Database Management Systems

Subject Teacher: Zartasha Baloch

The Relational Database Model

Lecture # 5, 6, & 7

Disclaimer: The material used in this presentation to deliver the lecture i.e., definitions/text and pictures/graphs etc. does not solely belong to the author/presenter. The presenter has gathered this lecture material from various sources on web/textbooks. Following sources are especially acknowledged:

1. Connolly, Thomas M., and Carolyn E. Begg. *Database systems: a practical approach to design, implementation, and management*. Pearson Education, 2005.
2. 2. Hoffer, Jeffrey A., Venkataraman Ramesh, and Heikki Topi. *Modern database management*. Upper Saddle River, NJ: Prentice Hall,, 2011.

Data Model

- Integrated collection of concepts for describing data, relationships between data, and constraints on the data.
- Has three components:
  - a structural part;
  - a manipulative part;
  - a set of integrity rules.

Relational DBMS

- Dr. Edgar F. Codd at IBM invented the relational database in 1970, called Father of RDBMS.
- The main elements of RDBMS are based on Codd's 13 rules for a relational system.
- A DBMS is said to be relational if it follows Codd's rules.

Relational Model Terminologies

- **Tables (Relation)** − In relational data model, relations are saved in the format of Tables. This format stores the relation among entities. A table has rows and columns, where rows represents records and columns represent the attributes.
- **Tuple** − A single row of a table, which contains a single record for that relation is called a tuple.
- **Attribute:** The named column of a relation.
- **Attribute domain** − Every attribute has some pre-defined value scope, known as attribute domain. It is set of allowable values for one or more attributes.
- **Relational Database** - collection of normalized relations with distinct relation names.

Relational Model Terminologies

- **Relation instance** − A finite set of tuples in the relational database system represents relation instance. Relation instances do not have duplicate tuples.
- **Relation schema** − A relation schema describes the relation name (table name), attributes, and their names.
- **Relation key** − Each row has one or more attributes, known as relation key, which can identify the row in the relation (table) uniquely.

