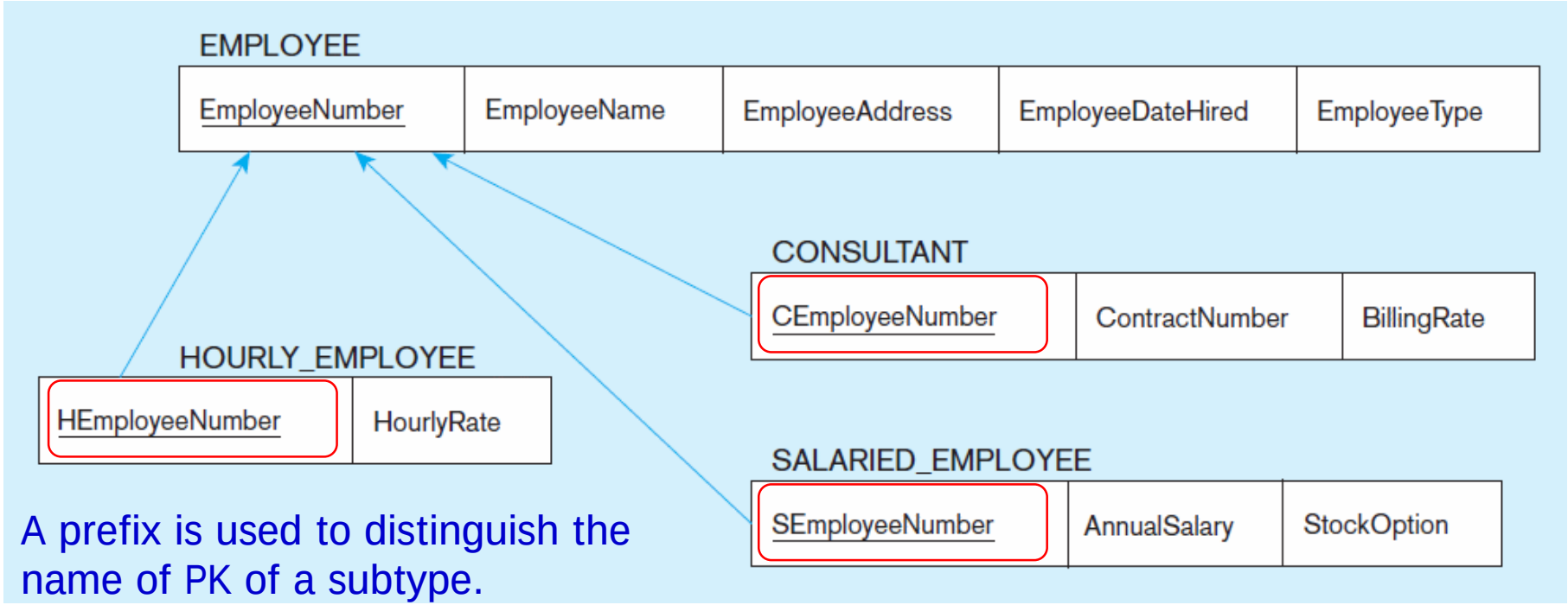
Mapping Supertype/Subtype Relationships

- Create a separate relation for supertype and for each subtype
- Supertype attributes (including identifier and *subtype discriminator*) go into the supertype relation
- Subtype attributes go into respective subtypes; the *primary key* of supertype relation also becomes *primary key* of subtype relation
- Establish 1:1 relationship between the 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. The FKs all point to the EmployeeNumber, the PK of EMPLOYEE relation.

# Normalization

- The process of decomposing relations with *anomalies* (errors or inconsistencies that may result when updating a table containing redundant data) to produce smaller, *well-structured* relations, in which the attributes are grouped in a proper way with all the anomalies removed.
- It is primarily a tool to validate and improve a logical design so that it satisfies certain constraints that *avoid problematic duplication of data.*

# Well-Structured Relation

- A relation that contains minimal data redundancy and allows users to insert, delete, and update rows without causing data inconsistencies
- The goal is to avoid three kinds of anomalies
  - **Insertion Anomaly**–adding a new row forces users to add other unnecessary or redundant data
  - **Deletion Anomaly**–deleting rows may cause a loss of data that would be needed for other purpose
  - **Modification Anomaly**–changing data in a row forces changes to other rows because of data duplication

**General rule of thumb: A relation (or table) should not be related to more than one entity type**

# Example

EMPLOYEE2

| EmpID | Name | DeptName | Salary | CourseTitle | 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 |
| 19 0 | Lorenzo Davis | Finance | 55,000 | | |
| 15 0 | Susan Martin | Marketing | 42,000 | SPSS | 6/19/201X |
| 15 0 | Susan Martin | Marketing | 42,000 | Java | 8/12/201X |

# Anomalies in this Table

- **Insertion**–can't enter a new employee without having the employee take a class
- **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 Dependency

A constraint between two attributes in which the value of one attribute (the ***determinant***) determines the value of another attribute.

