Chapter 5: Logical Database Design and the Relational Model

Modern Database Management

8th Edition

Jeffrey A. Hoffer, Mary B. Prescott, Fred R. McFadden

Objectives

- Definition of 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 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

is different from ralationship
and it's equal to table

Relation

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

  NOTE: all *relations* are in ***1st Normal form***

Example of Relation

Figure 5-1 EMPLOYEE1 relation with sample data

EMPLOYEE1

Emp_ID	Name	Dept_Name	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

we can express the structure of the employee relation as:

Employee1 (Emp_ID, Name, Dept_Name, Salary)

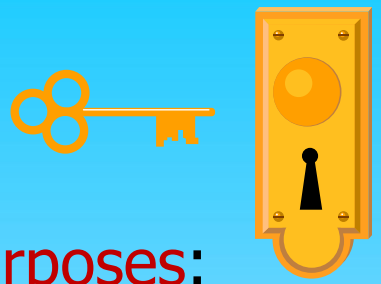
foreign key is dash under lined also

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)

Key Fields



- Keys are special fields that serve two main purposes:
  - ***Primary keys*** are unique identifiers of the relation in question.
- Keys can be ***simple*** (a single field) or ***composite*** (more than one field)
  - *This is how we can guarantee that all rows are unique*
  - Examples:

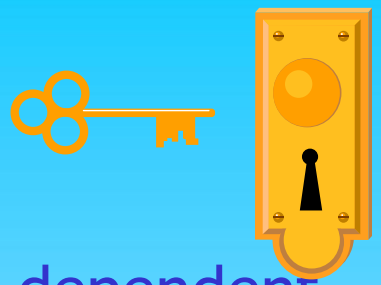
Simple Attribute:

Employee(Emp_ID, Name, Dept_Name, Salary)

Composite Key:

Order_Line(Order_No, Product_No, Quantity)

Key Fields



- ***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)
- Example:

Relationship between Employee and Department Tables

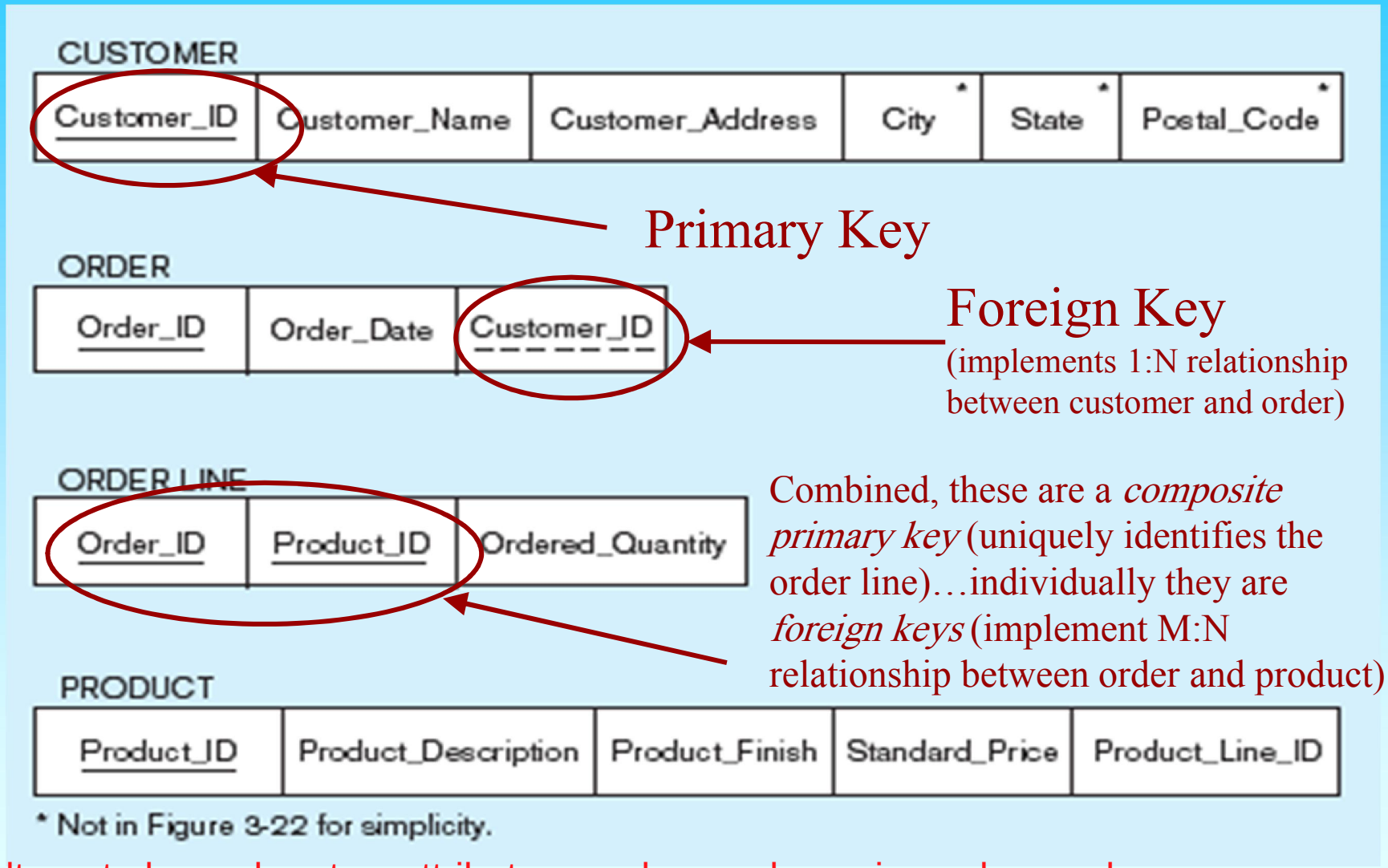
Employee(Emp_ID, Name, Dept_No, Salary)

Department(Dept_No, Location , Fax)

- Keys usually are used as indexes to speed up the response to user queries (More on this in Ch. 6)

important shape / tables / schema of relation

Figure 5-3 Schema for four relations (Pine Valley Furniture Company)



alternate key: when two attributes can be used as primary key and we can use any of them one of them is primary key and the other one is the alternate key the two are called candidate ID

Integrity Constraints

- Relational model contains several types of constraints (Business Rules) to facilitate maintaining the accuracy and Integrity of data , such as:
- Domain Constraints
  - A domain is the set of values assigned to an attribute
  - Domain definition consists of:
    - Domain name
    - Domain meaning
    - data type
    - Data size
    - Allowable values

Table 5-1 Domain Definitions for INVOICE Attributes

Attribute	*Domain Name*	*Description*	*Domain*
Customer_ID	Customer_IDs	Set of all possible customer IDs	character: size 5
Customer_Name	Customer_Names	Set of all possible customer names	character: size 25
Customer_Address	Customer_Addresses	Set of all possible customer addresses	character: size 30
City	Cities	Set of all possible cities	character: size 20
State	States	Set of all possible states	character: size 2
Postal_Code	Postal_Codes	Set of all possible postal zip codes	character: size 10
Order_ID	Order_IDs	Set of all possible order IDs	character: size 5
Order_Date	Order_Dates	Set of all possible order dates	date format mm/dd/yy
Product_ID	Product_IDs	Set of all possible product IDs	character: size 5
Product_Description	Product_Descriptions	Set of all possible product descriptions	character size 25
Product_Finish	Product_Finishes	Set of all possible product finishes	character: size 15
Standard_Price	Unit_Prices	Set of all possible unit prices	monetary: 6 digits
Product_Line_ID	Product_Line_IDs	Set of all possible product line IDs	integer: 3 digits
Ordered_Quantity	Quantities	Set of all possible ordered quantities	integer: 3 digits