Alternative Terminology

- Relation, attribute, tuple
- Table, column, record
- File, field, row

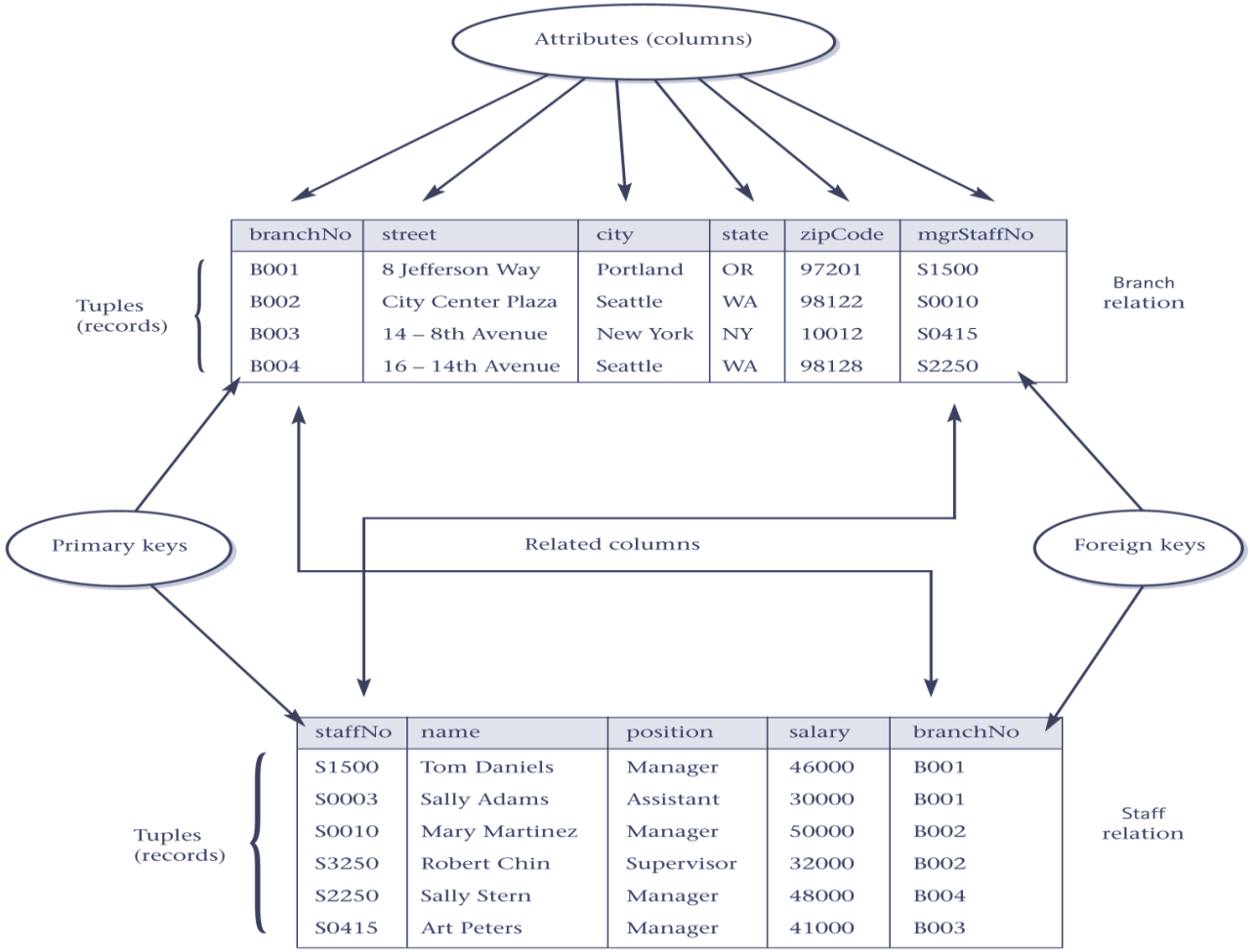
Properties of Relations

- Each record is distinct; there are no duplicate records.
- Order of columns has no significance.
- Order of records has no significance, theoretically.

Properties of Relations

- Table name is distinct from all other table names in the database.
- Each cell of table contains exactly one atomic (single) value.
- Each column has a distinct name.
- Values of a column are all from the same domain.

Instances of Branch and Staff (part) Relations



branchNo	street	city	state	zipCode	mgrStaffNo
B001	8 Jefferson Way	Portland	OR	97201	S1500
B002	City Center Plaza	Seattle	WA	98122	S0010
B003	14 – 8th Avenue	New York	NY	10012	S0415
B004	16 – 14th Avenue	Seattle	WA	98128	S2250

staffNo	name	position	salary	branchNo
S1500	Tom Daniels	Manager	46000	B001
S0003	Sally Adams	Assistant	30000	B001
S0010	Mary Martinez	Manager	50000	B002
S3250	Robert Chin	Supervisor	32000	B002
S2250	Sally Stern	Manager	48000	B004
S0415	Art Peters	Manager	41000	B003

Example Attribute Domains

Attribute	Domain name	Meaning	Domain definition
branchNo	Branch_Numbers	Set of all possible branch numbers.	Alphanumeric: size 4, range B001–B999
street	Street_Names	Set of all possible street names.	Alphanumeric: size 60
staffNo	Staff_Numbers	Set of all possible staff numbers.	Alphanumeric: size 5, range S0001–S9999
position	Staff_Positions	Set of all possible staff positions.	One of Director, Manager, Supervisor, Assistant, Buyer
salary	Staff_Salaries	Possible values of staff salaries.	Monetary: 8 digits, range $10,000.00–$100,000.00

Relational Keys

- Super key
  - A column, or a set of columns, that uniquely identifies a record within a table.
  - It is further reducible without loosing uniqueness.
- Candidate Key
  - Super key (K) such that no proper subset is a super key within the table.
  - In each record, values of K uniquely identify that record (*uniqueness*).
  - No proper subset of K has the uniqueness property (*irreducibility*).

Relational Keys

- Primary Key
  - Candidate key selected to identify records uniquely within table.
- Alternate Keys
  - Candidate keys that are not selected to be primary key.
- Foreign Key
  - Column, or set of columns, within one table that matches candidate key of some (possibly same) table.

Relational Integrity

- **Null**
  - Represents value for a column that is currently unknown or not applicable for record.
  - Deals with incomplete or exceptional data.
  - Represents the absence of a value and is not the same as zero or spaces, which are values.

Relational Integrity constraints

- Every relation has some conditions that must hold for it to be a valid relation. These conditions are called Relational Integrity Constraints.
- There are three main integrity constraints –
  - Key constraints (Entity Integrity)
  - Referential integrity constraints
  - Domain constraints (Business rules)

Relational Integrity

- **Entity Integrity**
  - In a base table, no column of a primary key can be null.

- **Business Rules**
  - Rules that define or constrain some aspect of the organization.

