Chapter Five

Logical Database Design

Introduction

- Logical design is the process of constructing a relational model of the information used in an enterprise based on a specific data model
- The first step before applying the rules in relational data model is **converting the conceptual design to a form suitable for relational logical model**, which is in a form of tables.

Logical DB Design and Relational Model

- Relational Data Model
- Transforming E-R Diagrams into Relations
- Normalization

Relation

- **Requirements:**
  - Every relation has a unique name
  - Every attribute value is atomic (not multi-valued, not composite)
  - Every row is unique (can't have two rows with exactly the same values for all their fields)
  - Attributes (columns) in tables have unique names
  - The order of the columns is irrelevant
  - The order of the rows is irrelevant

Key Fields

- Keys are special fields that serve two main purposes:
  - **Primary key** is an attribute that uniquely identifies each row in a relation
  - **Foreign key** is an attribute in a relation of a database that serves as the primary key of another relation in the same database
- Keys can be **simple** (a single attribute) or **composite** (more than one attribute)

Anomalies (1)

- Well-structured relations
  - a relation that contains minimal data redundancy and allows users to insert, delete, and modify rows without causing data inconsistencies (or anomalies)
- The purpose of normalization is to reduce the chances for anomalies to occur in a database.

Anomalies (2)

- Types of 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/updating anomaly**
    - changing data in a row forces changes to other rows because of duplication

Example

EMPLOYEE2

Emp_ID	Name	Dept_Name	Salary	Course_Title	Date_Completed
100	Margaret Simpson	Marketing	48,000	SPSS	6/19/200X
100	Margaret Simpson	Marketing	48,000	Surveys	10/7/200X
140	Alan Beeton	Accounting	52,000	Tax Acc	12/8/200X
110	Chris Lucero	Info Systems	43,000	Visual Basic	1/12/200X
110	Chris Lucero	Info Systems	43,000	C++	4/22/200X
190	Lorenzo Davis	Finance	55,000		
150	Susan Martin	Marketing	42,000	SPSS	6/19/200X
150	Susan Martin	Marketing	42,000	Java	8/12/200X

- **Insertion** – can't enter a new employee without having the employee take a class.
- **Deletion** – if we delete 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.

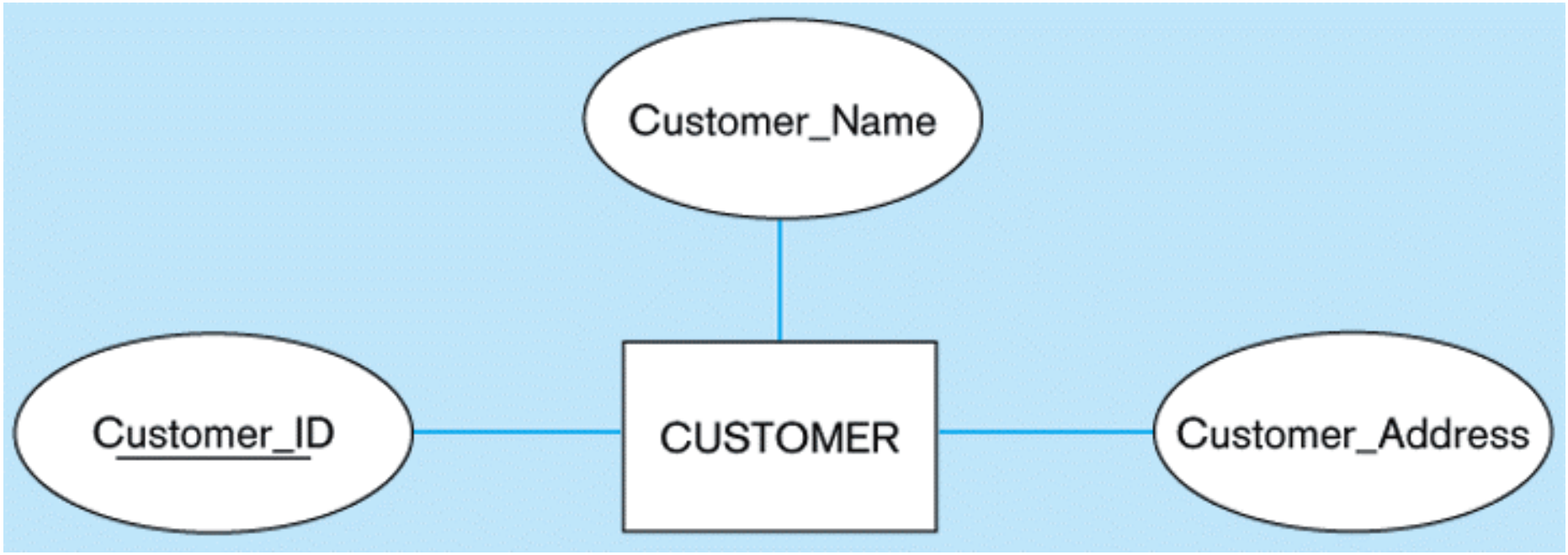
ERD to Relational Table(1)

1. Mapping Regular Entities to Relations

- Simple attributes
  - E-R attributes map directly onto the relation
- Composite attributes
  - use only their simple, component attributes
- Multi-valued attribute
  - becomes a separate relation with a foreign key taken from the original entity