Domain definitions enforce domain integrity constraints

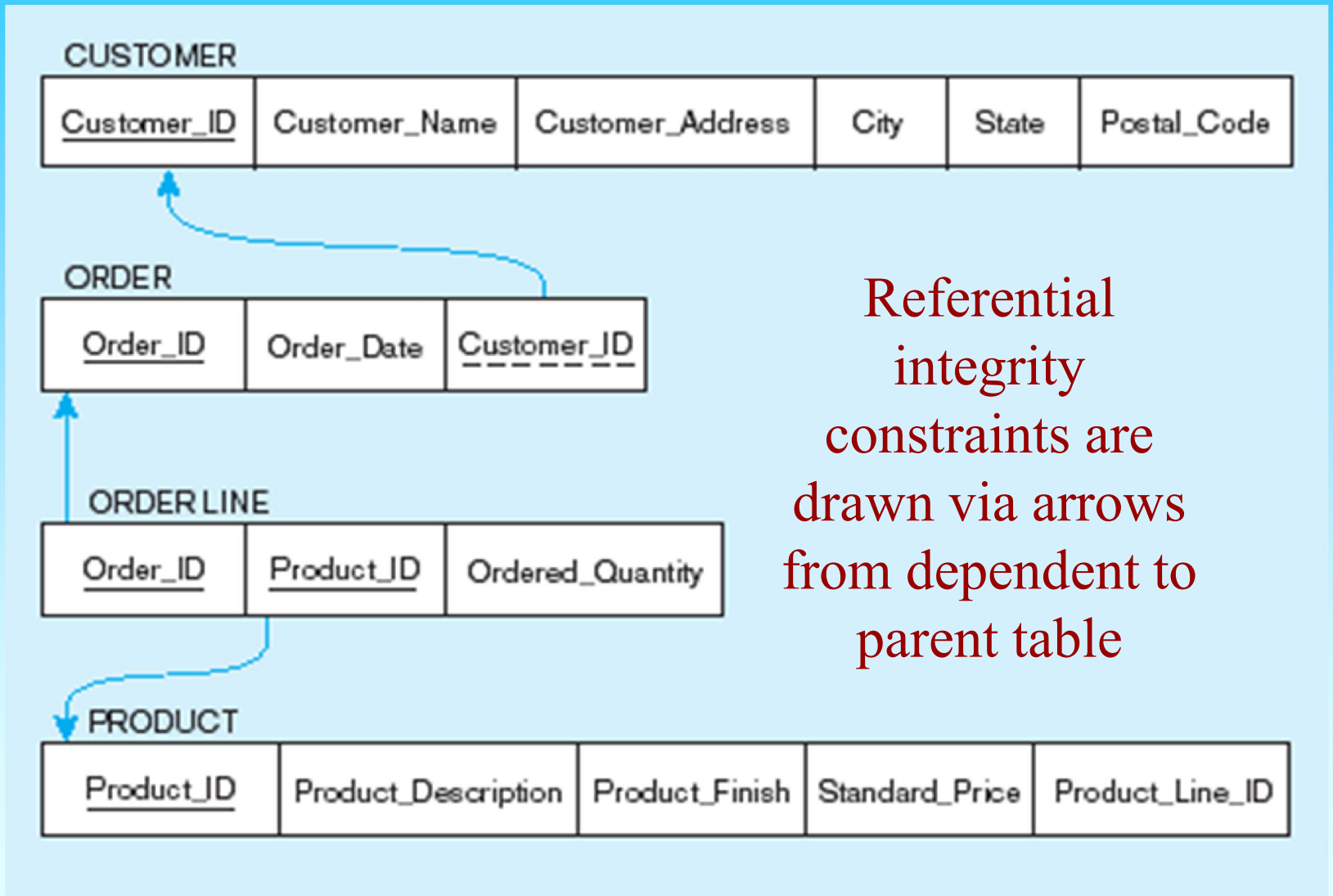
Integrity Constraints

- Entity Integrity important
  - Each relation must have a primary key
  - No primary key attribute may be null.
  - All primary key fields **MUST** have data
  - Null value may be assigned to non-key attribute when no other value applies or when the applicable value is unknown
- Action Assertions
  - Business rules. Recall from Ch. 4

Integrity Constraints

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

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



Creating Relational Tables

SQL table definitions

```
CREATE TABLE CUSTOMER
        (CUSTOMER_ID            VARCHAR(5)      NOT NULL,
        CUSTOMER_NAME           VARCHAR(25)     NOT NULL,
        CUSTOMER ADDRESS        VARCHAR(30)     NOT NULL,
        CITY                    VARCHAR(20)     NOT NULL,
        STATE                   CHAR(2)         NOT NULL,
        POSTAL_CODE             CHAR(10)        NOT NULL,
PRIMARY KEY (CUSTOMER_ID);

CREATE TABLE ORDER
        (ORDER_ID               CHAR(5)         NOT NULL,
        ORDER DATE              DATE            NOT NULL,
        CUSTOMER_ID             VARCHAR(5)      NOT NULL,
PRIMARY KEY (ORDER_ID),
FOREIGN KEY (CUSTOMER_ID) REFERENCES CUSTOMER (CUSTOMER_ID);

CREATE TABLE ORDER_LINE
        (ORDER_ID               CHAR(5)         NOT NULL,
        PRODUCT_ID              CHAR(5)         NOT NULL,
        ORDERED_QUANTITY        INT             NOT NULL,
PRIMARY KEY (ORDER_ID, PRODUCT_ID),
FOREIGN KEY (ORDER_ID) REFERENCES ORDER (ORDER_ID),
FOREIGN KEY (PRODUCT_ID) REFERENCES PRODUCT (PRODUCT_ID);

CREATE TABLE PRODUCT
        (PRODUCT_ID             CHAR(5)         NOT NULL,
        PRODUCT_DESCRIPTION     VARCHAR(25),
        PRODUCT_FINISH          VARCHAR(12),
        STANDARD_PRICE          DECIMAL(8,2)    NOT NULL,
        PRODUCT_LINE_ID         INT             NOT NULL,
PRIMARY KEY (PRODUCT_ID);
```

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

Transforming EER Diagrams into Relations

until here

Transforming (mapping) ER-Diagrams to Relations is straightforward with a well-defined set of rules.

Types of Entities

1. Regular Entities: have an independent existence such as persons, products, ….
2. Weak Entities: dependent and cannot exist without identifying relationship with an owner (Regular Entity)
3. Associative Entities: formed from many-to-many relationships between other entities

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

Figure 5-8 Mapping a regular entity

(a) CUSTOMER entity type with simple attributes

(b) CUSTOMER relation

CUSTOMER

Customer_ID	Customer_Name	Customer_Address	Postal_Code

Figure 5-9 Mapping a composite attribute

(a) CUSTOMER entity type with composite attribute

(b) CUSTOMER relation with address detail

CUSTOMER

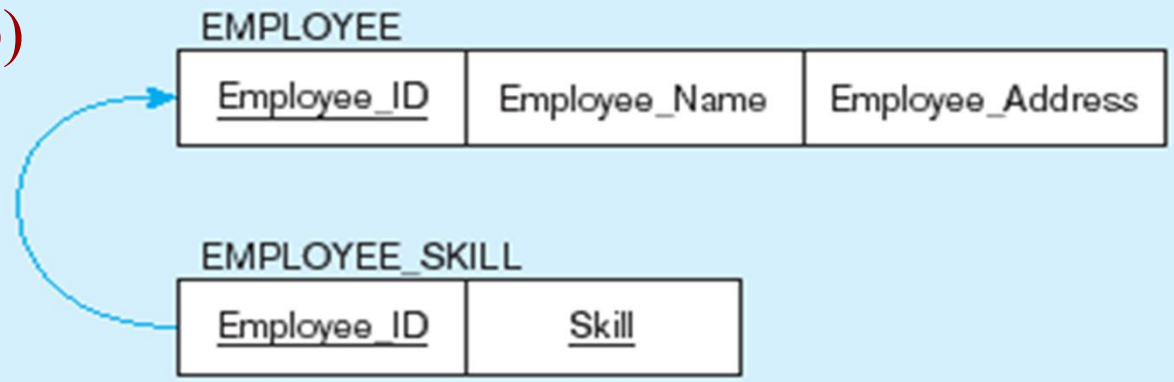
Customer_ID	Customer_Name	Street	City	State	Postal_Code

Figure 5-10 Mapping an entity with a multivalued attribute

(a)

Multivalued attribute becomes a separate relation with foreign key

(b)



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

Transforming EER Diagrams into Relations (cont.)

Mapping Weak Entities

- Weak entity cannot exist independently
- It does not have a complete Identifier
- 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)