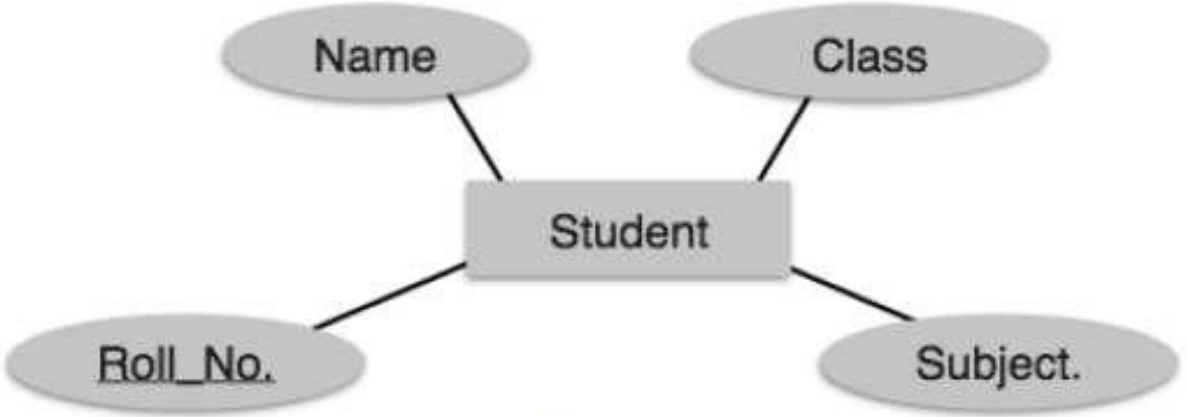
Relational Integrity

- **Referential Integrity**

  If FK exists in a table, either FK value must match a candidate key value of some record in its home table or FK value must be wholly 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

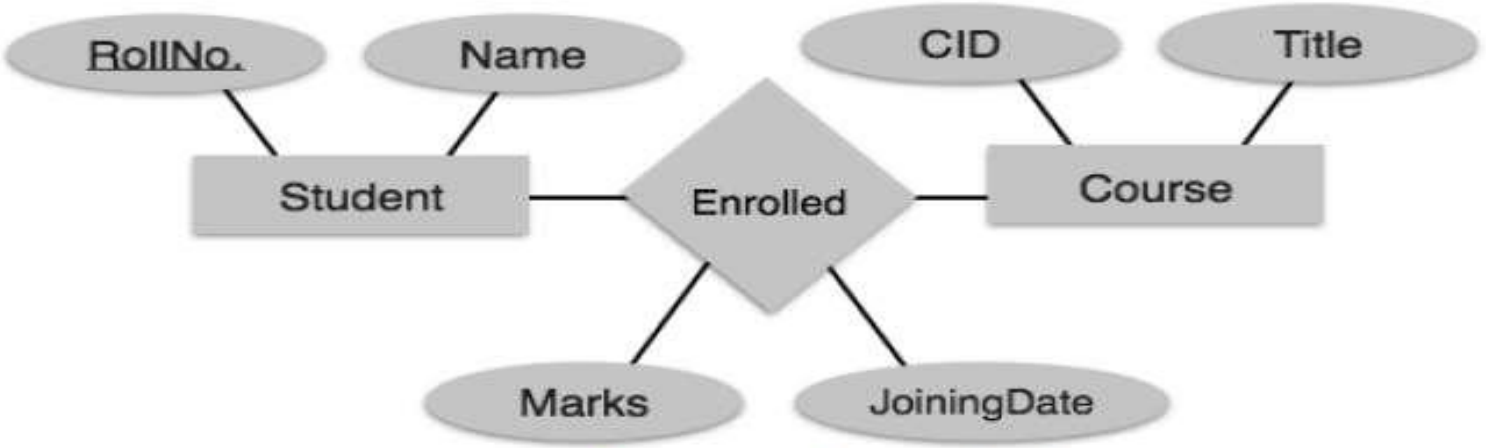
Transforming ER Model to relational model



[Image: Mapping Entity]

- Create table for each entity
- Entity's attributes should become fields of tables with their respective data types.
- Declare primary key

Transforming ER Model to relational model



[Image: Mapping relationship]

- Create table for a relationship
- Add the primary keys of all participating Entities as fields of table with their respective data types.
- If relationship has any attribute, add each attribute as field of table.
- Declare a primary key composing all the primary keys of participating entities.
- Declare all foreign key constraints.