Attribute B is functionally dependent on A (or a value of A uniquely determines the value of B): A → B, but it is not a mathematical dependency.

Examples: SSN → Name, Address, Birthdate

(For a known value of SSN, there can be only one name, one address and one Birthdate.)

VIN → Make, Model, Color

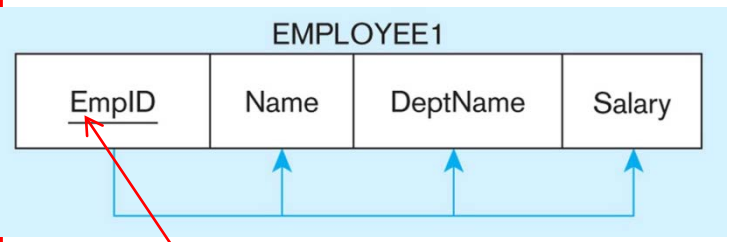
EmpID, CourseTitle → DateCompleted

**Logical AND operator**

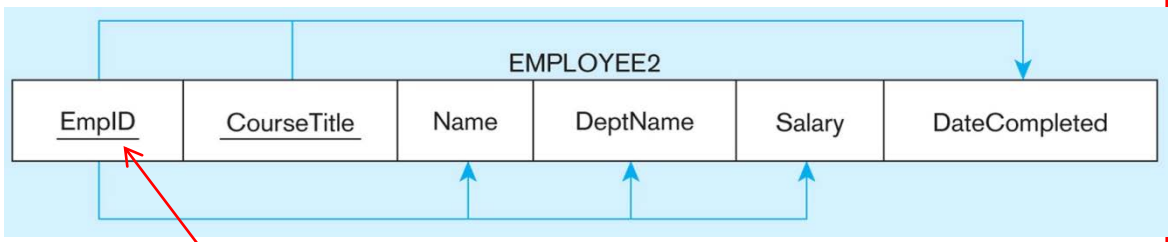
# Candidate Keys

- 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 SS# in a table...in this case both are candidate keys
  - Each non-key attribute is functionally dependent on every candidate key because the value of the key uniquely identifies that row.




EmpID uniquely identify a row.




EmpID alone does not uniquely identify a row.