Figure 5-11 Example of mapping a weak entity

a) Weak entity DEPENDENT

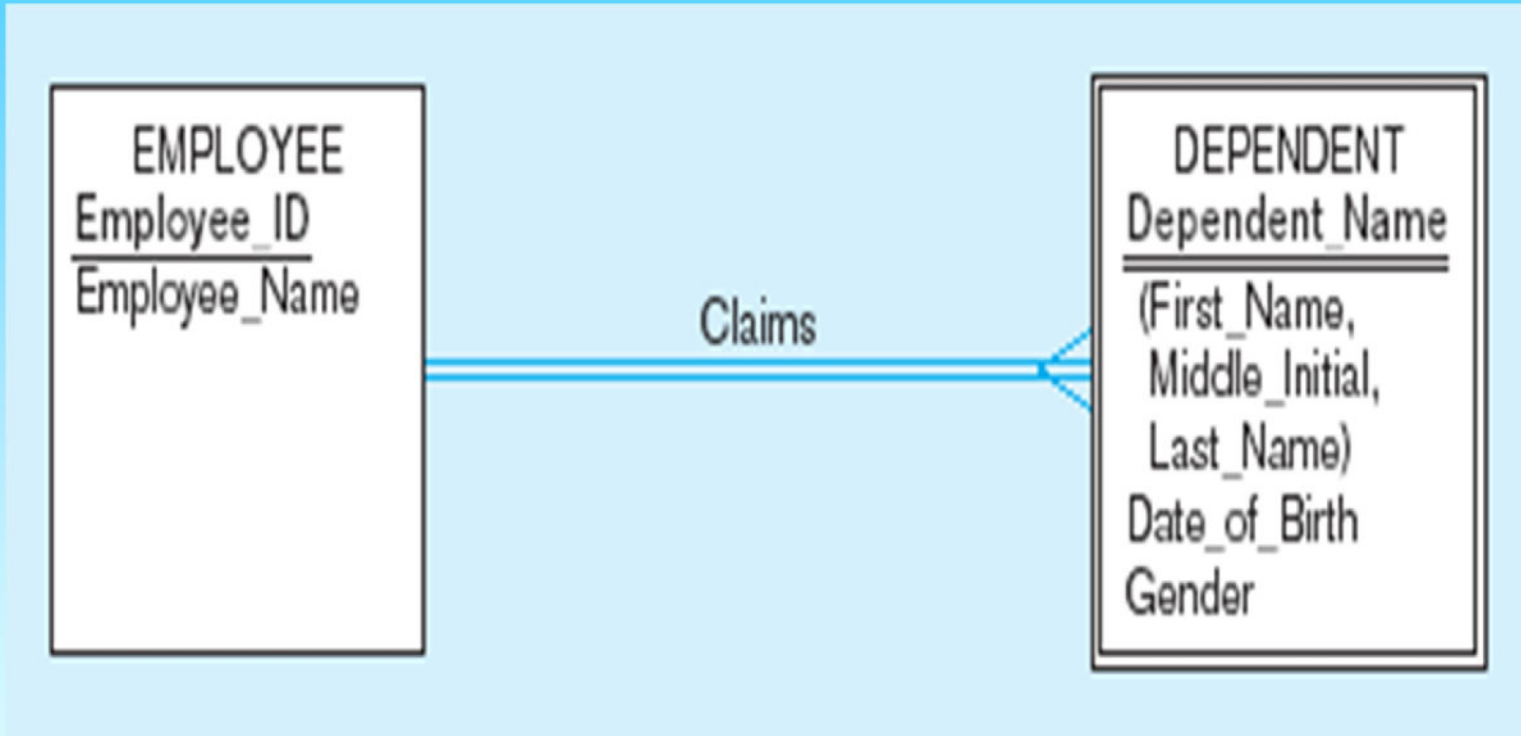
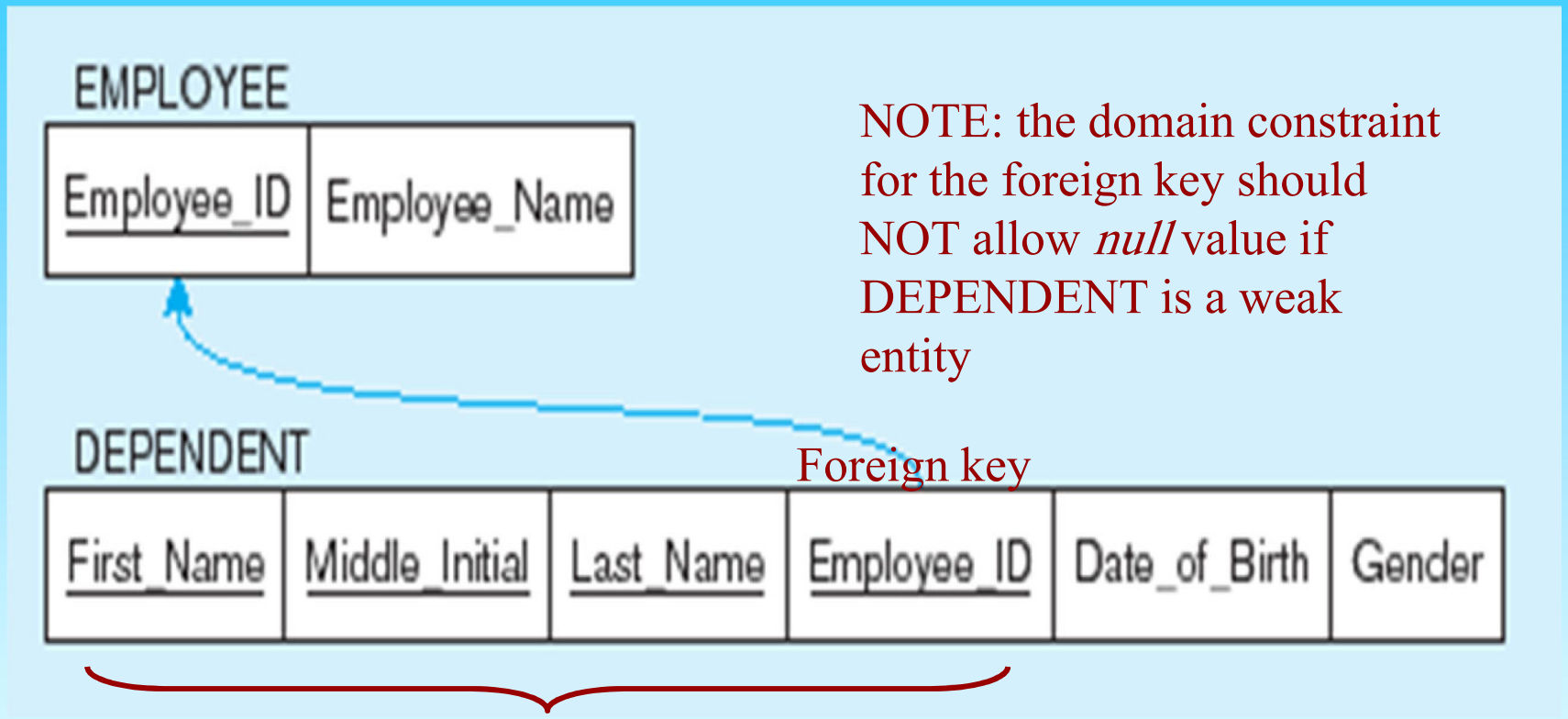


Figure 5-11 Example of mapping a weak entity (cont.)
b) Relations resulting from weak entity



Alternate approach:
Dependent(Employee_ID, Dependent#, First_Name, Middle_Name, Last_Name, Date_of_Birth, Gender)

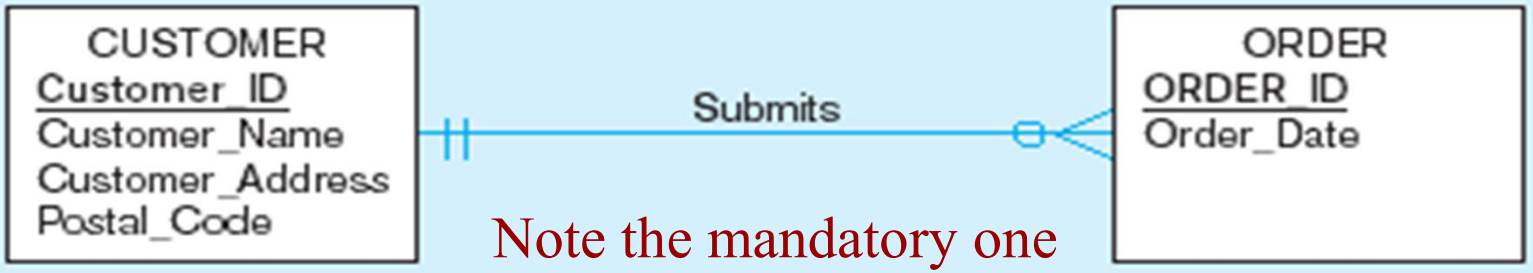
Transforming EER Diagrams into Relations (cont.)