Student

RollNo	SName	Address

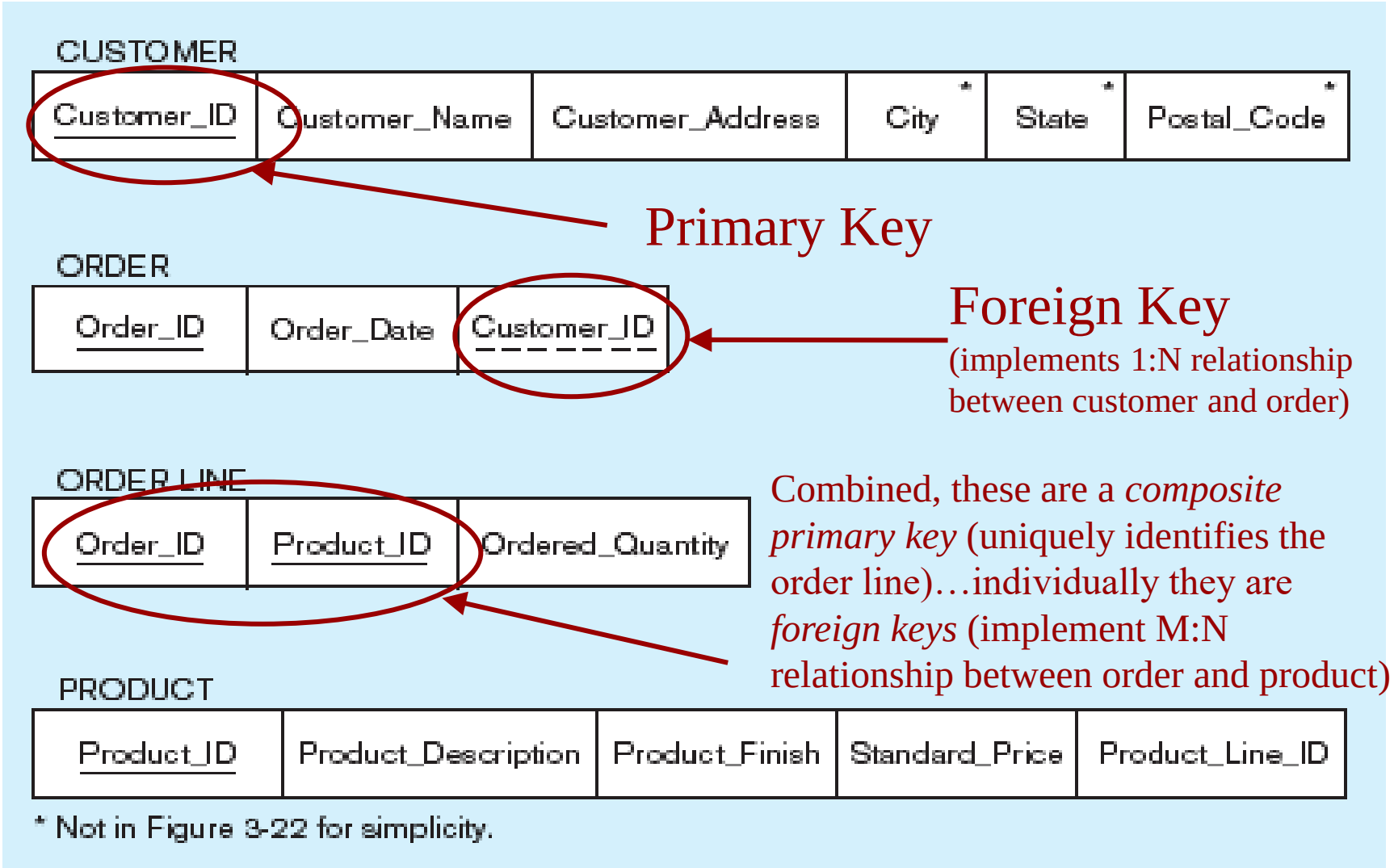
Enrollment

RollNo	CId	Marks	Joining Date

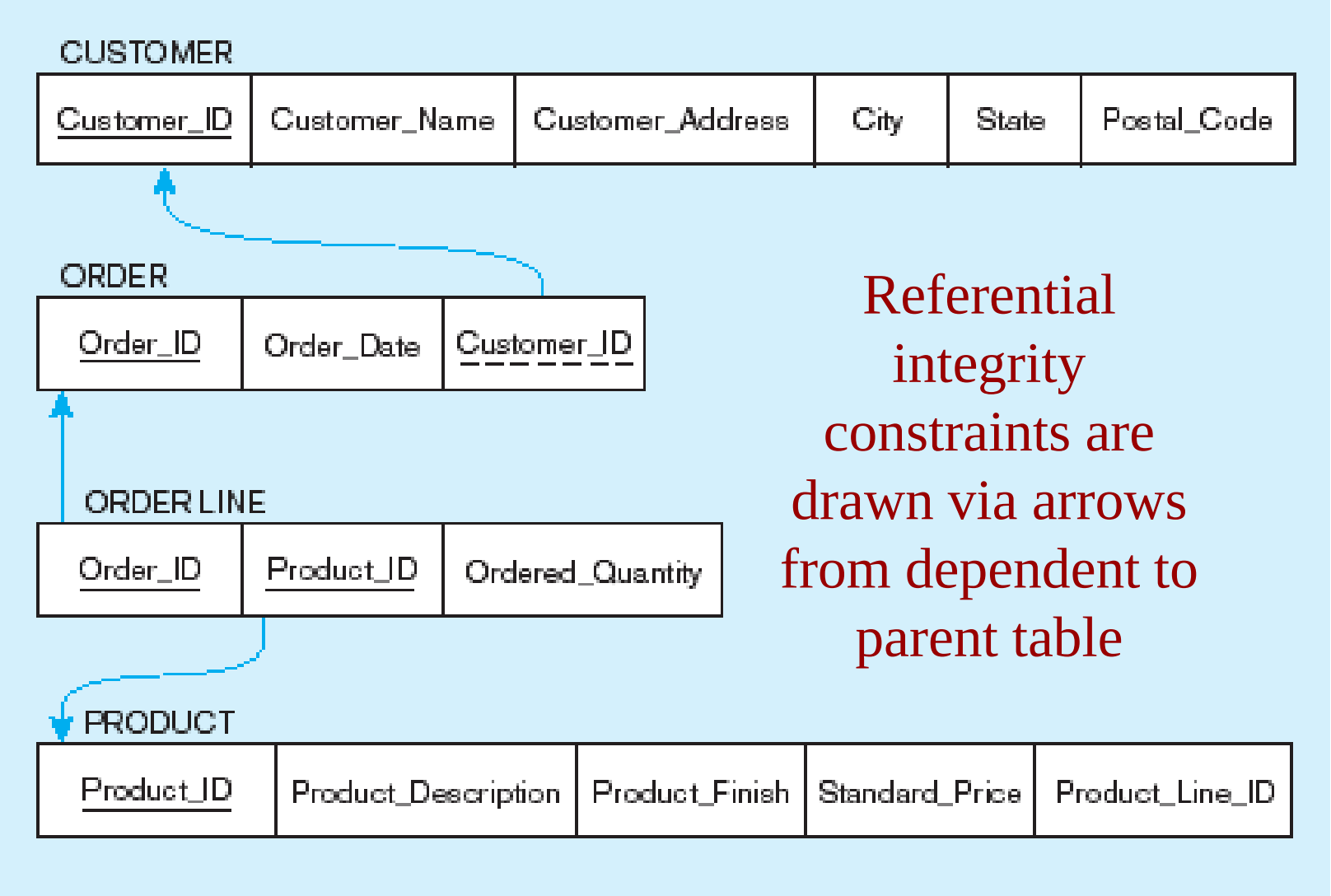
Course

CId	Title

Schema for four relations (Pine Valley Furniture Company)



Referential integrity constraints (Pine Valley Furniture)



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

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