Mapping a regular entity

(a) CUSTOMER entity type with simple attributes



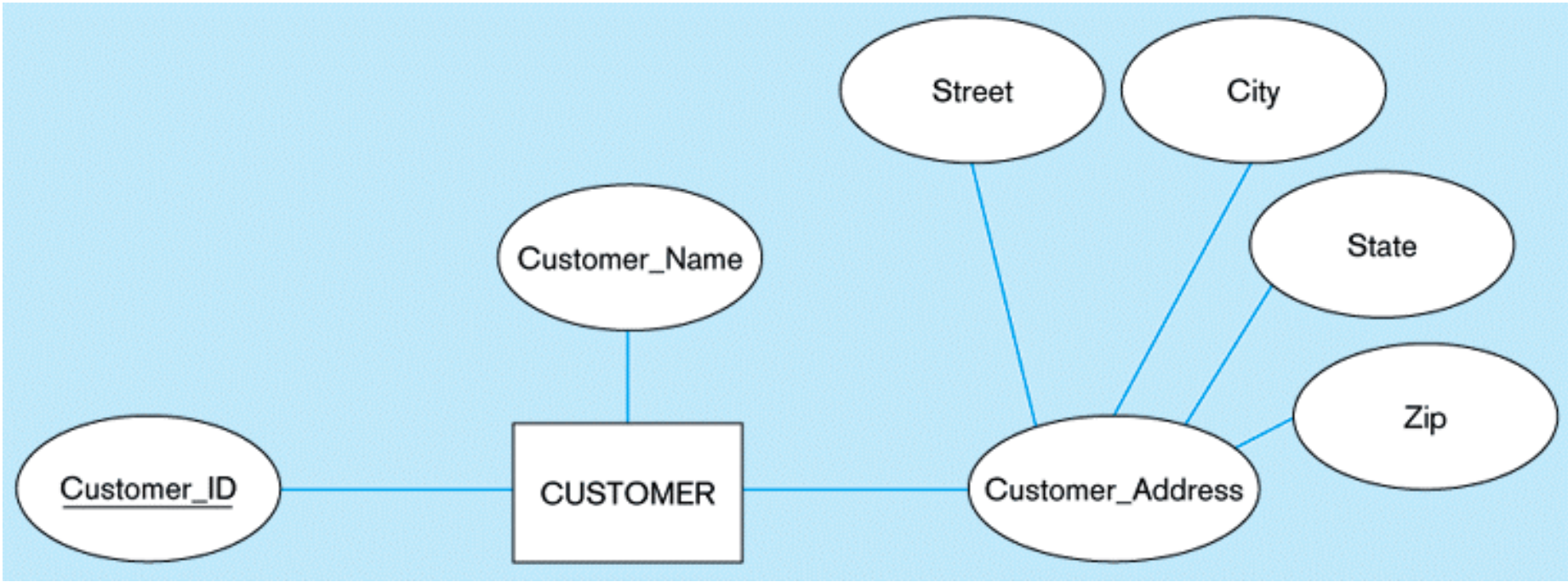
(b) CUSTOMER relation

CUSTOMER

Customer_ID	Customer_Name	Customer_Address

Mapping a composite attribute

(a) CUSTOMER entity type with composite attribute

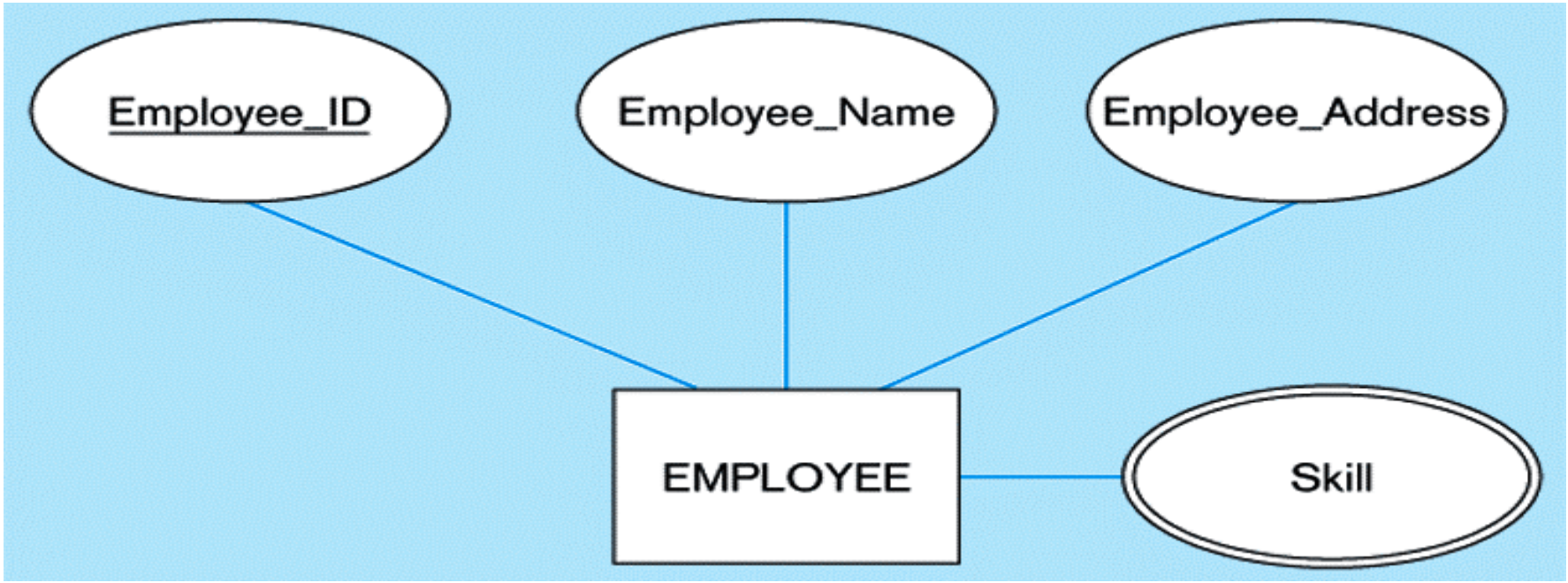


(b) CUSTOMER relation with address detail

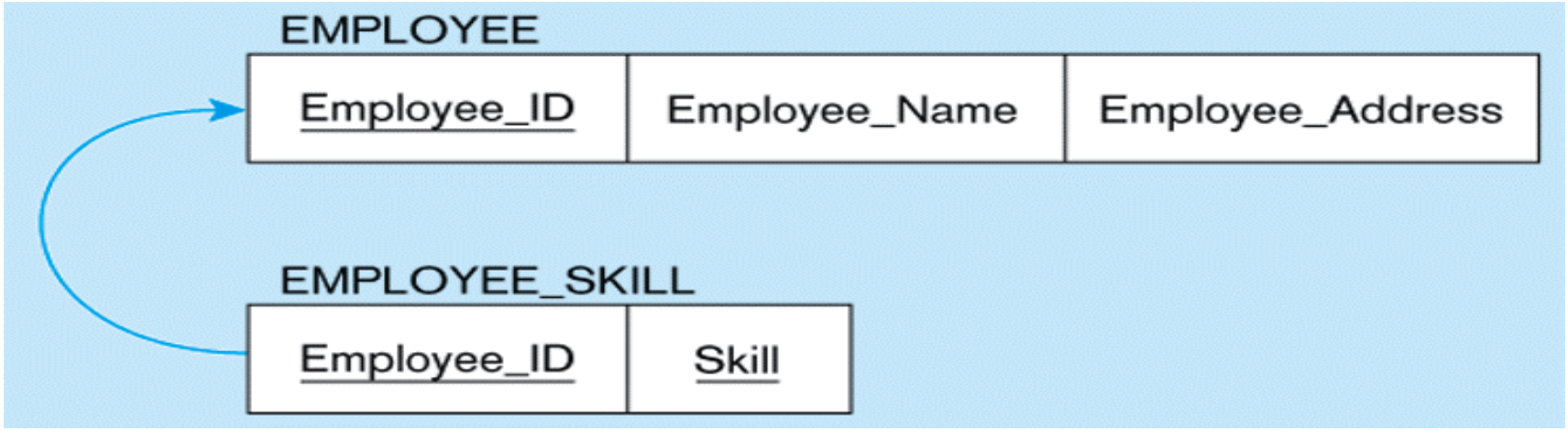
CUSTOMER

Customer_ID	Customer_Name	Street	City	State	Zip

Mapping multi-valued attribute



Multi-valued attribute becomes a separate relation with foreign key



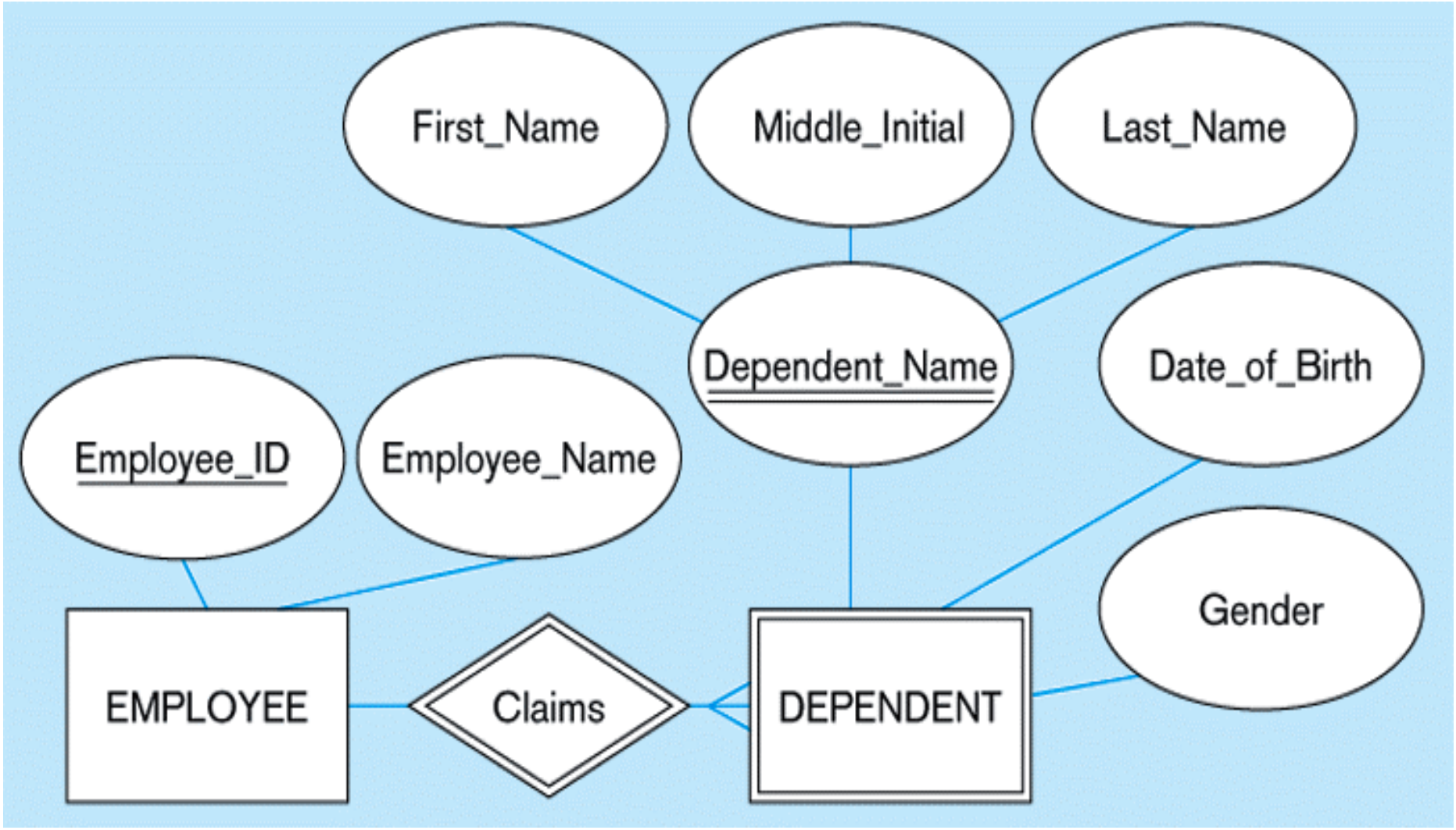
ERD to Relational Table(2)

2. Mapping Weak Entities

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

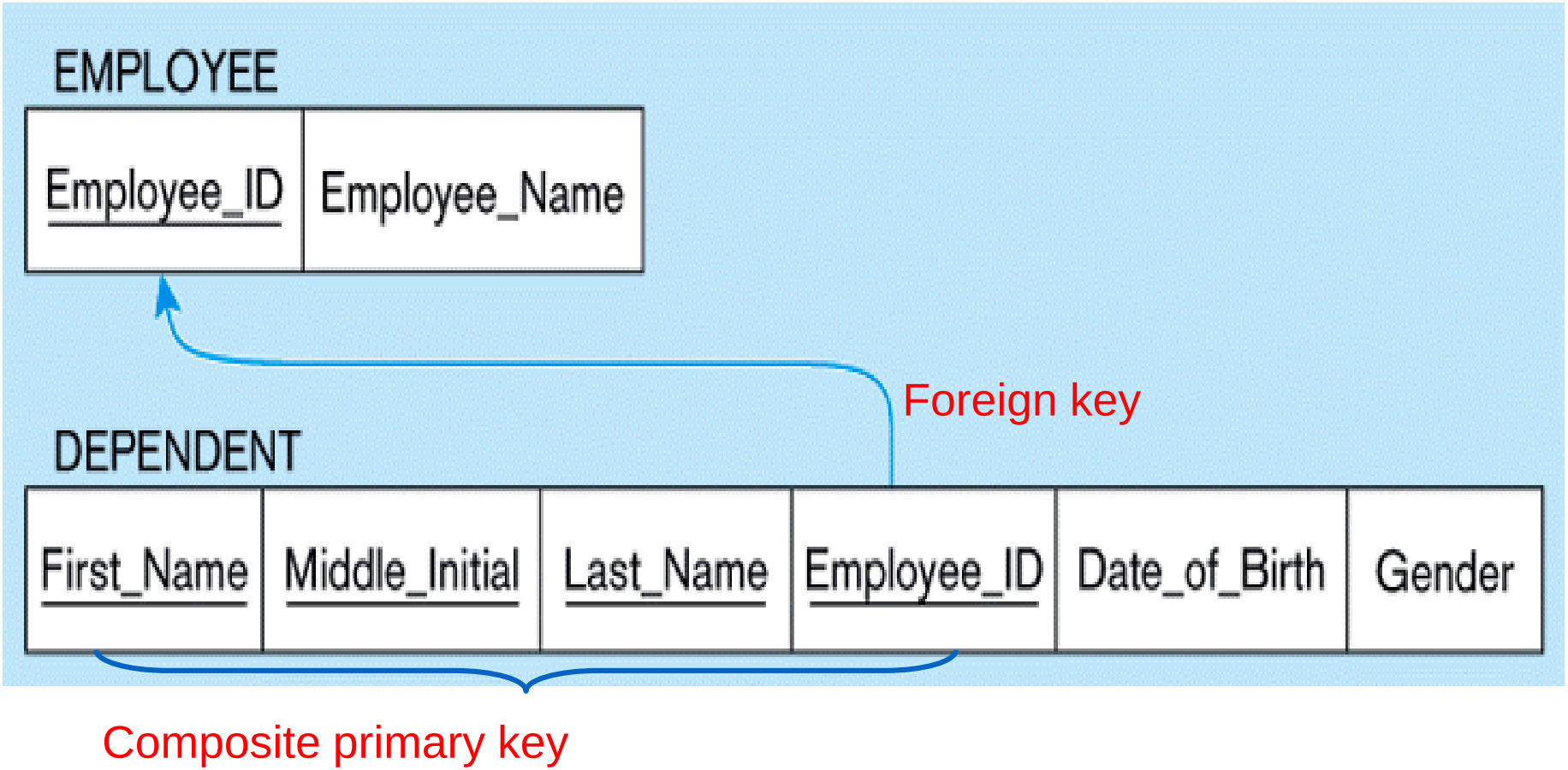
Mapping a weak entity(1)

(a) Weak entity DEPENDENT



Mapping a weak entity(2)