Mapping Binary Relationships

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

Figure 5-12 Example of mapping a 1:M relationship

a) Relationship between customers and orders



b) Mapping the relationship

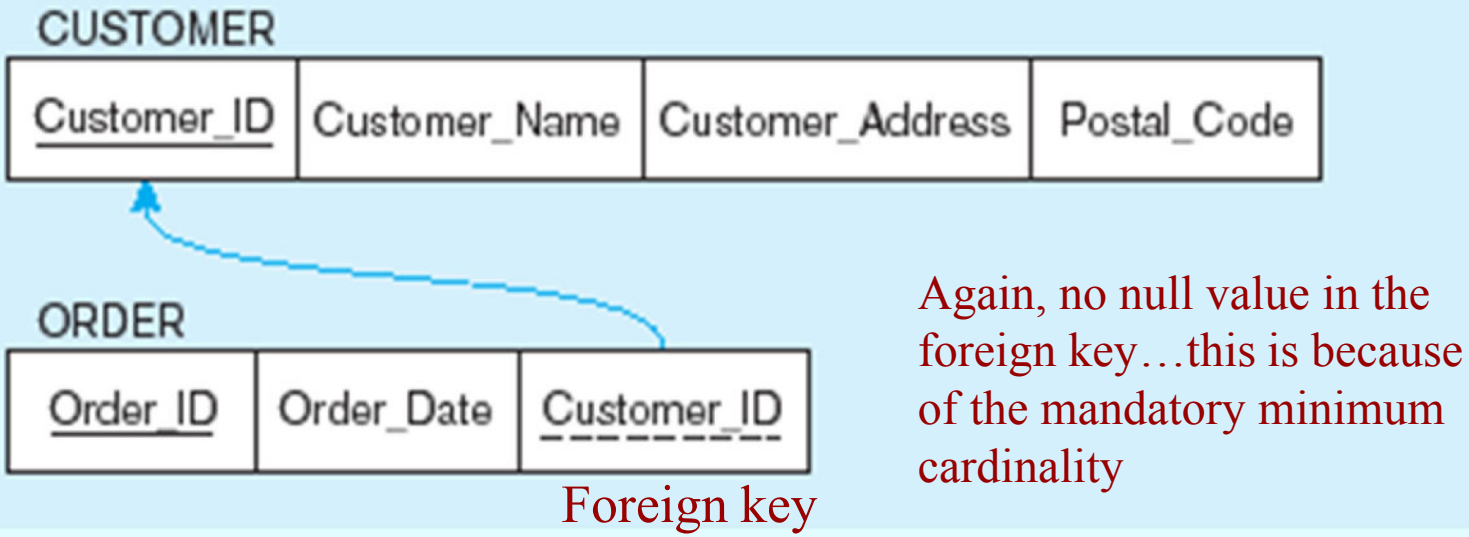
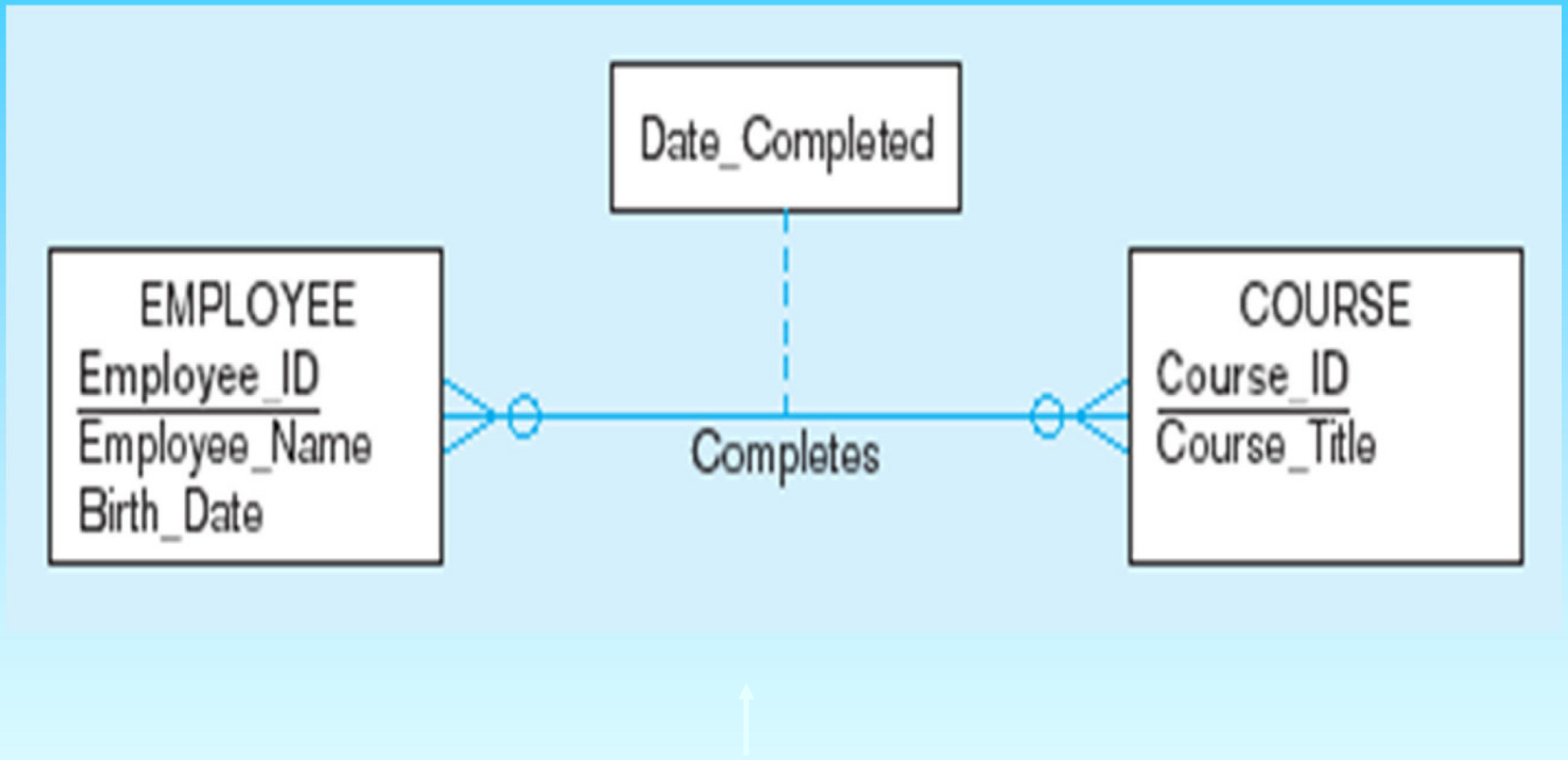


Figure 5-13 Example of mapping an M:N relationship

a) Completes relationship (M:N)



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

Figure 5-13 Example of mapping an M:N relationship (cont.)

b) Three resulting relations

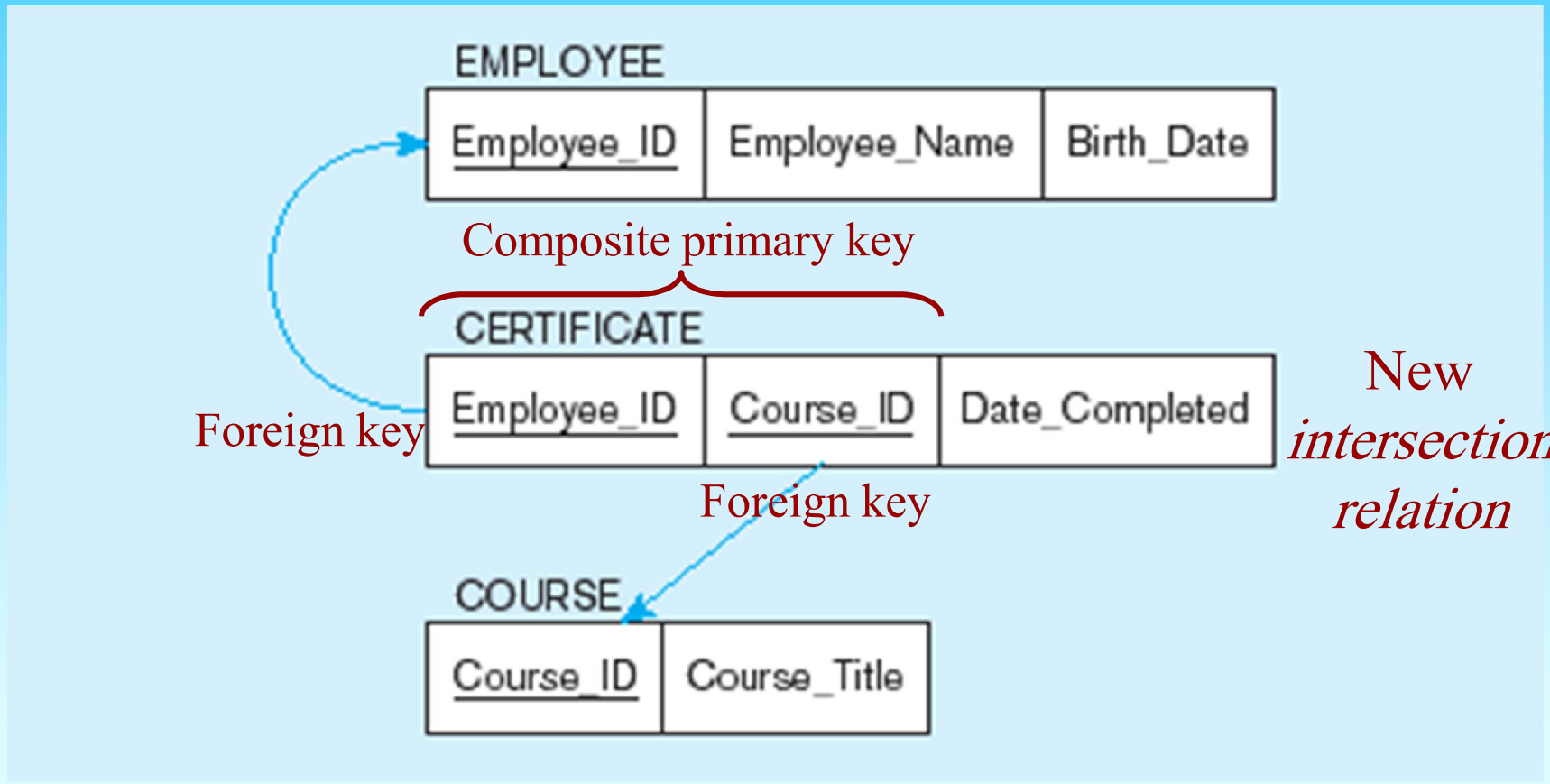
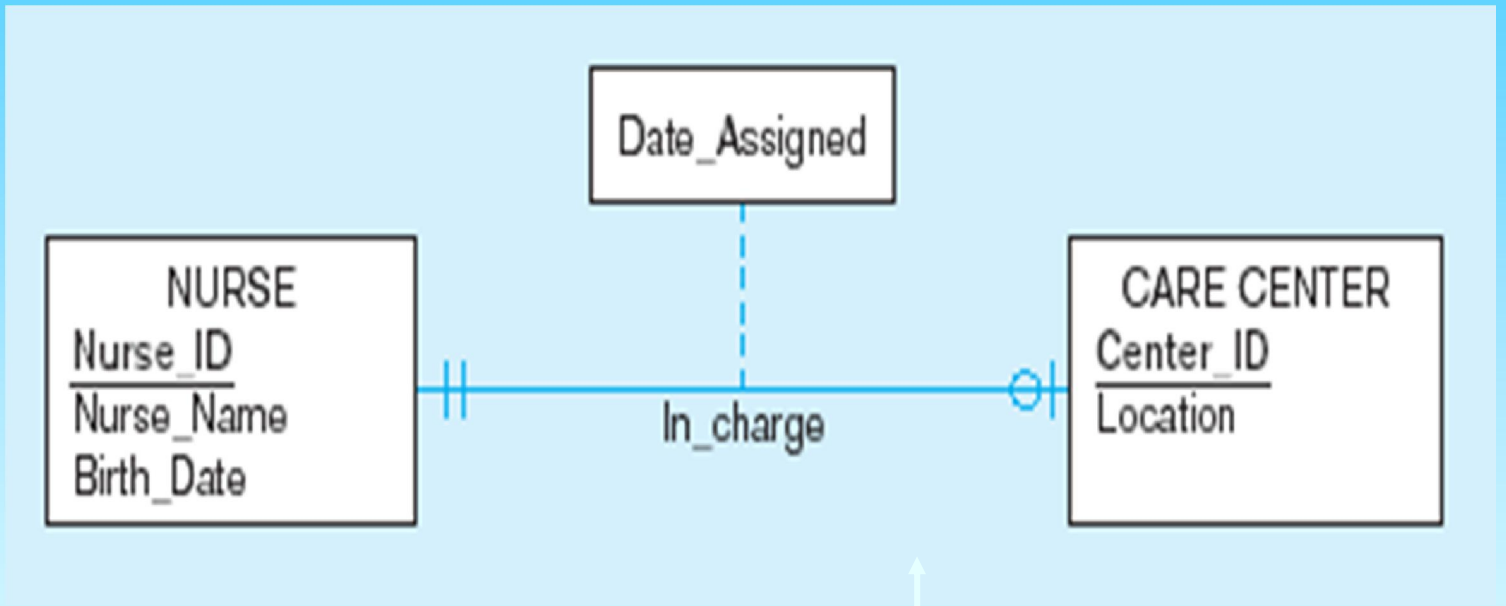