Customer_ID	Customer_Name	Customer_Address	Postal_Code

Mapping a composite attribute

(a) CUSTOMER entity type with composite attribute

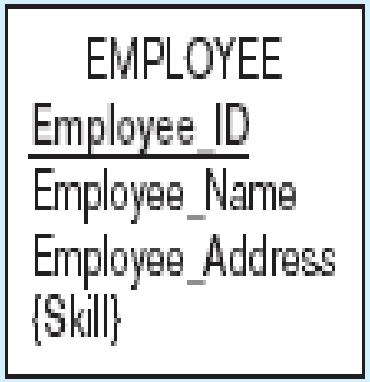
(b) CUSTOMER relation with address detail

CUSTOMER

Customer_ID	Customer_Name	Street	City	State	Postal_Code

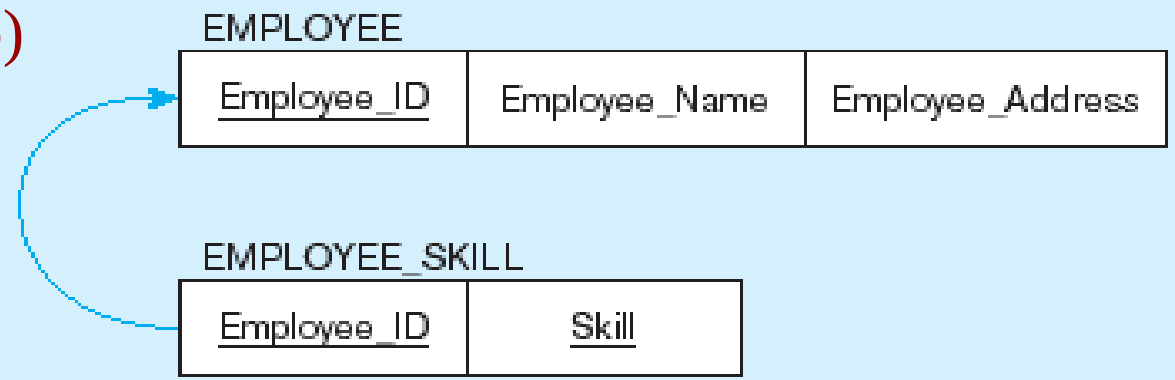
Mapping an entity with a multivalued attribute

(a)



Multivalued attribute becomes a separate relation with foreign key

(b)