(b) Relations resulting from weak entity



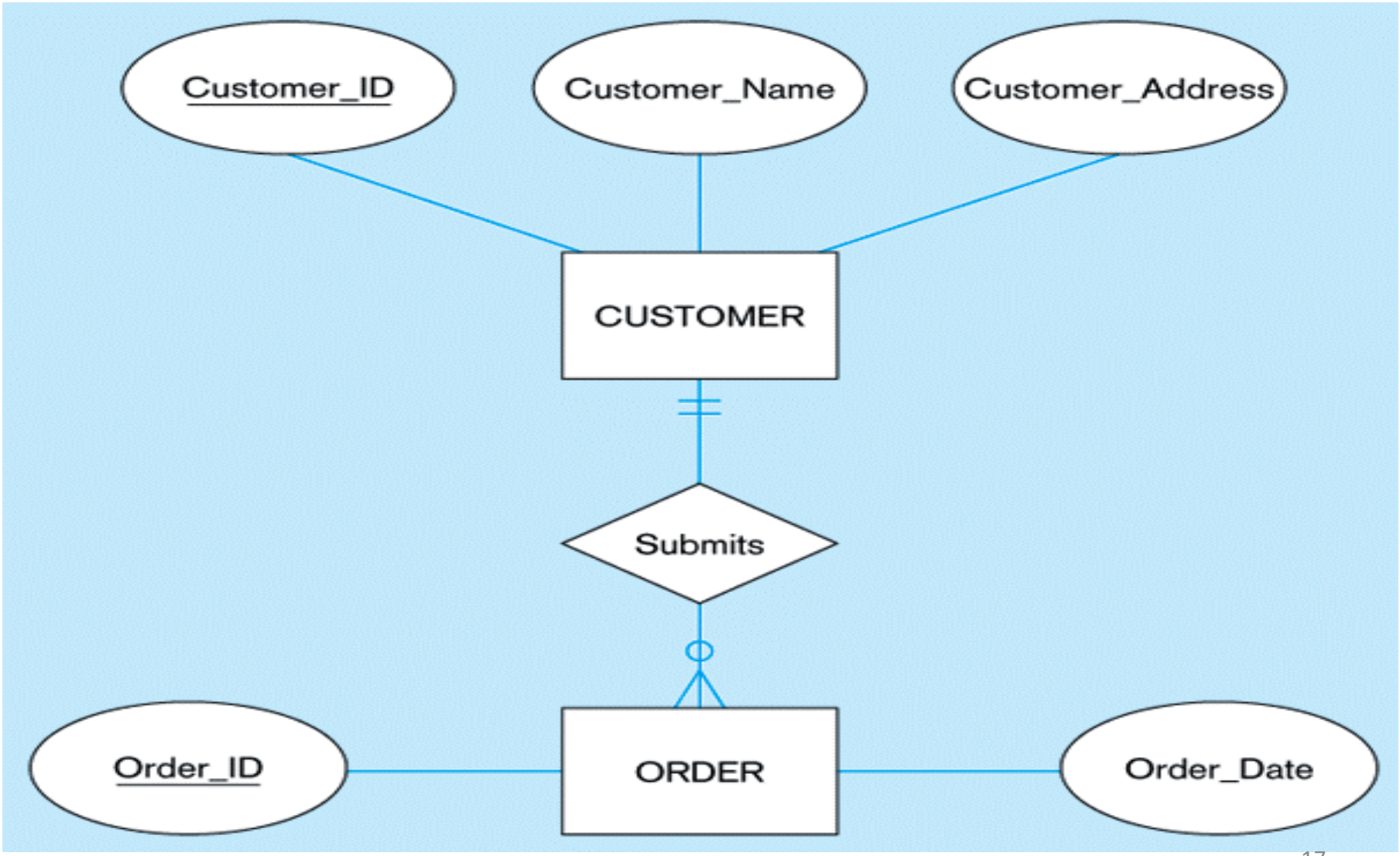
ERD to Relational Table(3)

3. 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
  - avoids the need to store null values in the foreign key
  - any attributes associated with the relationship are also included in the same relation as the foreign key

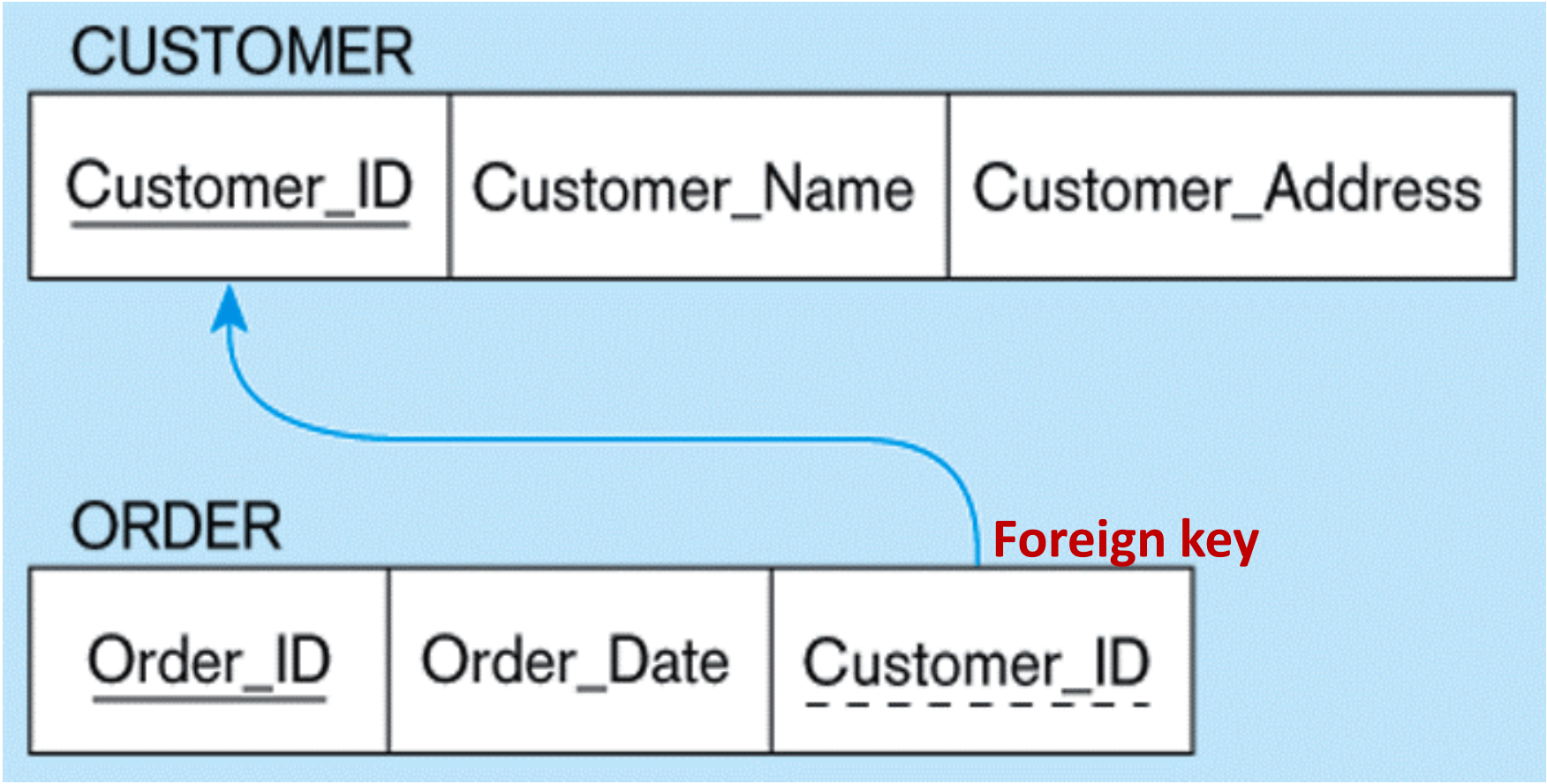
Mapping a 1:M relationship(1)

(a) Relationship between customers and orders



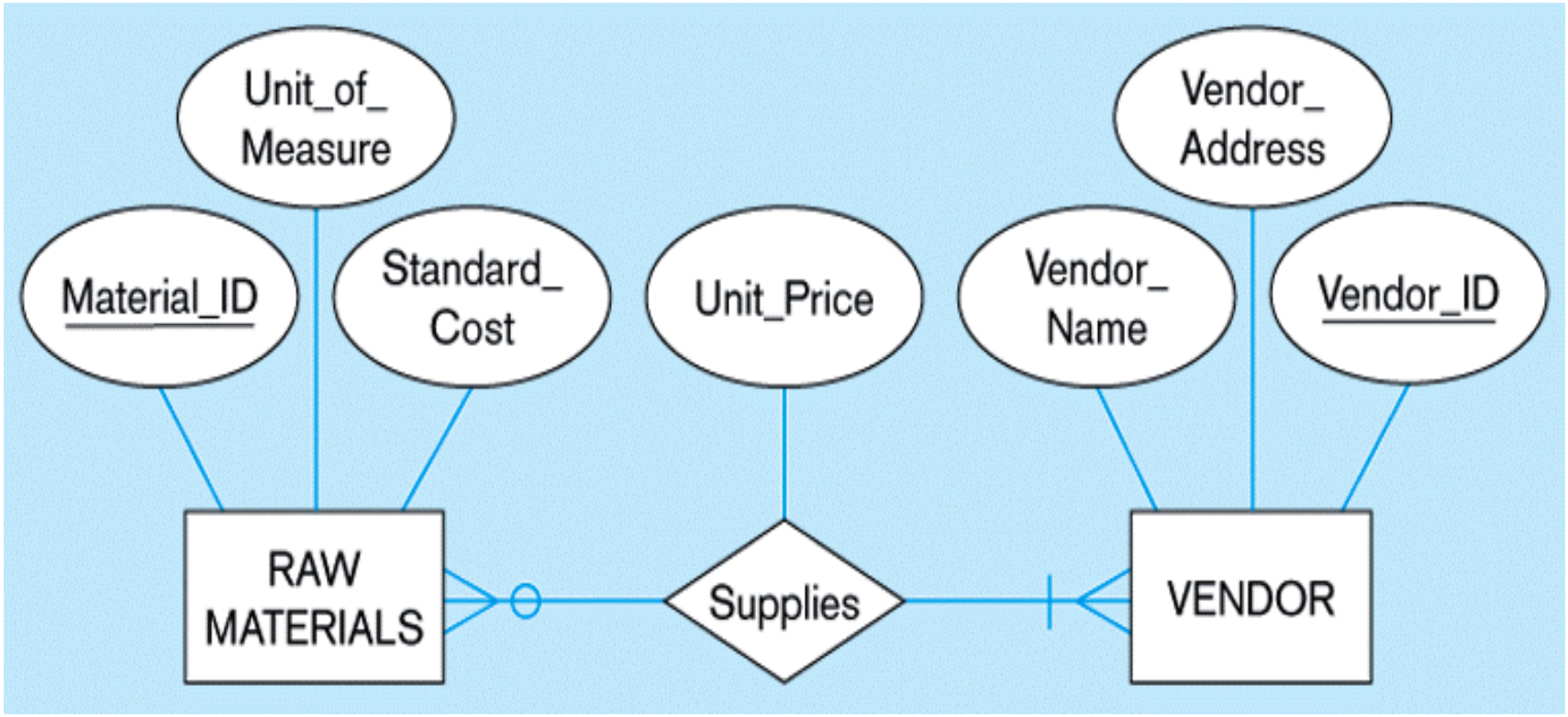
Mapping a 1:M relationship(2)

(b) Mapping the relationship



Mapping an M:N relationship(1)