Figure 5-14 Example of mapping a binary 1:1 relationship

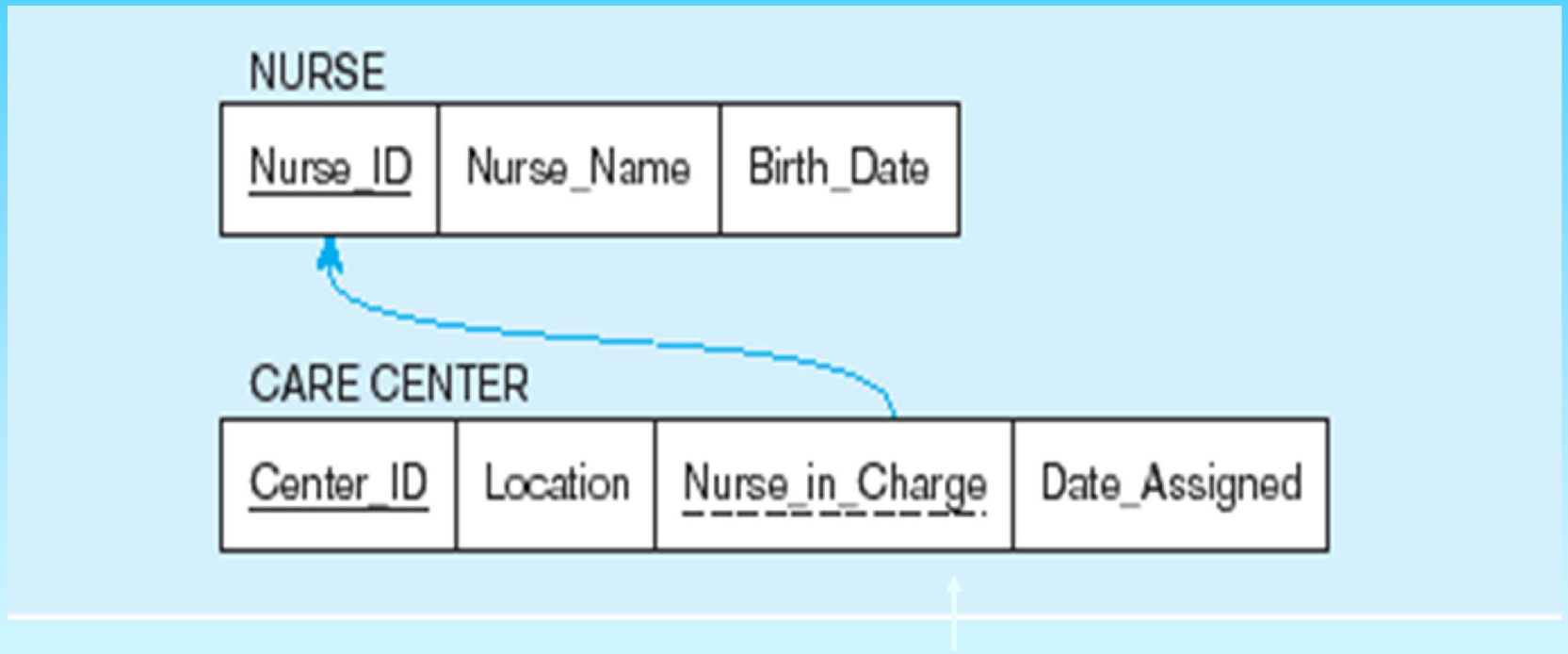
a) In_charge relationship (1:1)



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

Figure 5-14 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

- 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

Figure 5-15 Example of mapping an associative entity

a) An associative entity

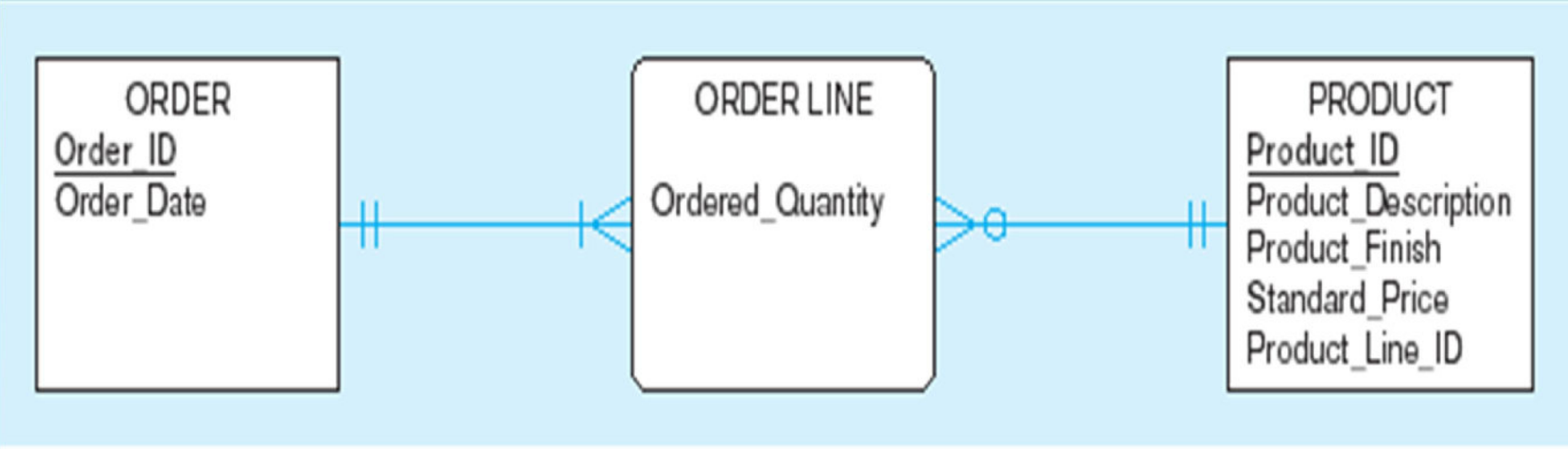


Figure 5-15 Example of mapping an associative entity (cont.)