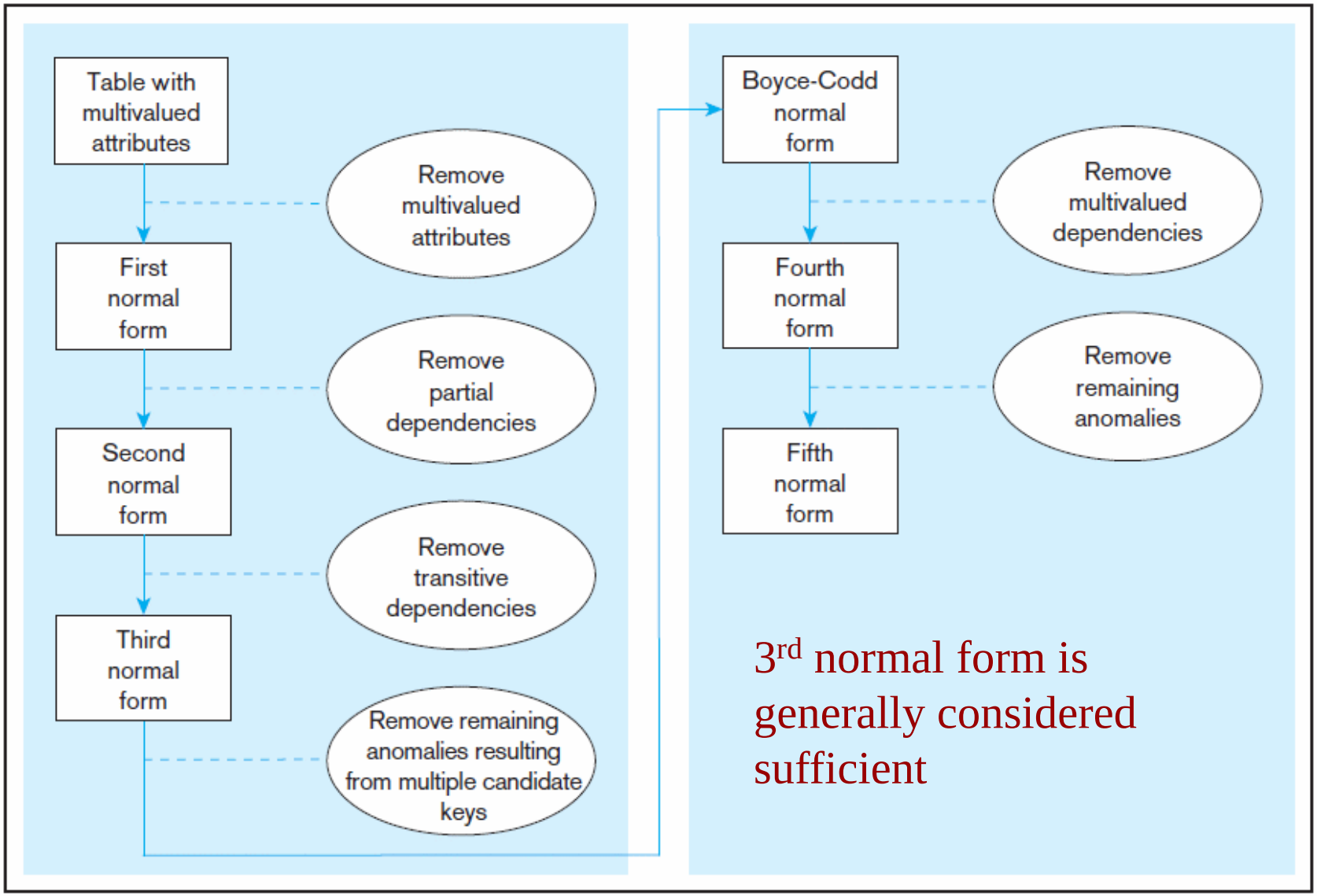
# Determinant vs. Candidate Key

- A candidate key is always a determinant
- A determinant may or may not be a candidate key (e.g., EmpID in EMPLOYEE2)
- A determinant may be part of a composite candidate key (e.g., EmpID in EMPLOYEE2), or even a non-key attribute

# Normal Form

A relation with a state that certain rules regarding the relationships between attributes or functional dependencies are satisfied, e.g., first normal form, second normal form, etc.

# Steps in normalization



# First Normal Form (1NF)

- Having a primary key
- No multivalued attributes (or repeating groups)
- Every attribute value is atomic