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

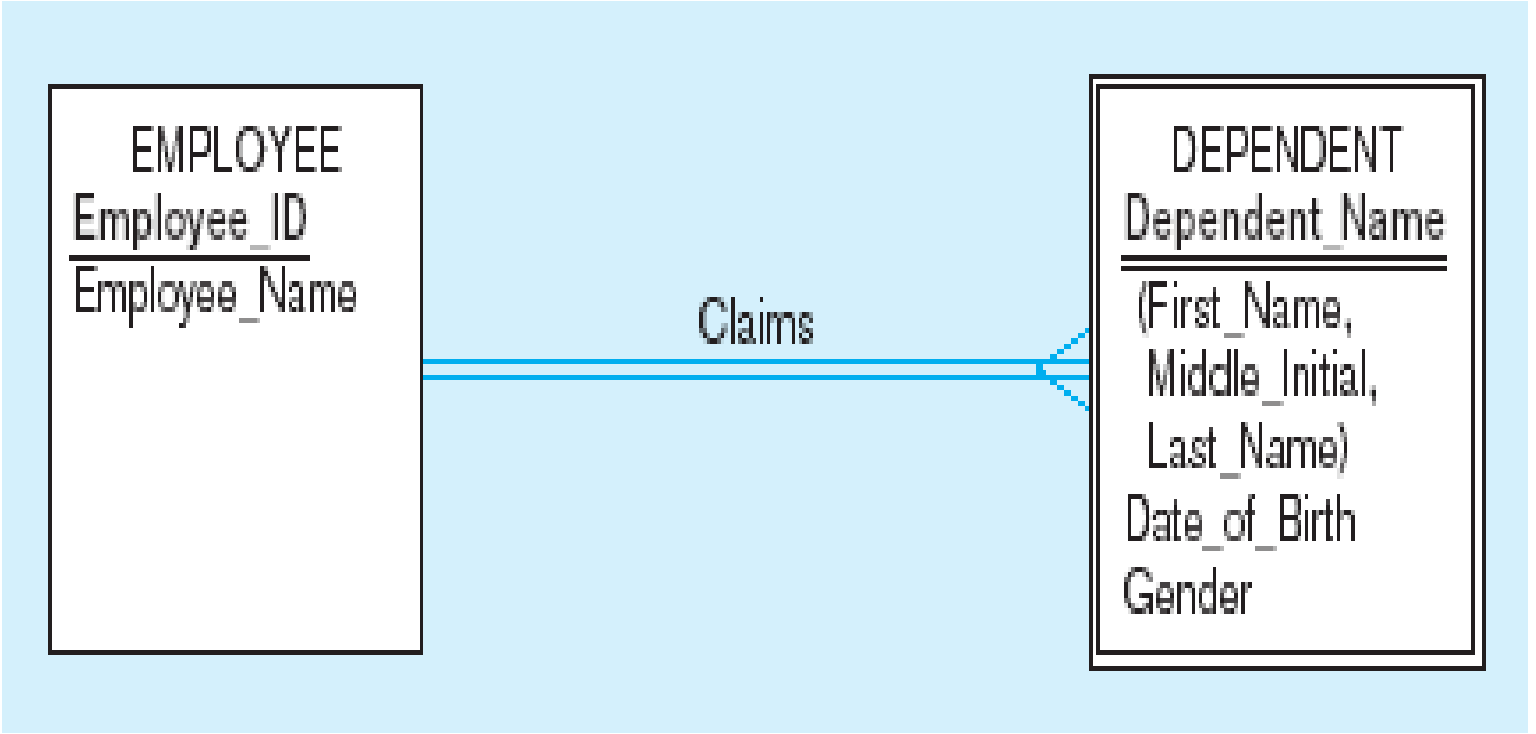
Transforming ER Diagrams into Relations (cont.)