b) Three resulting relations

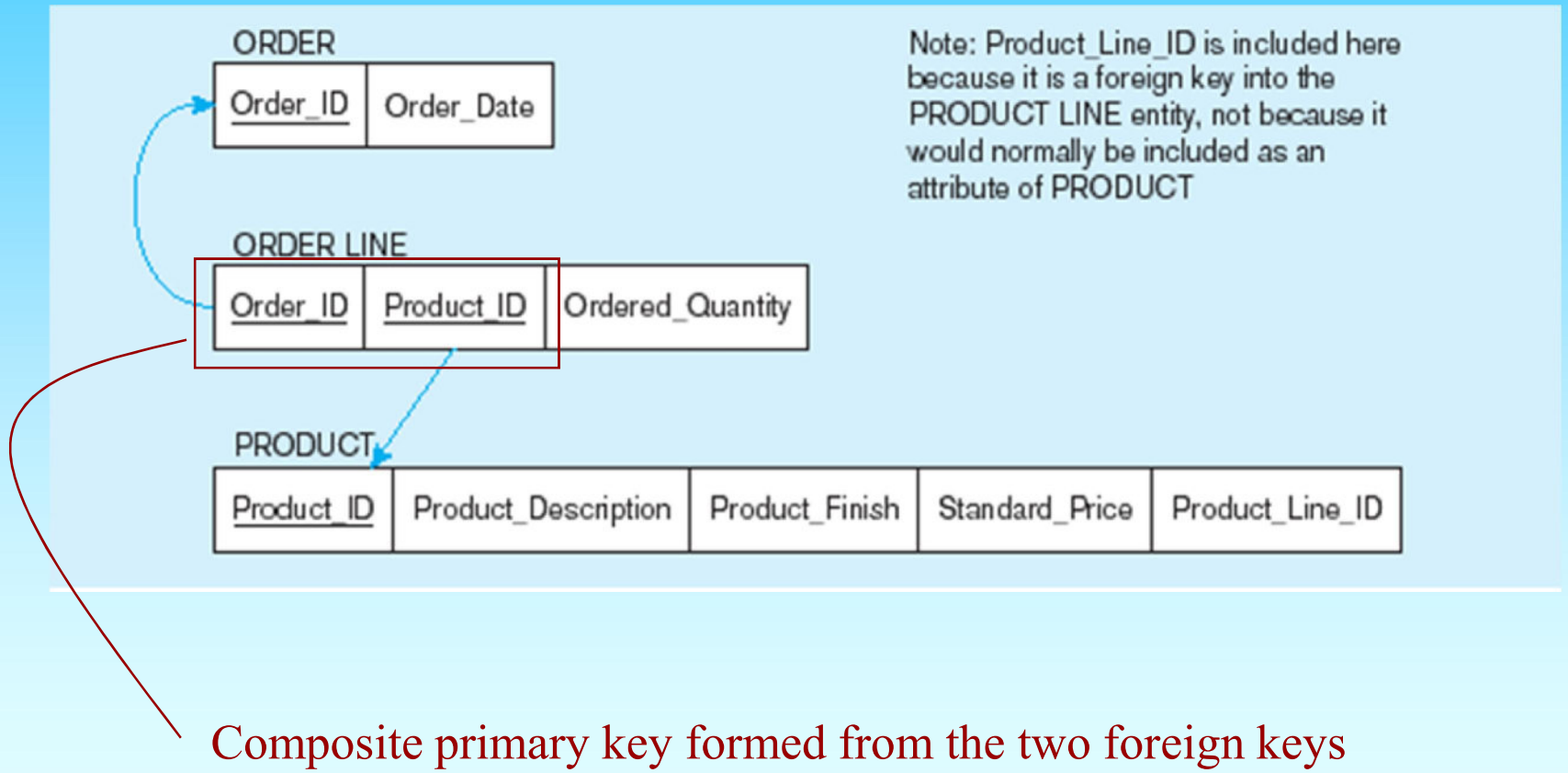
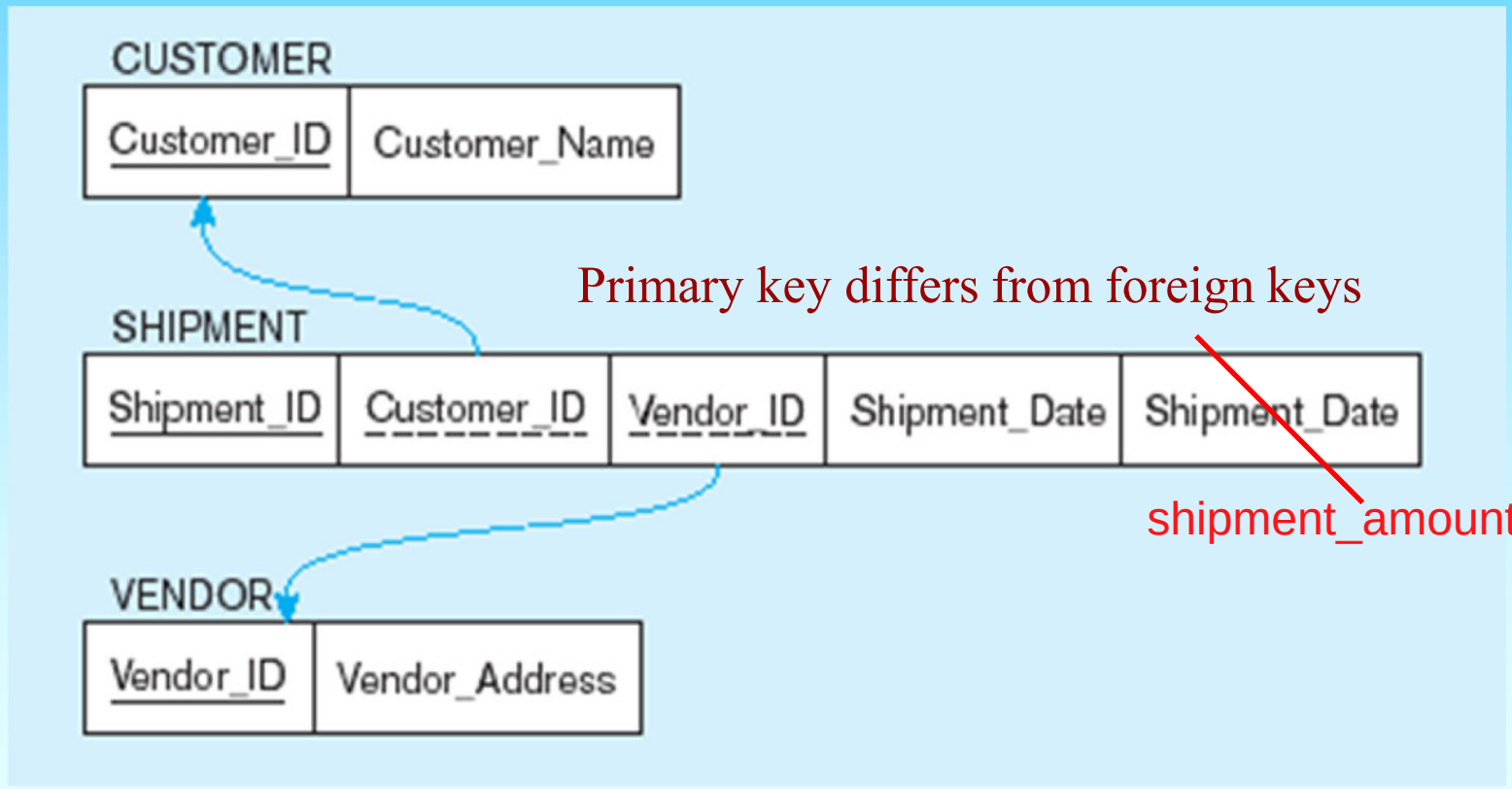


Figure 5-16 Example of mapping an associative entity with an identifier

a) SHIPMENT associative entity

Figure 5-16 Example of mapping an associative entity with an identifier (cont.)

b) Three resulting relations

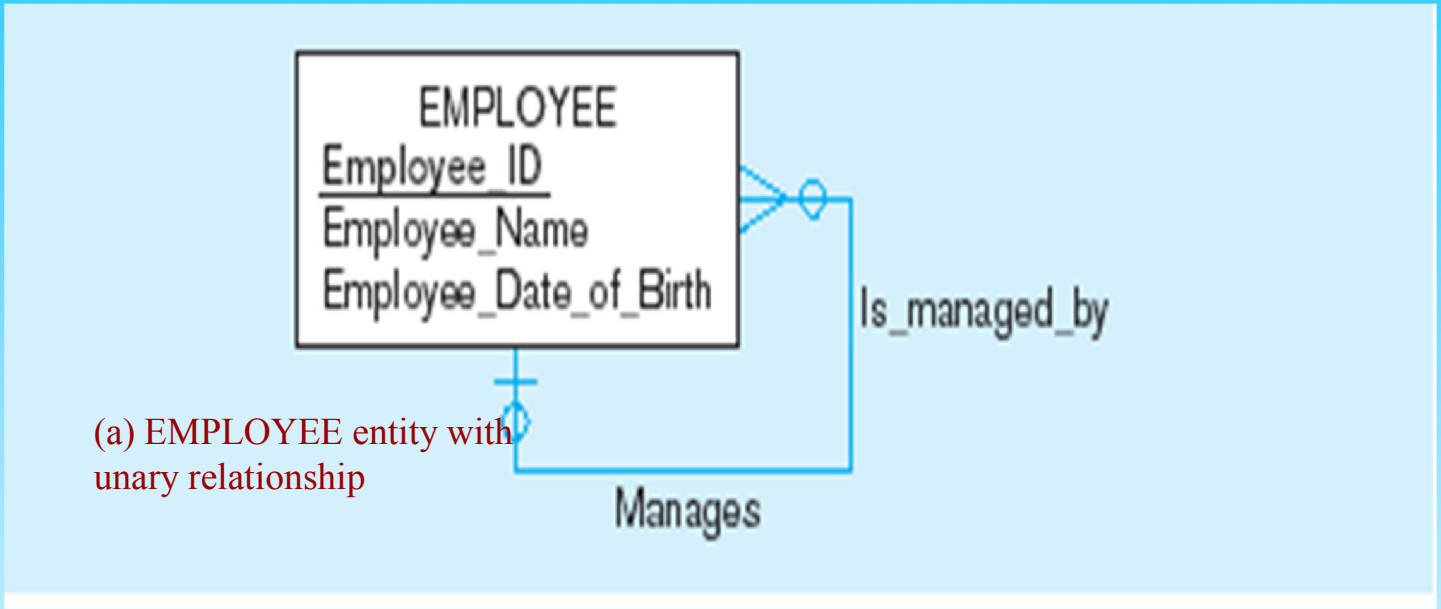


Transforming EER Diagrams into Relations (cont.)

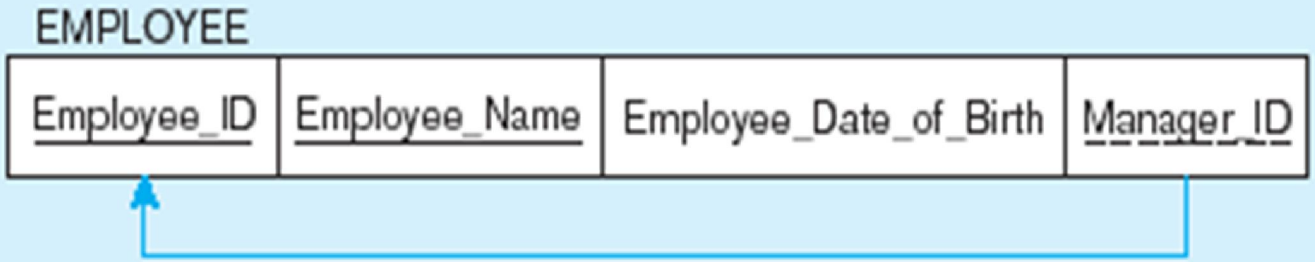
Mapping Unary Relationships

- One-to-Many – 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 5-17 Mapping a unary 1:N relationship