***All relations*** **are in 1st Normal Form**

# Table with multivalued attributes, not in 1st normal form

| 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 |

Note: this is NOT a relation

# Table with no multivalued attributes and with unique rows, in 1st normal form

| 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 relation

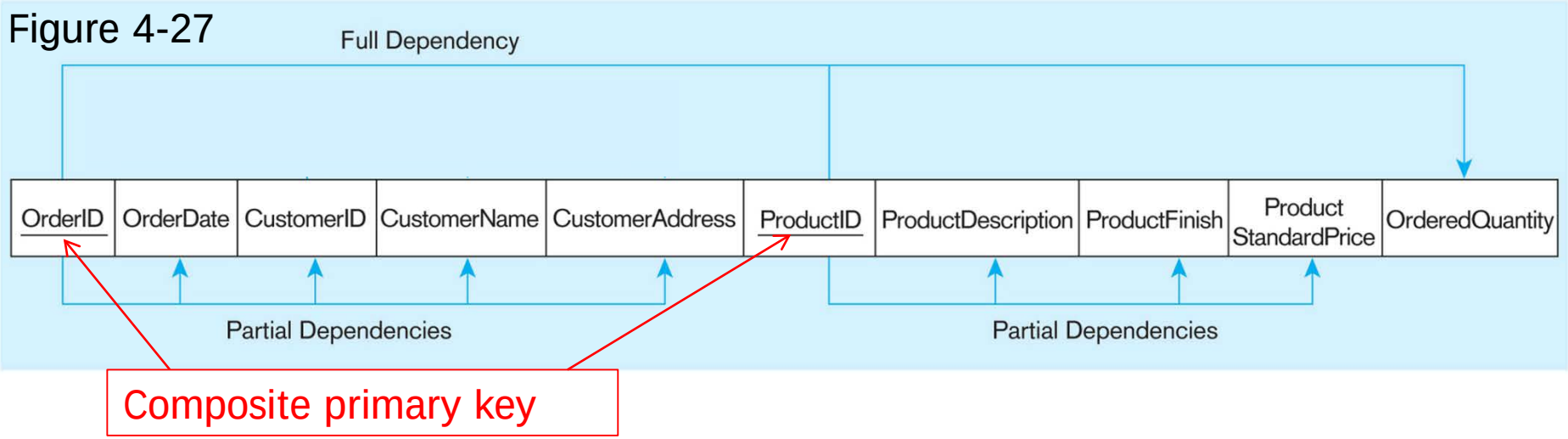
# Anomalies in this Table

- **Insertion**–if a new product is ordered for order 1007 of an 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 two records; or, if customer #2 has a new address, you have to change it in three places.

# An anomaly in 1NF

**Partial functional dependency:** a situation where one or more non-key attributes are functionally dependent on part (but not all) of the primary key.

Figure 4-27




OrderID ➔ OrderDate, CustomerID, CustomerName, CustomerAddress

ProductID ➔ ProductDescription, ProductFinish, ProductStandardPrice

OrderID, ProductID ➔ OrderQuantity

CustomerID ➔ CustomerName, CustomerAddress

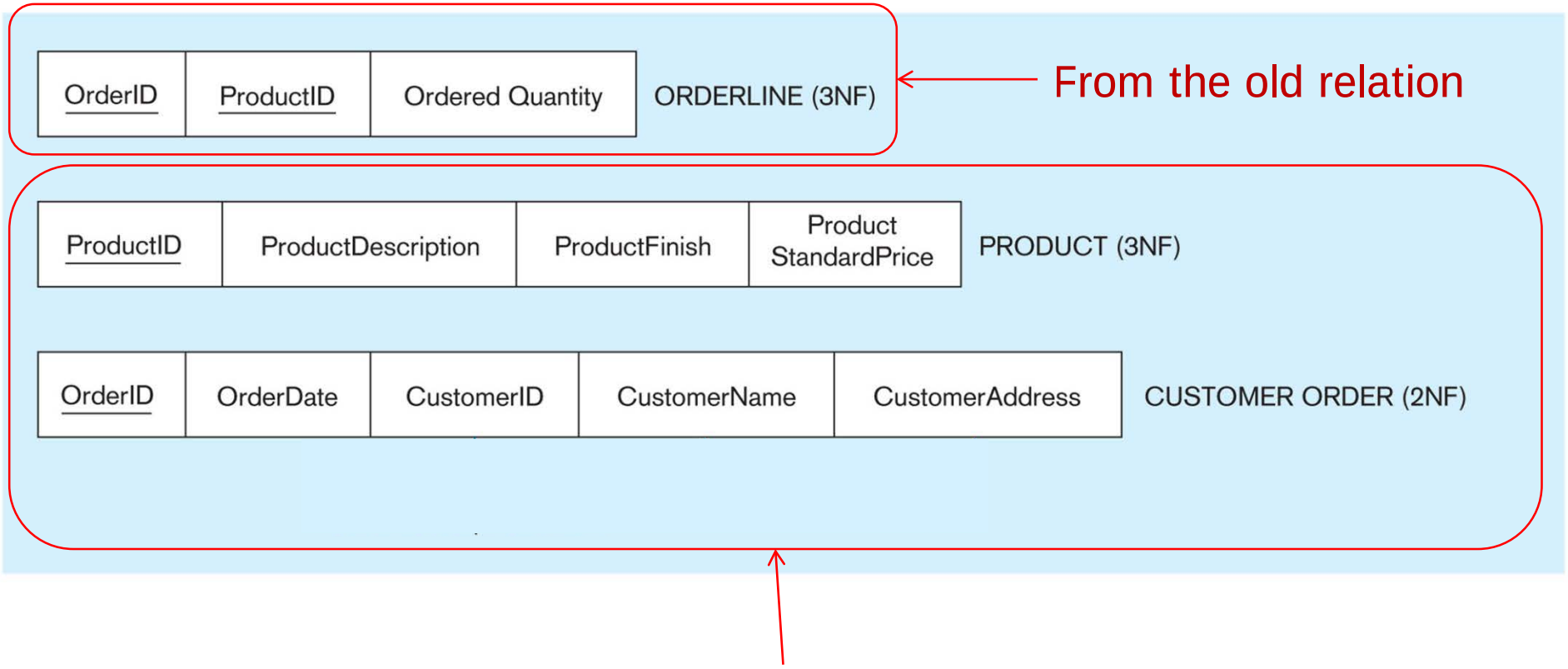
# Second Normal Form (2NF)