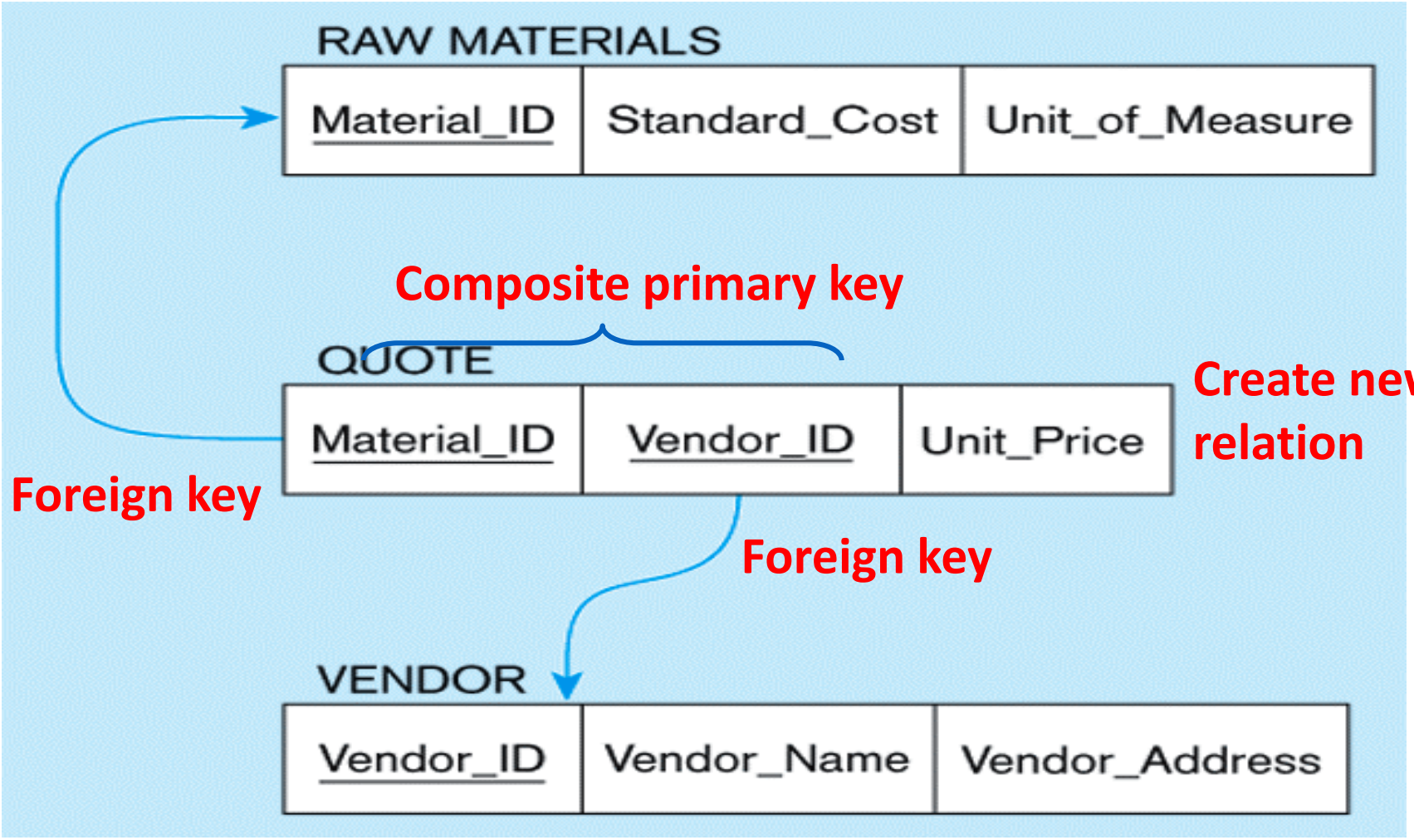
(a) Supplies relationship (M:N)



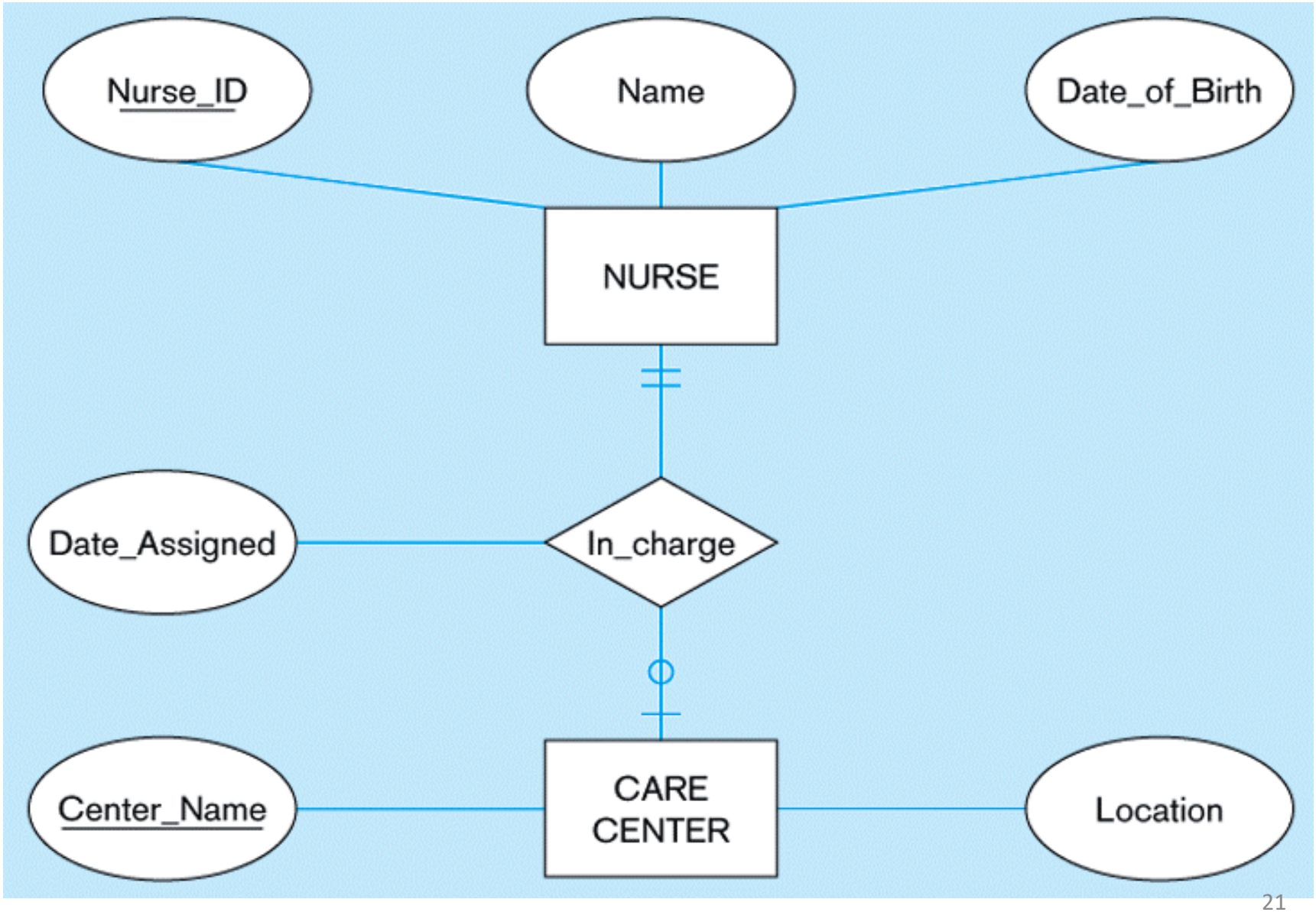
Note: The Supplies relationship will become a separate relation

Mapping an M:N relationship(2)

(b) Three resulting relations

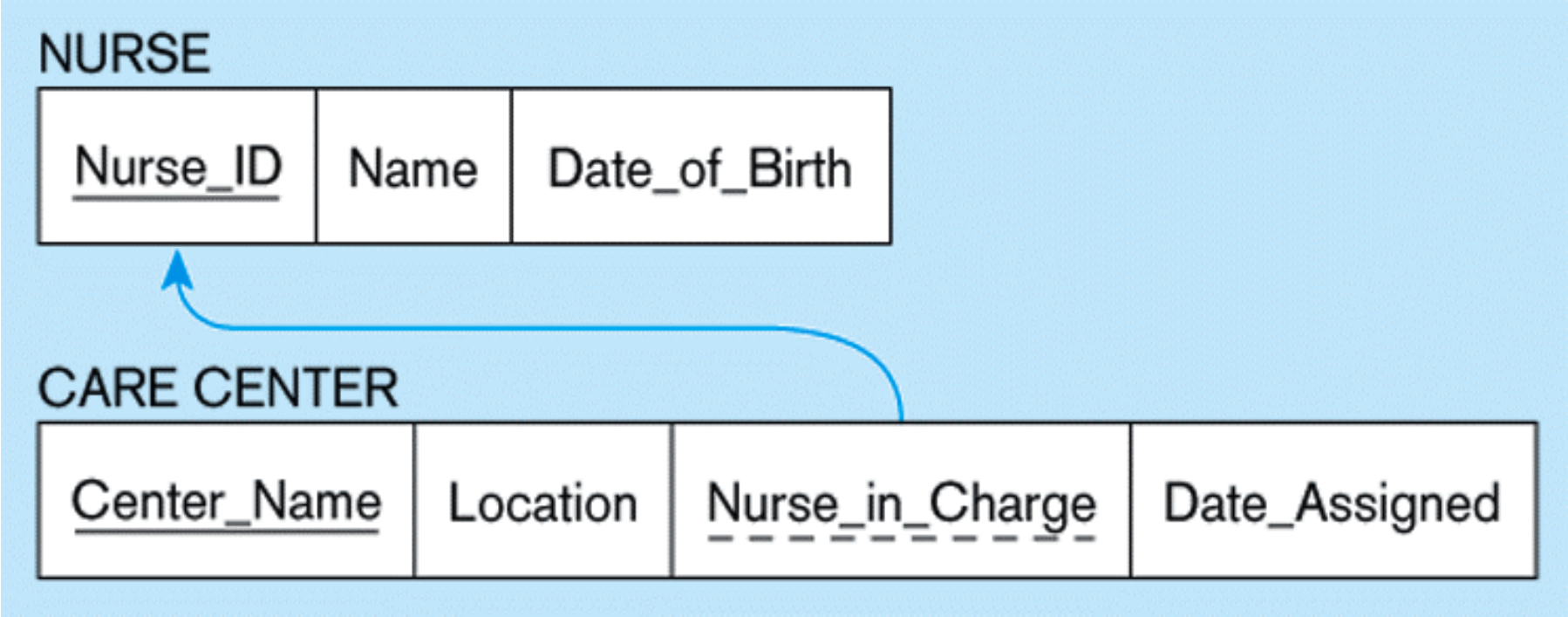


Mapping a binary 1:1 relationship(1)



Mapping a binary 1:1 relationship(2)

(b) Resulting relations



Note:

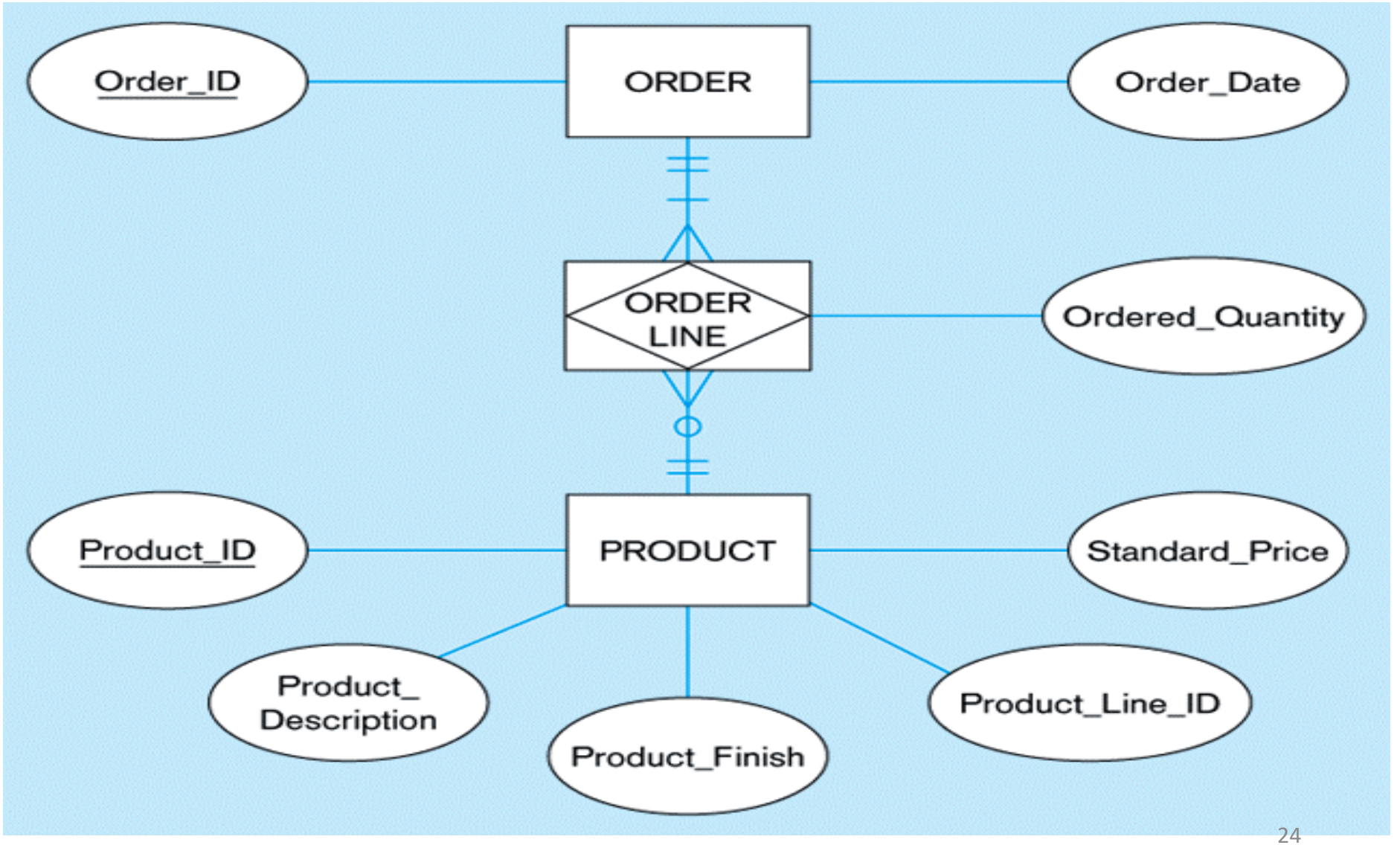
- **Nurse_in_Charge** is another name for **Nurse_ID.**
- Attribute attached to relationship is stored with foreign key.

ERD to Relational Table (4)

4. Mapping Associative Entities

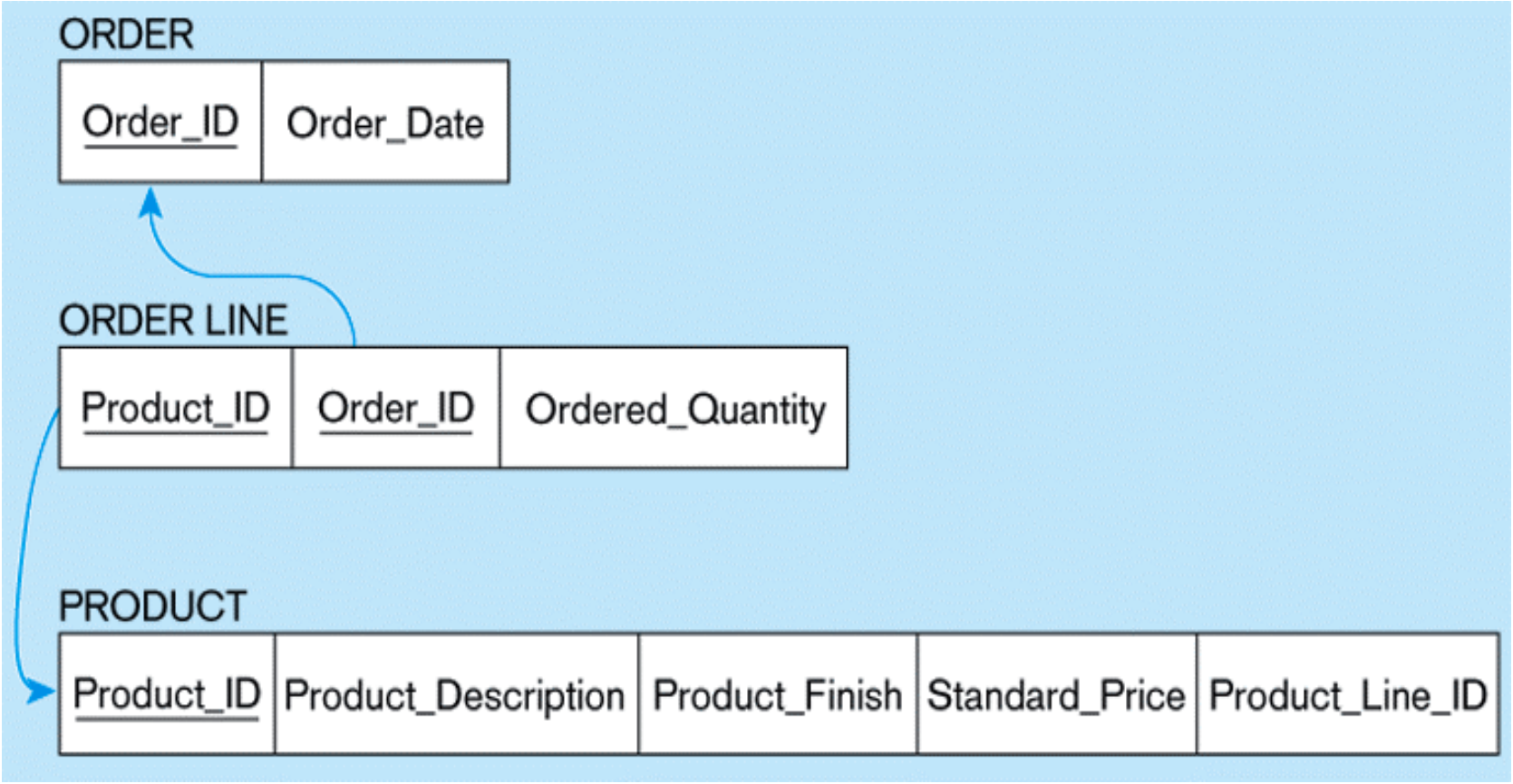
- Identifier not assigned
  - Default primary key for the associative relation is the primary keys of the two entities
- Identifier assigned
  - When it is natural and familiar to end-users
  - Default identifier may not uniquely identify instances of the associative entity

Mapping an associative entity with no assigned identifier (1)



Mapping an associative entity with no assigned identifier (1)

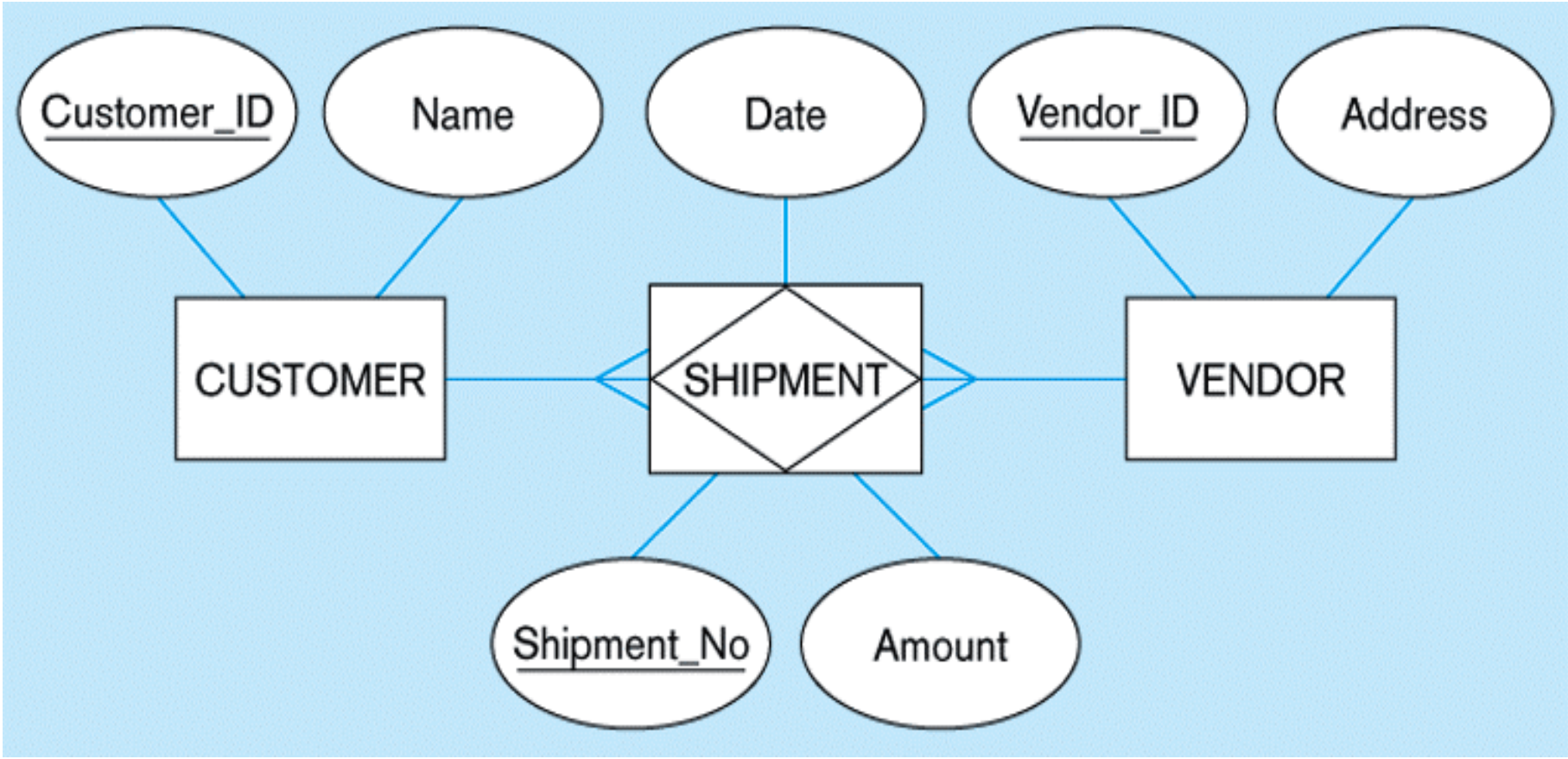
(b) Three resulting relations



Note: Default primary key for the associative relation is the primary keys of the two entities.