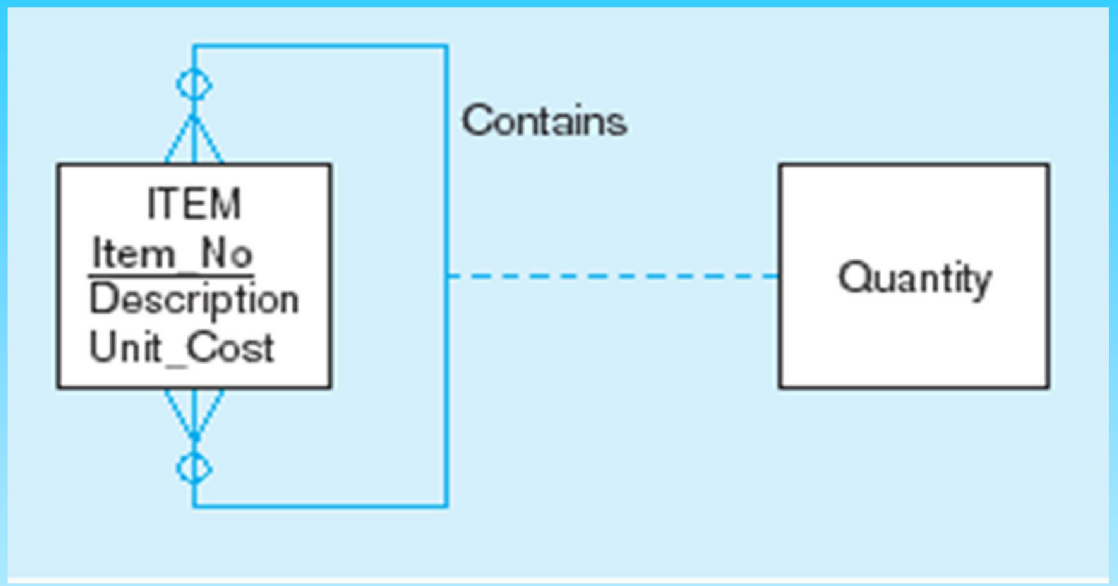


(a) EMPLOYEE entity with unary relationship



(b) EMPLOYEE relation with **recursive foreign key**

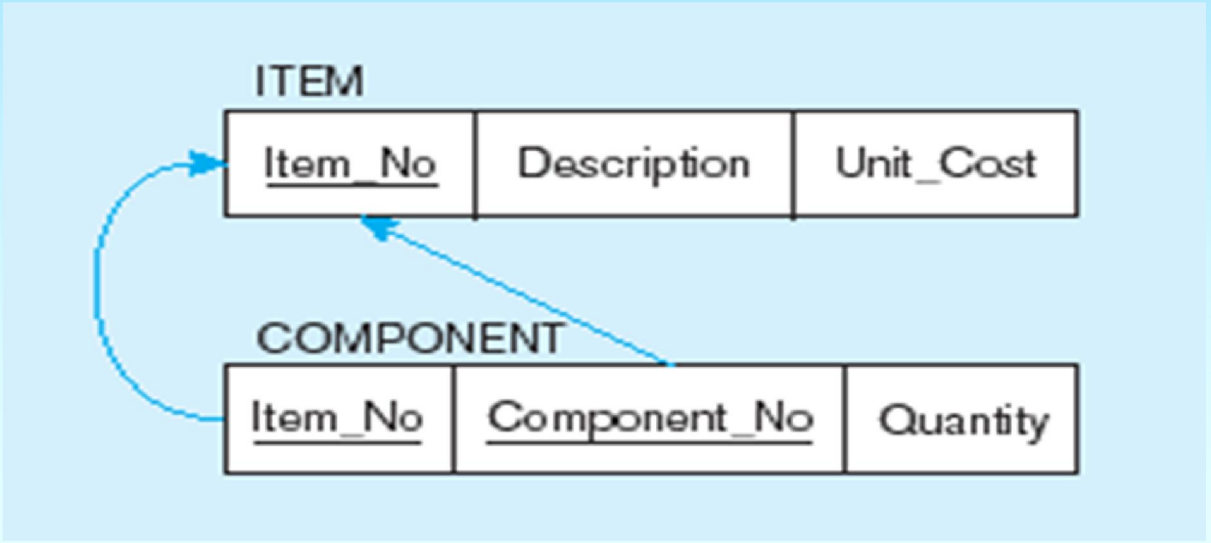
Figure 5-18 Mapping a unary M:N relationship



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

any item can be an item or component of items like in furniture

(b) ITEM and COMPONENT relations



Transforming EER Diagrams into Relations (cont.)

- Mapping Ternary (and n-ary) Relationships
  - One relation for each entity and one for the associative entity
  - Associative entity has foreign keys to each entity in the relationship

Figure 5-19 Mapping a ternary relationship

a) PATIENT TREATMENT Ternary relationship with associative entity

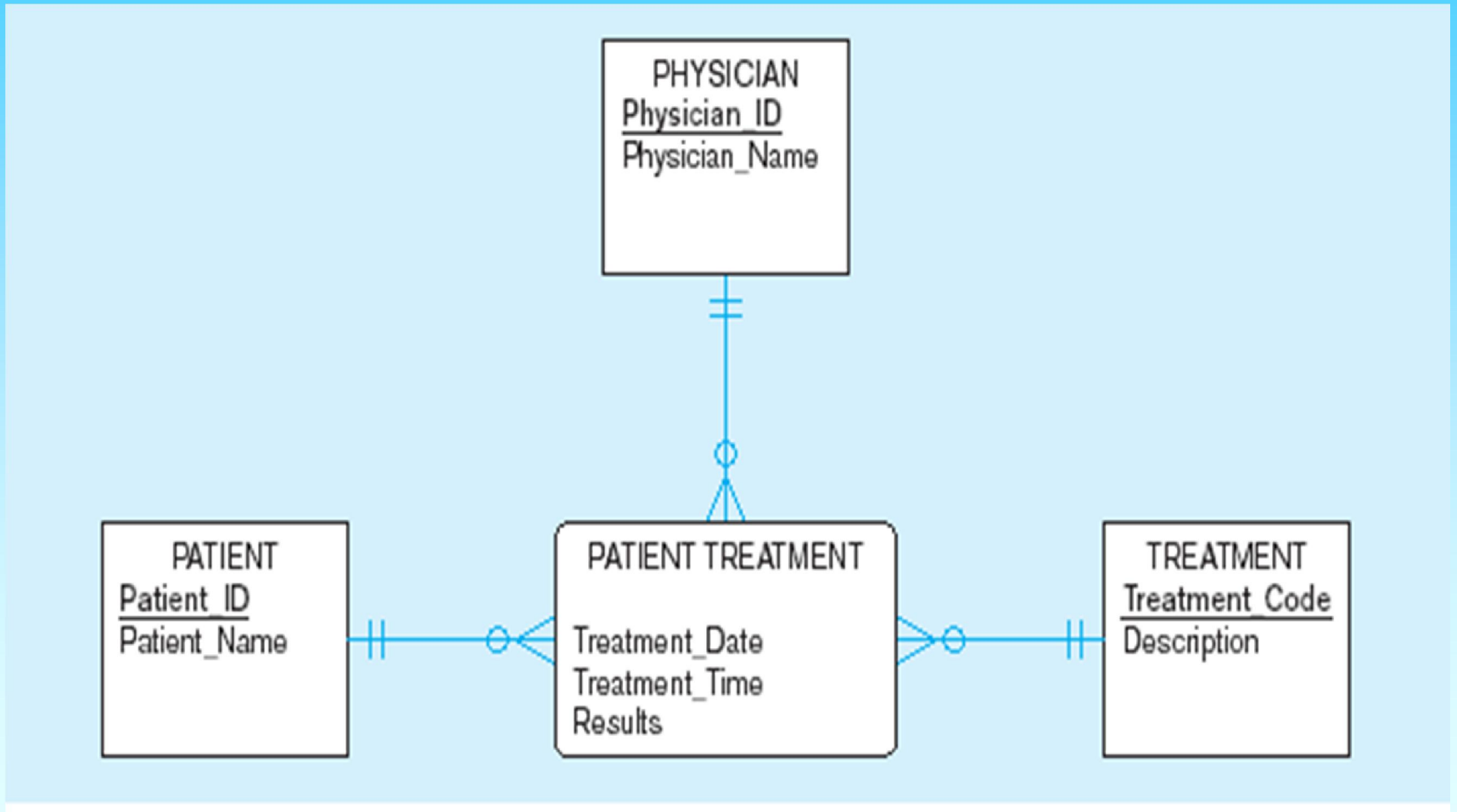
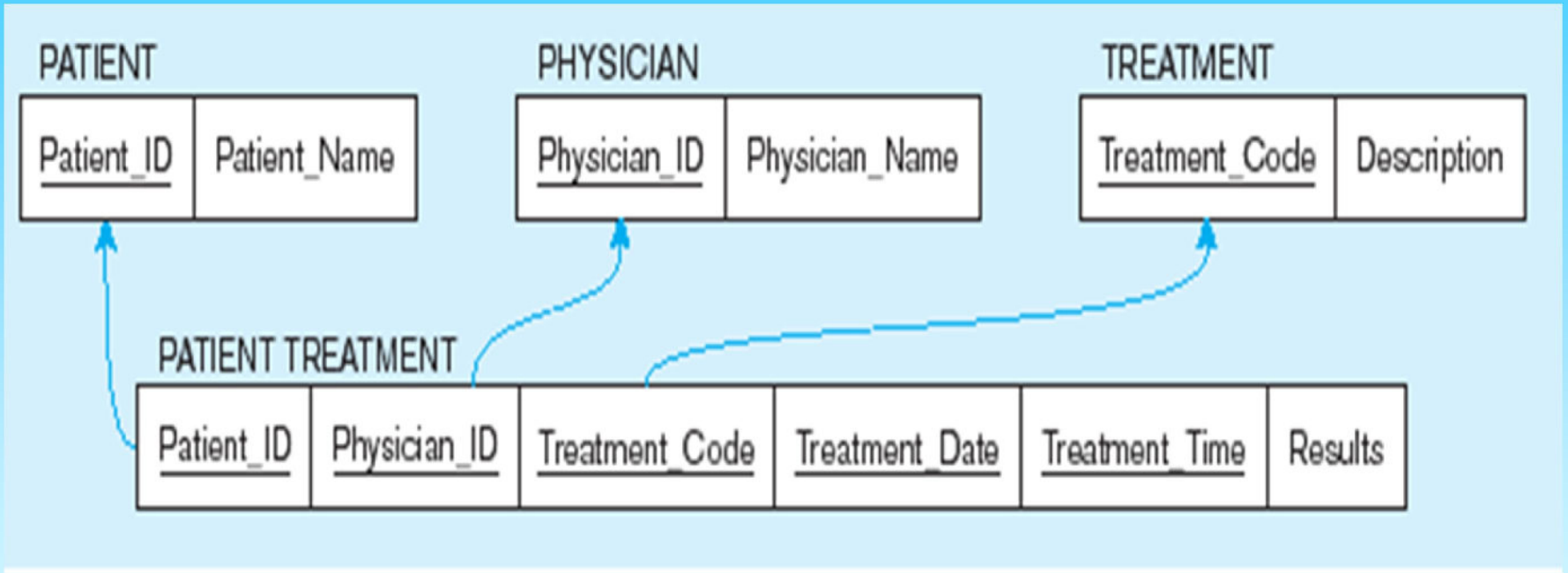