1NF PLUS every non-key attribute is fully functionally dependent on the ENTIRE primary key

Two steps to convert a 1NF w/ partial dependency to a 2NF:

- Create a new relation for each partial dependency
- Make the primary key attribute (or a determinant) a primary key of the new relation(s)

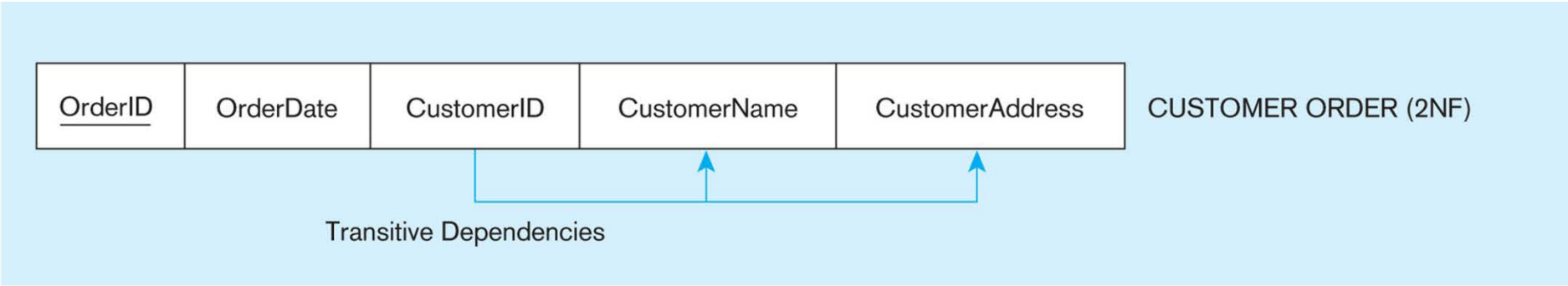
# Removing partial dependencies



# An anomaly in 2NF

**Transitive dependency:** A functional dependency between the primary key and one or more non-key attributes that are dependent on primary key via another non-key attribute.

(Or, it can be considered as a functional dependency within non-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.

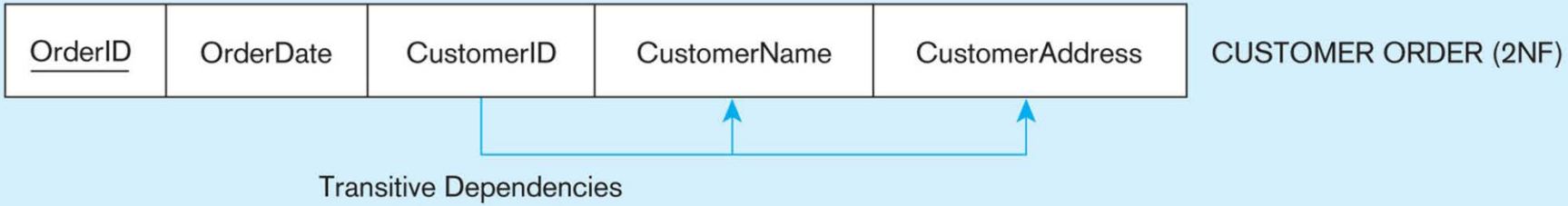
# Third Normal Form (3NF)

2NF PLUS ***no transitive dependencies*** (functional dependencies on non-primary-key attributes)