Mapping Weak Entities

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

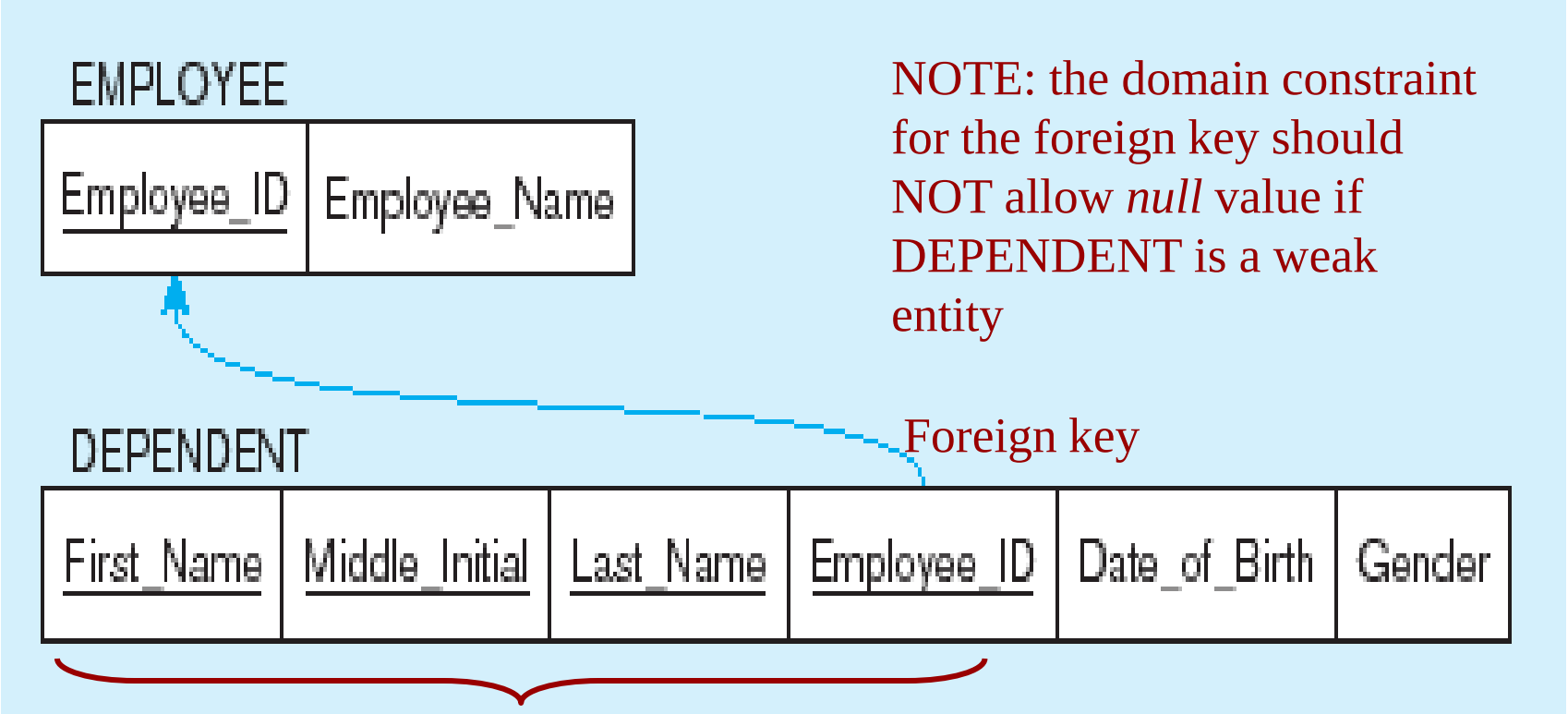
Example of mapping a weak entity

a) Weak entity DEPENDENT



Example of mapping a weak entity (cont.)

b) Relations resulting from weak entity



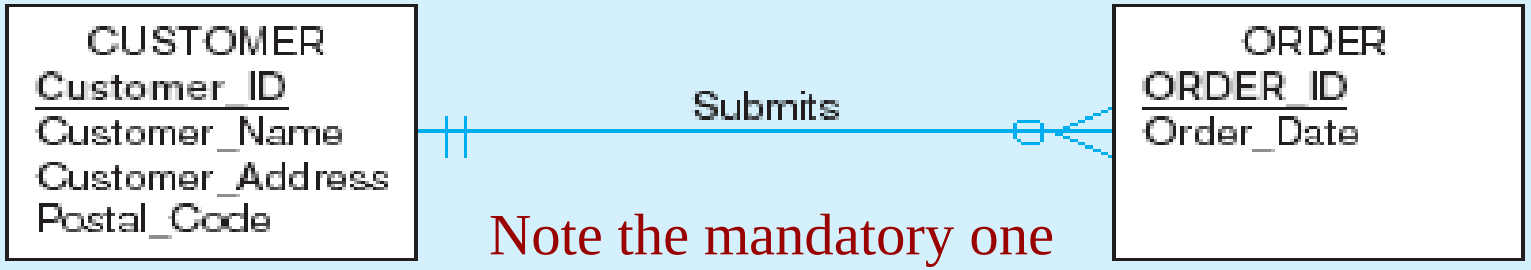
Transforming ER Diagrams into Relations (cont.)