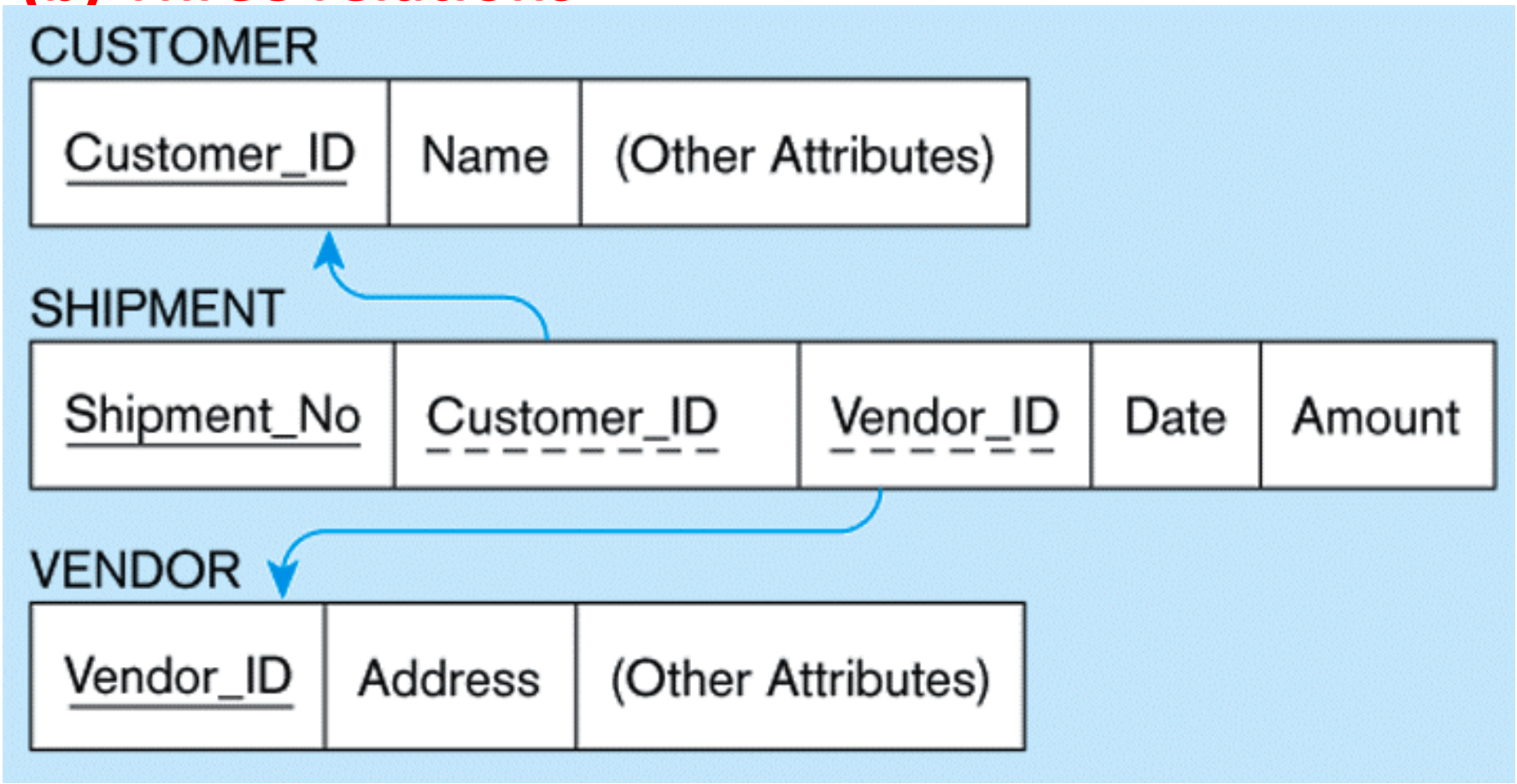
Mapping an associative entity with an identifier(1)

(a) Associative entity



Mapping an associative entity with an identifier(2)

(b) Three relations



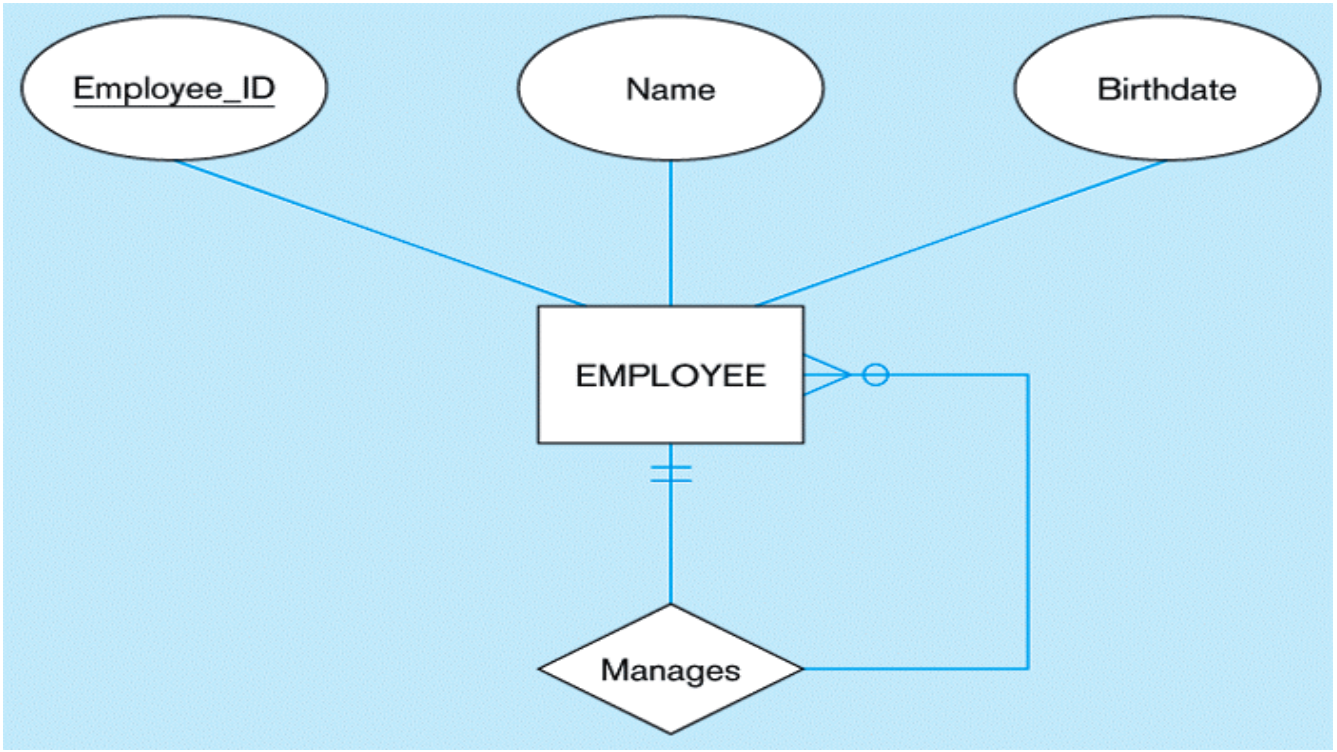
Note: Customer_ID and Vendor_ID together do not uniquely identify the SHIPMENT relation.

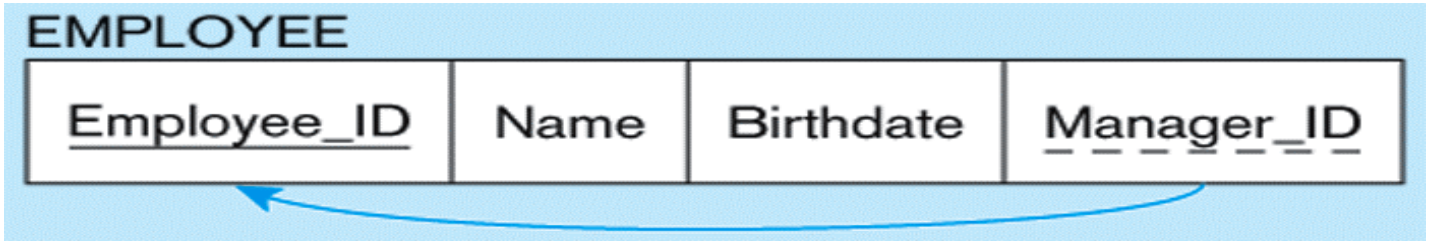
ERD to Relational Table (5)

5. Mapping Unary Relationships

- **One-to-Many**
  - Recursive foreign key in the same relation
  - A foreign key in a relation that references the primary key values of that same relation
- **Many-to-Many**
  - Create two relations
    - ✓One for the entity type
    - ✓One for an associative relation in which the primary key has two attributes, both taken from the primary key of the entity