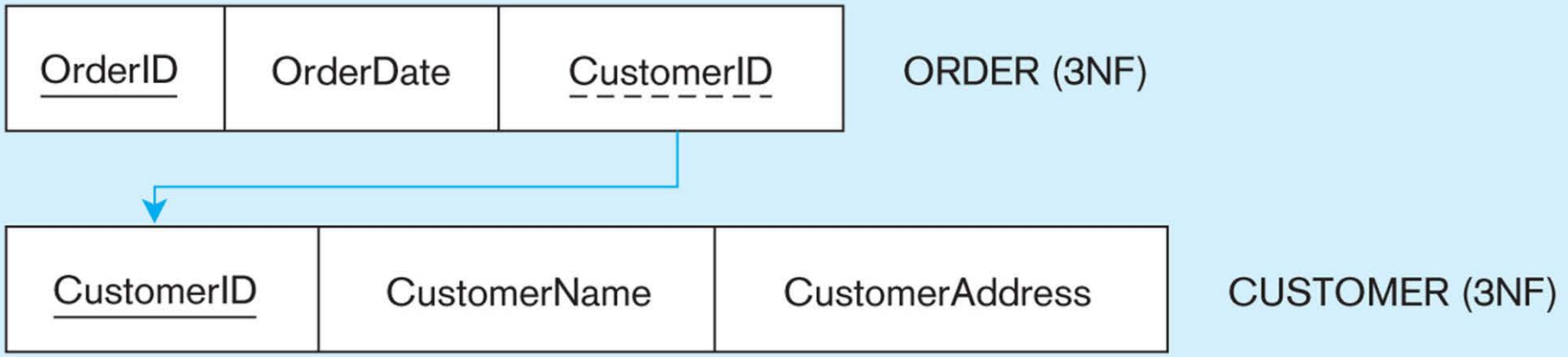
How to convert a 2NF w/ transitive dependency to a 3NF?

- The non-key determinant with transitive dependencies goes into a new relation
- The non-key determinant becomes primary key in the new relation and *stays as foreign key in the old table*

# Removing transitive dependencies




Getting it into a Third Normal Form

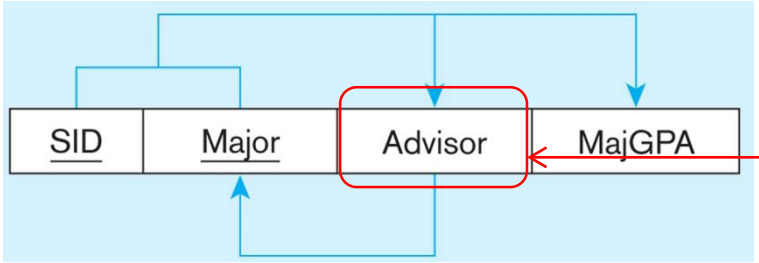



3NF is generally considered sufficient for most of practical DB applications.

# Normal Forms beyond 3NF

- Boyce-Codd normal form (BCNF): Every determinant is a candidate key so that there is no functionally dependent anomalies.

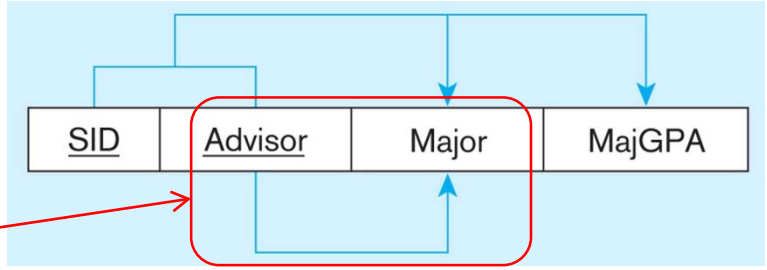
In 3NF




Determinant "Advisor" is not a candidate key

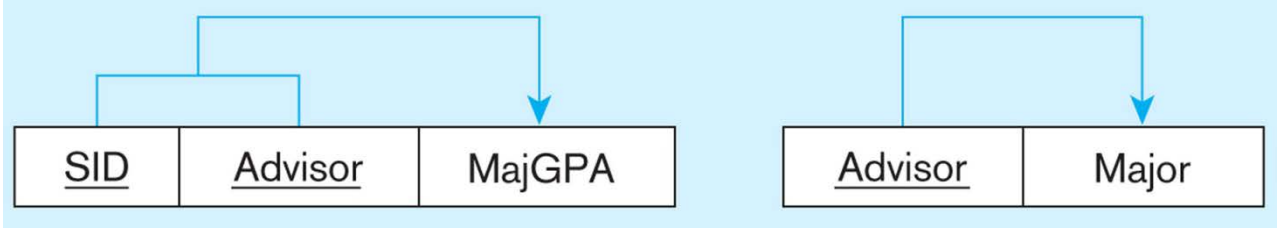
Reverse the relationship (relation restructuring)

In 1NF




Remove the partial dependency

In 3NF




If a relation has only one candidate key, 3NF and BCNF are equivalent.

# Fourth normal form (4NF)

A normal form in BCNF contains no multivalued dependencies (ones that exist when there are at least three attributes (A, B and C) in a relation, with a well-defined set of B and C values for each A value, but B and C are independent).