Figure 5-19 Mapping a ternary relationship (cont.)

b) Mapping the ternary relationship PATIENT TREATMENT



Remember that the primary key MUST be unique

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

But this makes a very cumbersome key…

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

Transforming EER Diagrams into Relations (cont.)

Mapping Supertype/Subtype Relationships

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

Figure 5-20 Supertype/subtype relationships

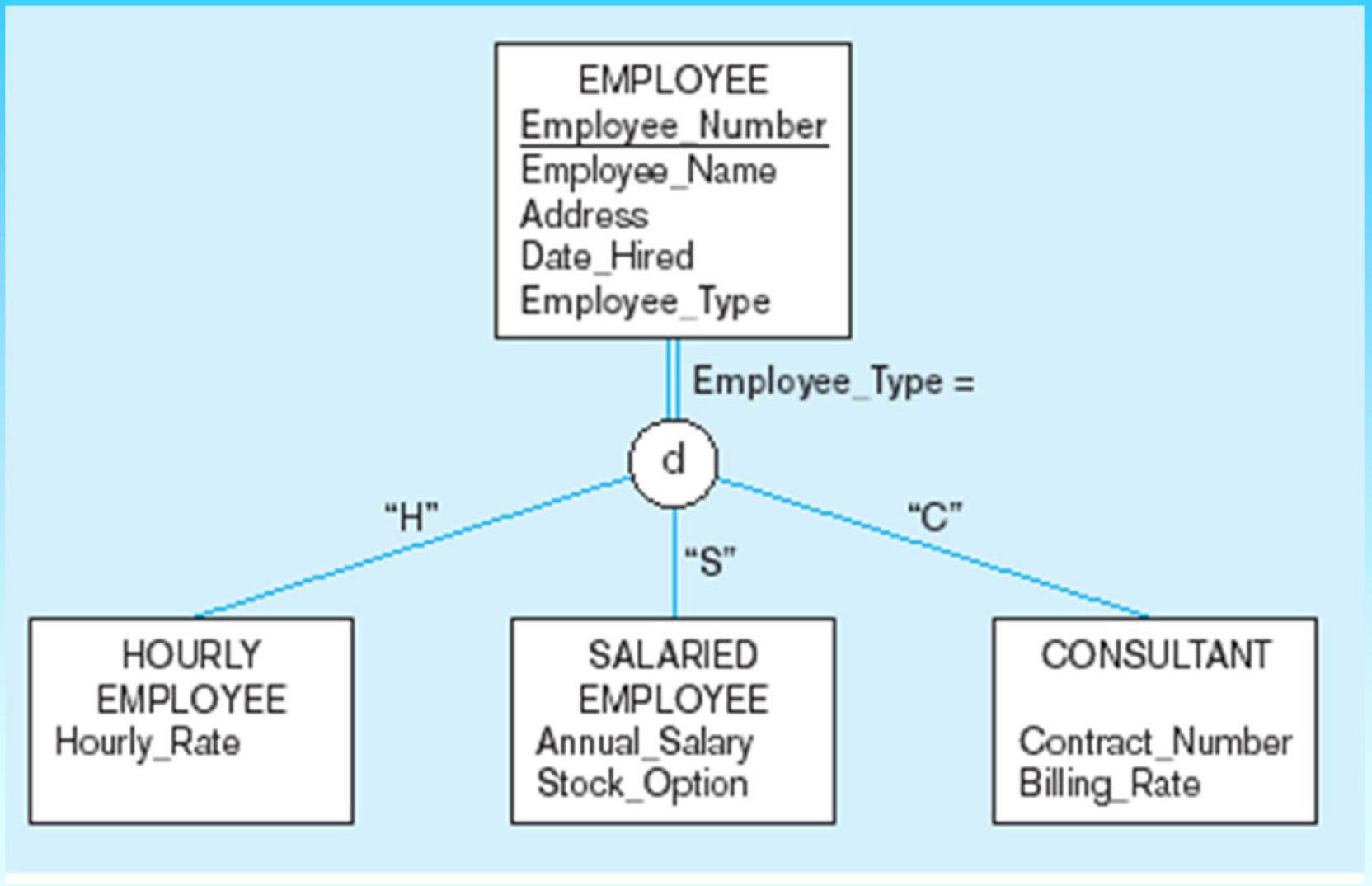
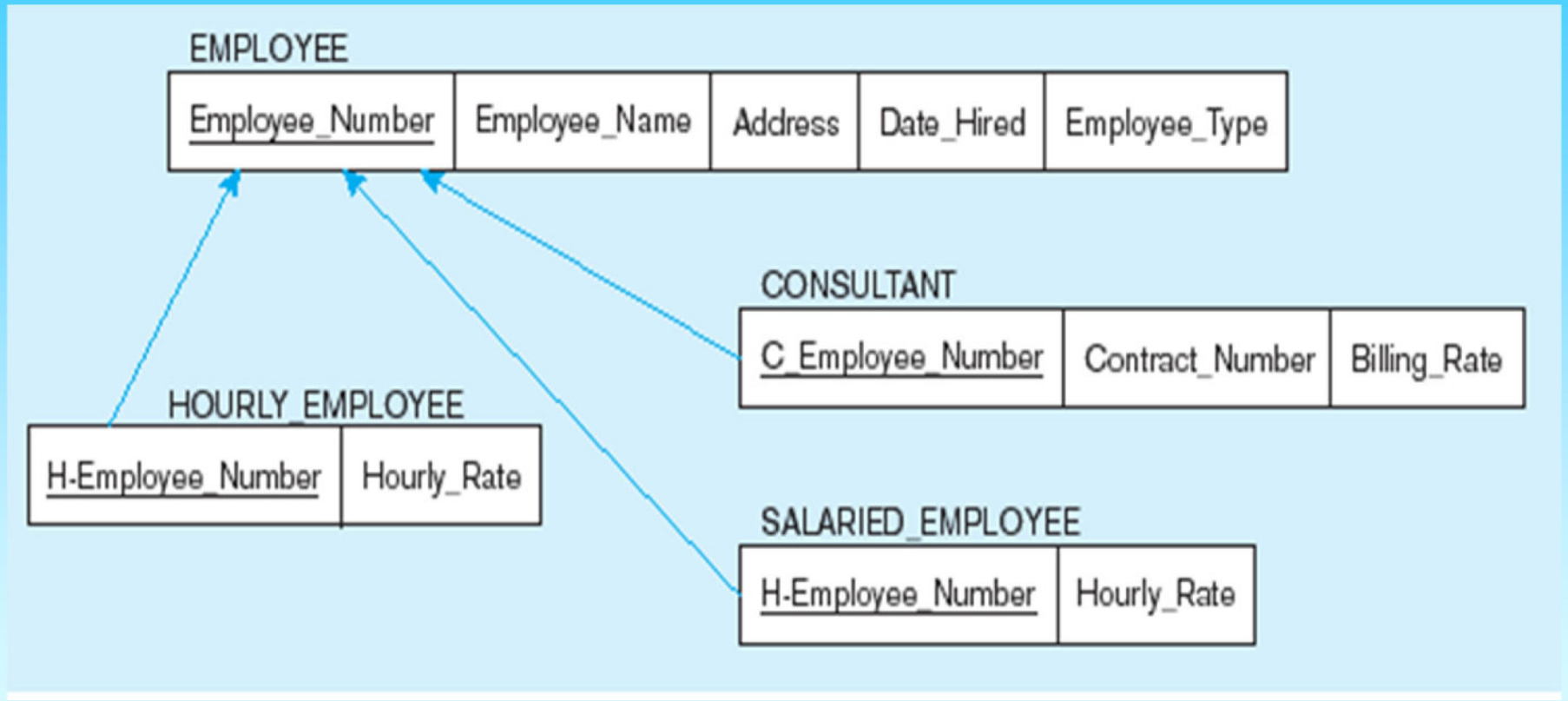


Figure 5-21
Mapping Supertype/subtype relationships to relations



These are implemented as one-to-one relationships