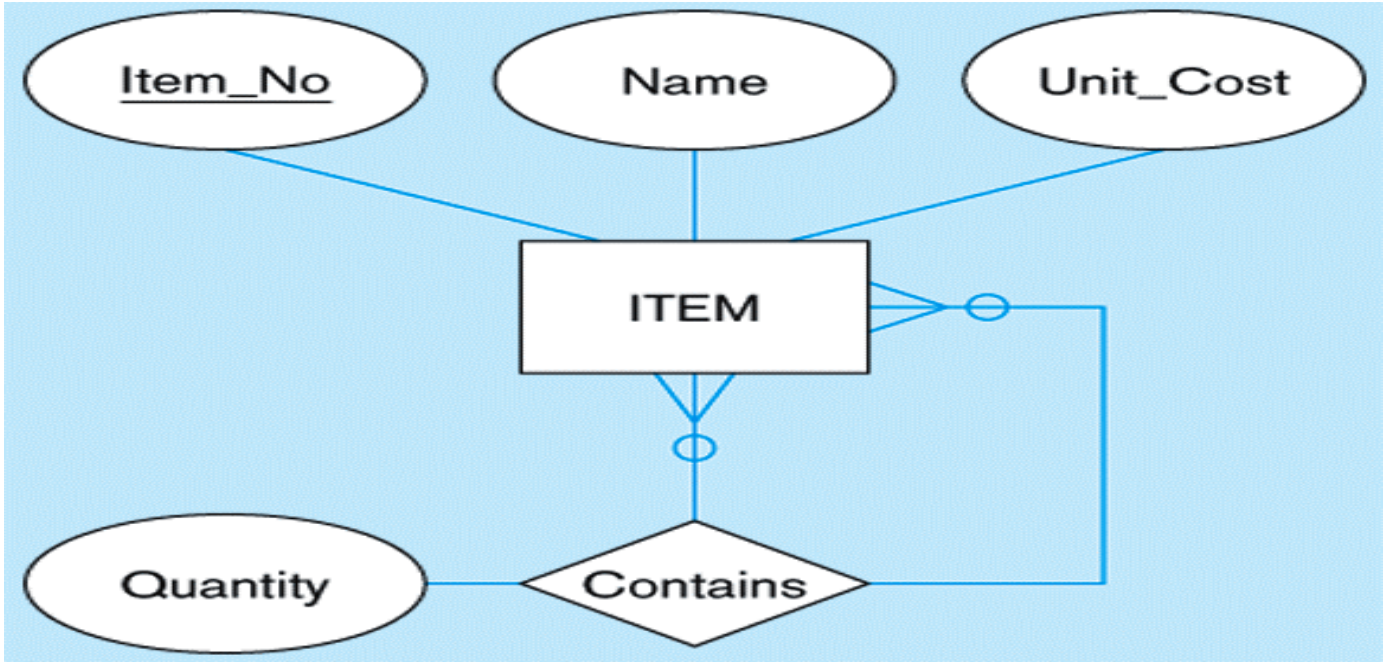
Mapping a unary 1:N relationship

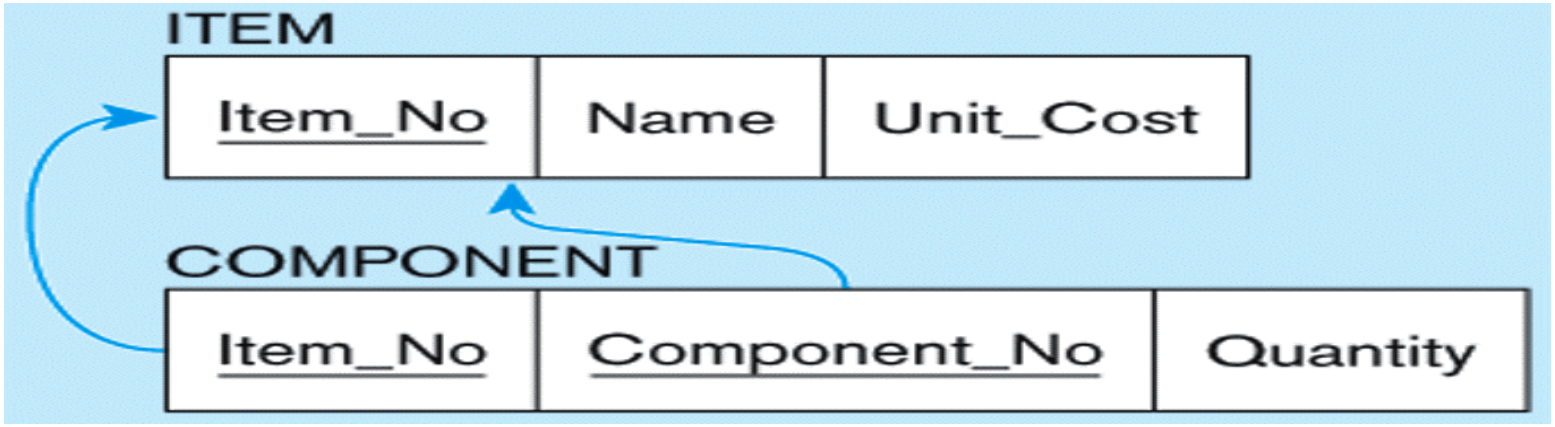




Note: **Manager_ID** and **Employee_ID** refer to the same primary key of the relation. Manager_ID is a foreign key that references itself.

Mapping a unary M:N relationship





Note: Item_No and Component_No refer to the same primary key.

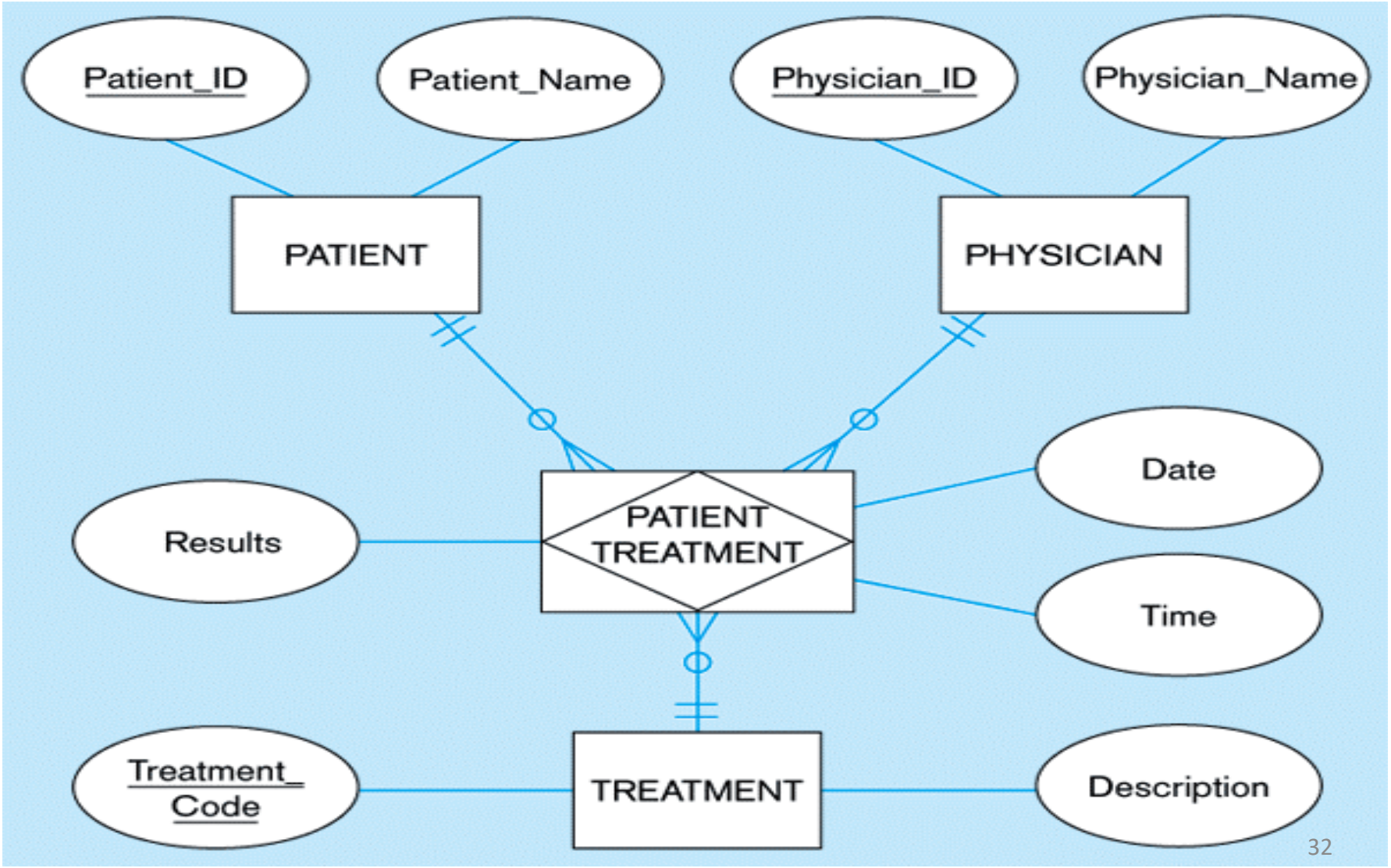
ERD to Relational Table (6)

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

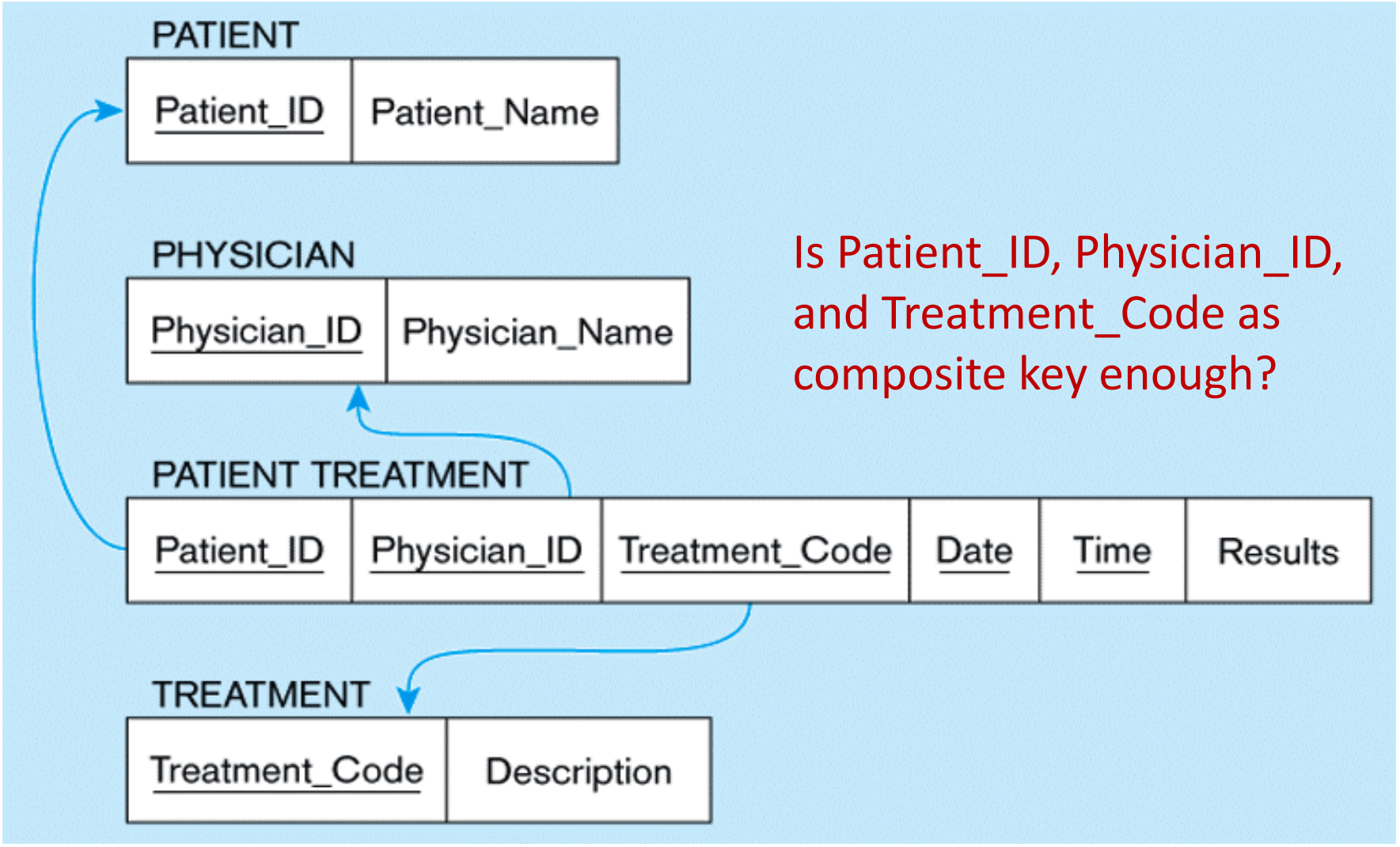
Mapping a ternary relationship(1)

(a) Ternary relationship with associative entity



Mapping a ternary relationship(2)