A user view

COURSE STAFF AND BOOK ASSIGNMENTS

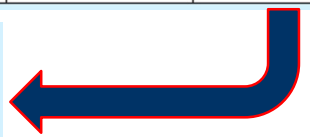
| Course | Instructor | Textbook |
|---|---|---|
| Management | White<br>Green<br>Black | Drucker<br>Peters |
| Finance | Gray | Jones<br>Chang |

A B C

OFFERING

| Course | Instructor | Textbook |
|---|---|---|
| Management | White | Drucker |
| Management | White | Peters |
| Management | Green | Drucker |
| Management | Green | Peters |
| Management | Black | Drucker |
| Management | Black | Peters |
| Finance | Gray | Jones |
| Finance | Gray | Chang |

In 1NF & in BCNF



Decomposition into AB, AC

In 4NF

TEACHER A B

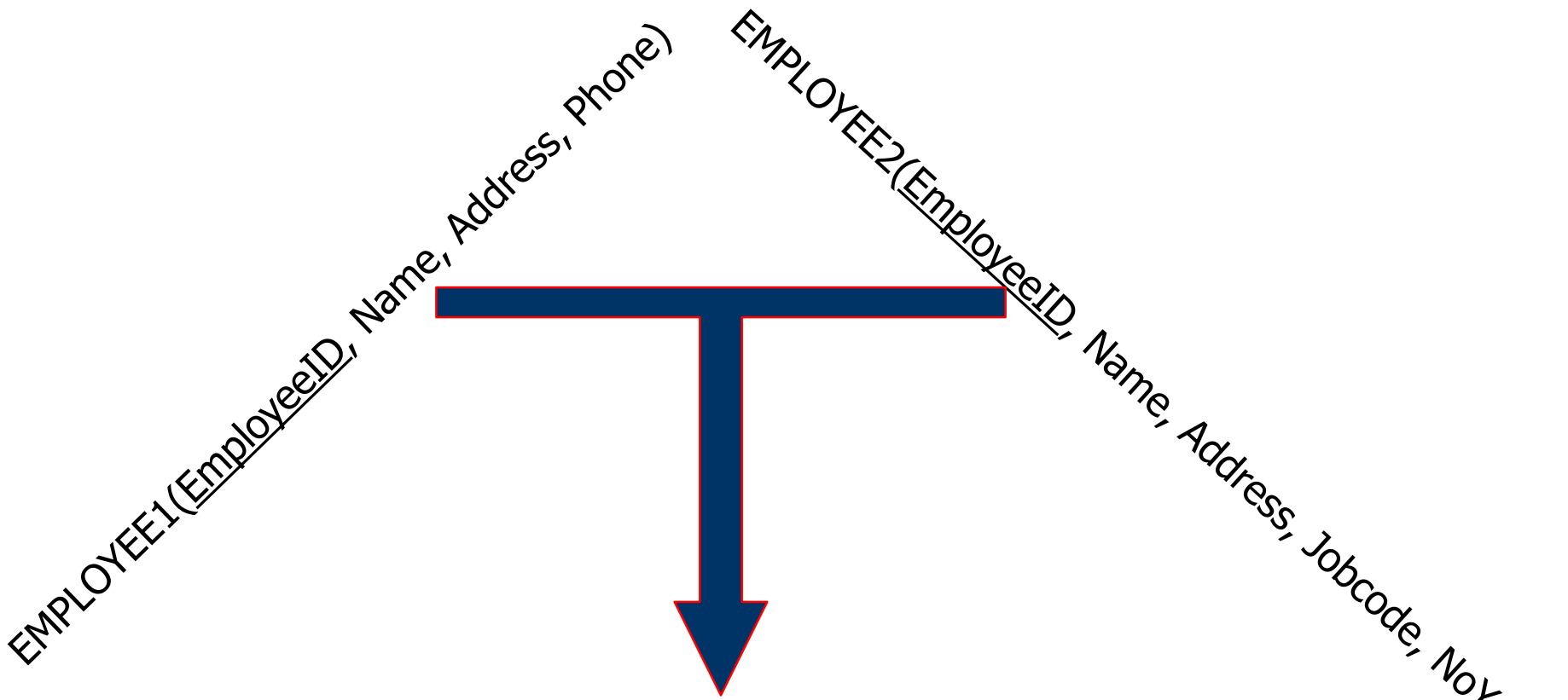
| Course | Instructor |
|---|---|
| Management | White |
| Management | Green |
| Management | Black |
| Finance | Gray |

TEXT A C

| Course | Textbook |
|---|---|
| Management | Drucker |
| Management | Peters |
| Finance | Jones |
| Finance | Chang |