(b) Mapping the ternary relationship

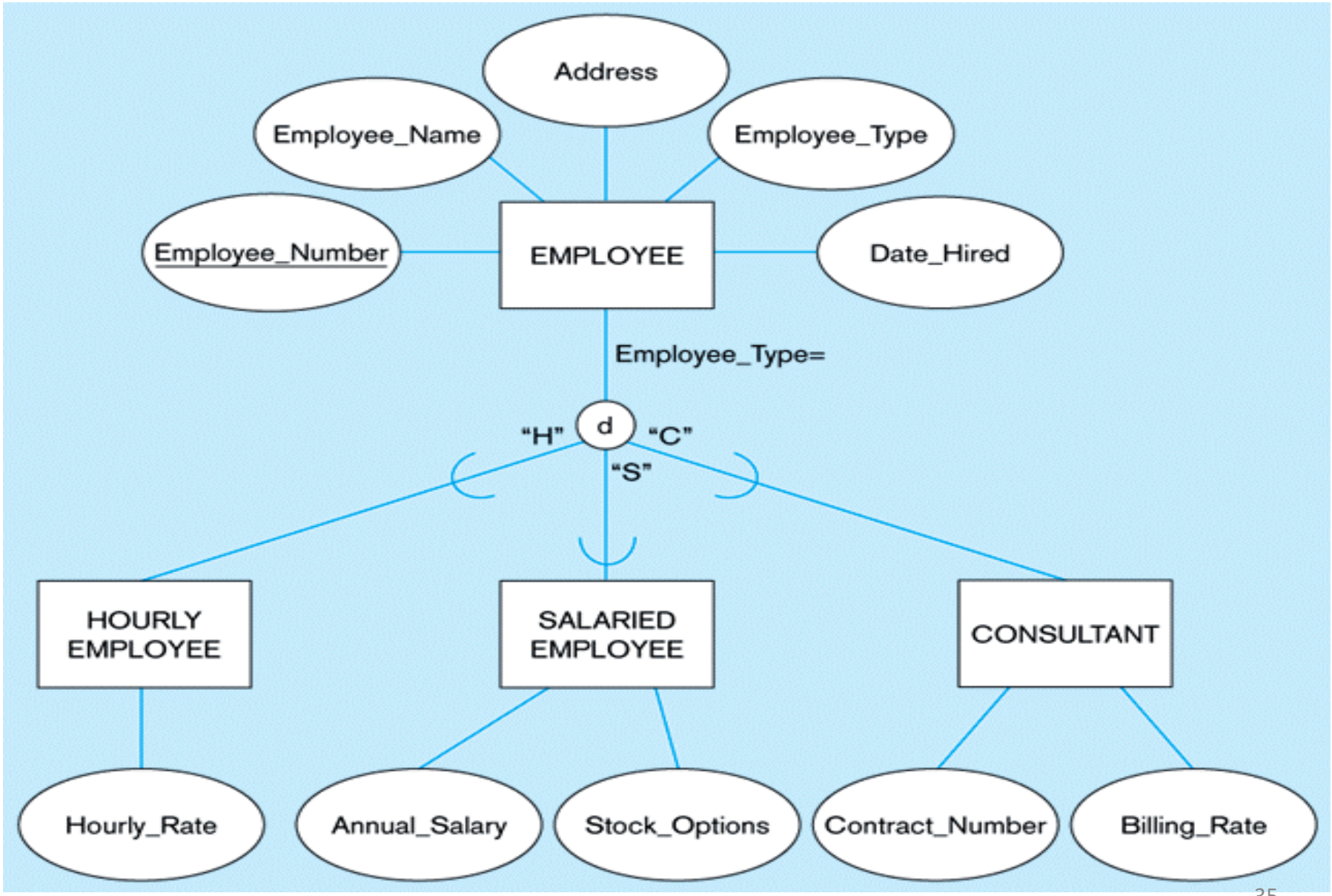


ERD to Relational Table (7)

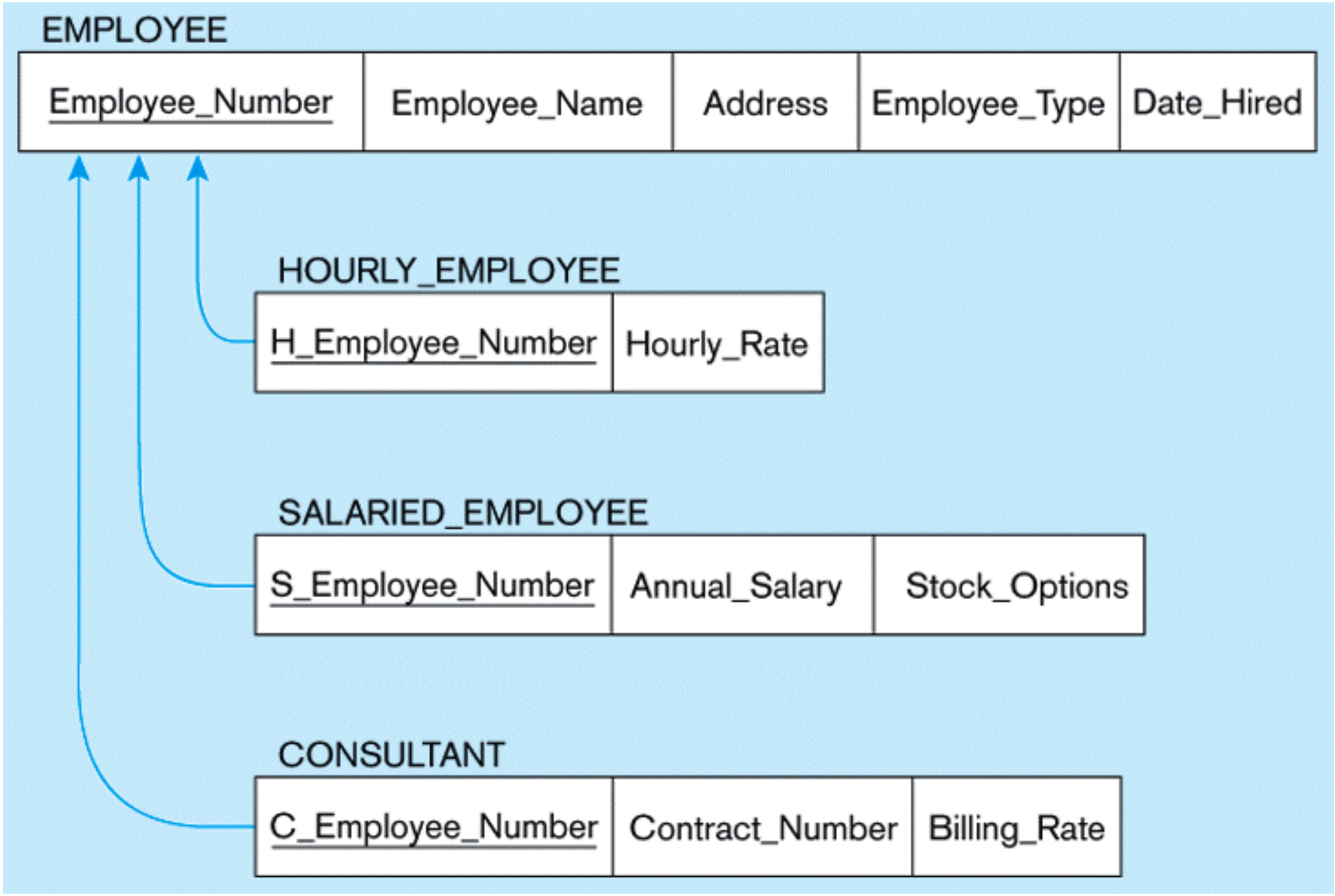
7. Mapping Supertype /Subtype relationships

- A separate relation for supertype and each subtype
- Super type attributes (including identifier and subtype discriminator) go into supertype relation
- Unique subtype attributes go into each subtype relation; primary key of supertype relation also becomes primary key of subtype relation

Supertype/Subtype Relationships



Mapping to Relation tables



Normalization

- Primarily a tool to validate and improve a logical design so that it satisfies certain constraints that avoid unnecessary duplication of data
- Process of decomposing relations with anomalies to produce smaller, well-structured relations
- **Database Normalization** is the process of removing redundant data from the tables into improve storage efficiency, data integrity, and scalability.

Functional Dependency (1)

- **Partial dependency**
  - Let {A, B} is the Primary Key and C is non key attribute.
  - Then if it should be {A, B}→C but B→C or A→C
  - Then C is partially functionally dependent on {A, B}.

Functional Dependency (2)

- **Total dependency**
  - Let {A, B} is the Primary Key and C is non key attribute.
  - Then if {A, B}→C and B→C and A→C does not hold (if B cannot determine C and A cannot determine C).
  - Then C Fully functionally dependent on {A, B}.

Functional Dependency (3)

- **Transitive dependency**

  - If A functionally governs B, AND
  - If B functionally governs C
  - THEN A functionally governs C.

  If A→B and if B→C, Then A→C

Steps of Normalization

- **Unnormalized Form:** Identify all data elements.
- **First Normal Form:** Find the key with which you can find all data.
- **Second Normal Form:** Remove part-key dependencies. Make all data dependent on the whole key.
- **Third Normal Form:** Remove non-key dependencies. Make all data dependent on nothing but the key.
- **Note:** For most practical purposes, databases are considered normalized if they adhere to third normal form.

First Normal Form (1)

- There are no duplicated rows in the table, **Unique identifier.**
- **Each cell is single-valued** (i.e., there are no repeating groups).
- Entries in a column (attribute, field) are of the same kind.
- All repeating groups are moved into a new table
- Foreign keys are used to link the tables
- When a relation contains no repeating groups, is in the first normal form
- Keys must be included to link the relations, tables

First Normal Form (2)

Unnormalised

Module	Dept	Lecturer	Topic
M1	D1	L1	T1, T2
M2	D1	L1	T1, T3
M3	D1	L2	T4
M4	D2	L3	T1, T5
M5	D2	L4	T6

1NF

Module	Dept	Lecturer	Topic
M1	D1	L1	T1
M1	D1	L1	T2
M2	D1	L1	T1
M2	D1	L1	T3
M3	D1	L2	T4
M4	D2	L3	T1
M4	D2	L3	T5
M5	D2	L4	T6

Second Normal Form (1)

- Remove any partial dependencies
- A partial dependency is when the data are only dependent on a part of a key field
- A relation is created for the data that are only dependent on part of the key and another for data that are dependent on both parts

Second Normal Form(2)

1NF

Module	Dept	Lecturer	Topic
M1	D1	L1	T1
M1	D1	L1	T2
M2	D1	L1	T1
M2	D1	L1	T3
M3	D1	L2	T4
M4	D2	L3	T1
M4	D2	L3	T5
M5	D2	L4	T6

2NFa

Module	Dept	Lecturer
M1	D1	L1
M2	D1	L1
M3	D1	L2
M4	D2	L3
M5	D2	L4

2NFb

Module	Topic
M1	T1
M1	T2
M2	T1
M2	T3
M3	T4
M4	T1
M4	T5
M1	T6

Third Normal Form(1)

- Remove any transitive dependencies
- A transitive dependency is when a relation contains data that are not part of the entity
- The problem with transitive dependencies is updating the data
- A single data item may be present on many records

Third Normal Form (2)

2NFa

Module	Dept	Lecturer
M1	D1	L1
M2	D1	L1
M3	D1	L2
M4	D2	L3
M5	D2	L4

3NFa

Lecturer	Dept
L1	D1
L2	D1
L3	D2
L4	D2

3NFb

Module	Lecturer
M1	L1
M2	L1
M3	L2
M4	L3
M5	L4

Other Forms of Normalization

- **Boyce-Codd Normal Form (BCNF)**
  - A table is in BCNF if it is in 3NF and if every determinant is a candidate key.
- **Fourth Normal form (4NF)**
  - A table is in 4NF if it is in BCNF and if it has no multi-valued dependencies
- **Fifth Normal Form (5NF)**
  - Projection-Join Normal Form" (PJNF),
  - if it is in 4NF and if every join dependency in the table is a consequence of the candidate keys of the table.
- **Domain-Key Normal Form (DKNF)**
  - A table is in DKNF if every constraint on the table is a logical consequence of the definition of keys and domains.
  - A model free from all modification anomalies.

Outcomes of Normalization

- Contain all the data necessary for the purposes that the database is to serve,
- Have as little redundancy as possible,
- Accommodate multiple values for types of data that require them
- Permit efficient updates of the data in the database, and
- Avoid the danger of losing data unknowingly.

Pitfalls of Normalization

- Requires data to see the problems
- May reduce performance of the system
- Is time consuming
- Difficult to design and apply and
- Prone to human error

Thank You !!!