# Merging Relations (View Integration)

A process of combining entities from multiple ER models into common relations




EMPLOYEE(EmployeeID, Name, Address, Phone, Jobcode, NoYears)

# Some possible issues for merging entities from different ER models

- Synonyms–two or more attributes with different names but the same meaning
  - Work with users to select one name or create a new one or use a alias
- Homonyms–attributes with the same name but different meanings
  - Need to create new attribute names
- Transitive dependencies–even if relations are in 3NF prior to merging, they may not be after merging
  - Remove any newly emerged dependencies
- Supertype/subtype relationships–may be hidden prior to merging
  - Create a new relation to reflect this relationship

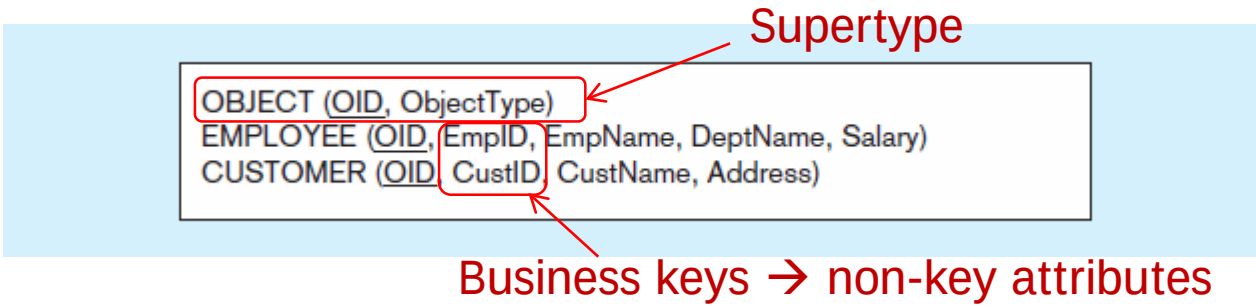
    PATIENT1(PatientID, Name, Address, DateTreated) and PATIENT2(PatientID, RoomNo) → PATIENT(PatientID, Name, Address), INPATIENT(PatientID, RoomNo) and OUTPATIENT(PatientID, DateTreated)

# Enterprise Keys

- Primary keys that are unique in the whole database, not just within a single relation.
- The primary keys become values internal to the DB system and have no business meanings.
- The traditional candidate (primary) key (business/natural key), e.g., EmpID, will be included as a non-key attribute.
- It is a relative new concept mainly for merging new relations into an existing DB w/o creating a ripple effect of foreign key changes.

# Enterprise Keys

a) Relations with an enterprise key (OID)




b) Sample data with an 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 |

# Enterprise Keys (continued)

c) Relations after adding the PERSON relation

OBJECT (OID, ObjectType)
EMPLOYEE (OID, EmpID, DeptName, Salary, PersonID)
CUSTOMER (OID, CustID, Address, PersonID)
PERSON (OID, Name)

New foreign keys

The new relation

d) Sample data after adding the PERSON relation

OBJECT

| OID | ObjectType |
|---|---|
| 1 | EMPLOYEE |
| 2 | CUSTOMER |
| 3 | CUSTOMER |
| 4 | EMPLOYEE |
| 5 | EMPLOYEE |
| 6 | CUSTOMER |
| 7 | CUSTOMER |
| 8 | PERSON |
| 9 | PERSON |
| 10 | PERSON |
| 11 | PERSON |
| 12 | PERSON |
| 13 | PERSON |
| 14 | PERSON |

PERSON

| OID | Name |
|---|---|
| 8 | Jennings, Fred |
| 9 | Fred's Warehouse |
| 10 | Bargain Bonanza |
| 11 | Hopkins, Dan |
| 12 | Huber, Ike |
| 13 | Jasper's |
| 14 | Desks 'R Us |

EMPLOYEE

| OID | EmpID | DeptName | Salary | PersonID |
|---|---|---|---|---|
| 1 | 100 | Marketing | 50000 | 8 |
| 4 | 101 | Purchasing | 45000 | 11 |
| 5 | 102 | Accounting | 45000 | 12 |

CUSTOMER

| OID | CustID | Address | PersonID |
|---|---|---|---|
| 2 | 100 | Greensboro, NC | 9 |
| 3 | 101 | Moscow, ID | 10 |
| 6 | 102 | Tallahassee, FL | 13 |
| 7 | 103 | Kettering, OH | 14 |