Mapping Binary Relationships

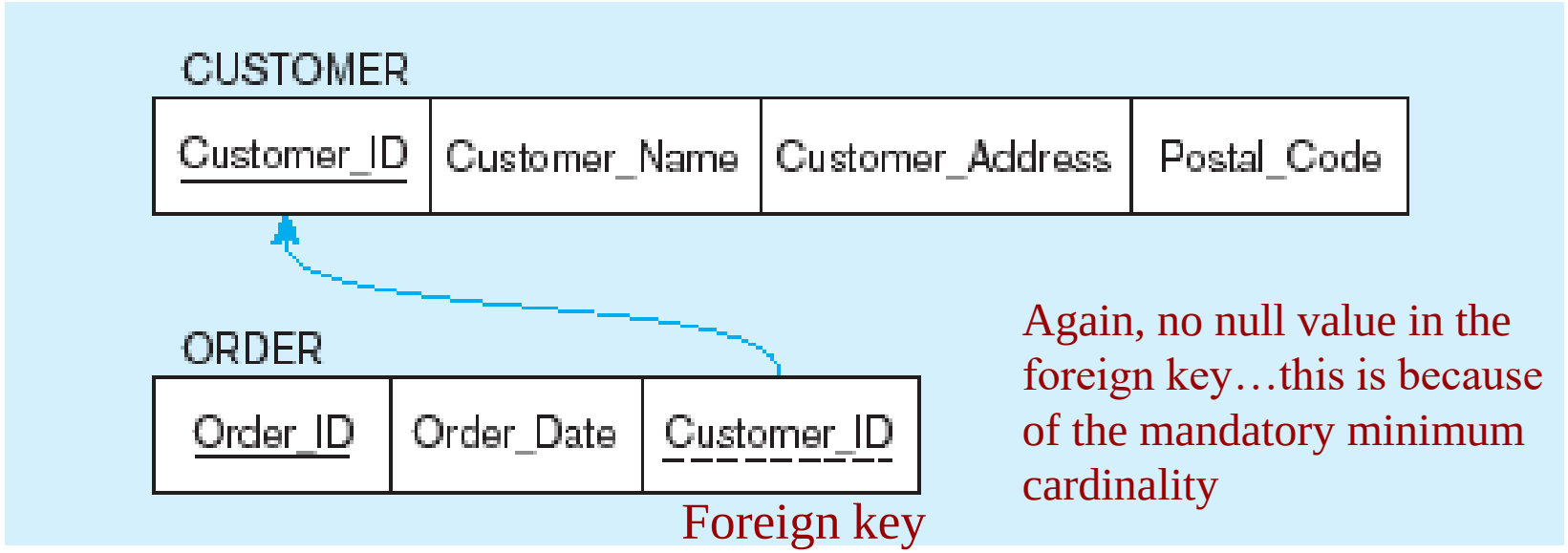
- One-to-Many–Primary key on the one side becomes a foreign key on the many side
- Many-to-Many–Create a ***new relation*** with the primary keys of the two entities as its primary key
- One-to-One–Primary key on the mandatory side becomes a foreign key on the optional side

Example of mapping a 1:M relationship

a) Relationship between customers and orders

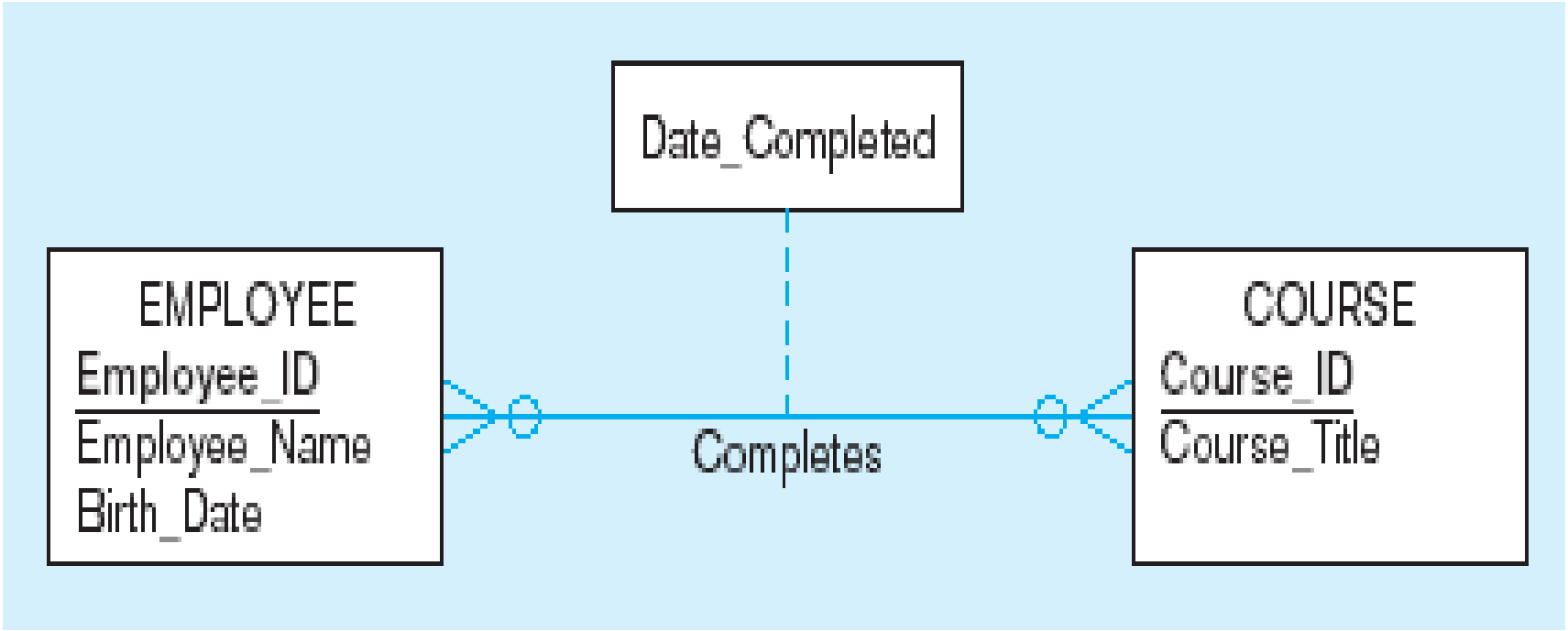


b) Mapping the relationship



Example of mapping an M:N relationship

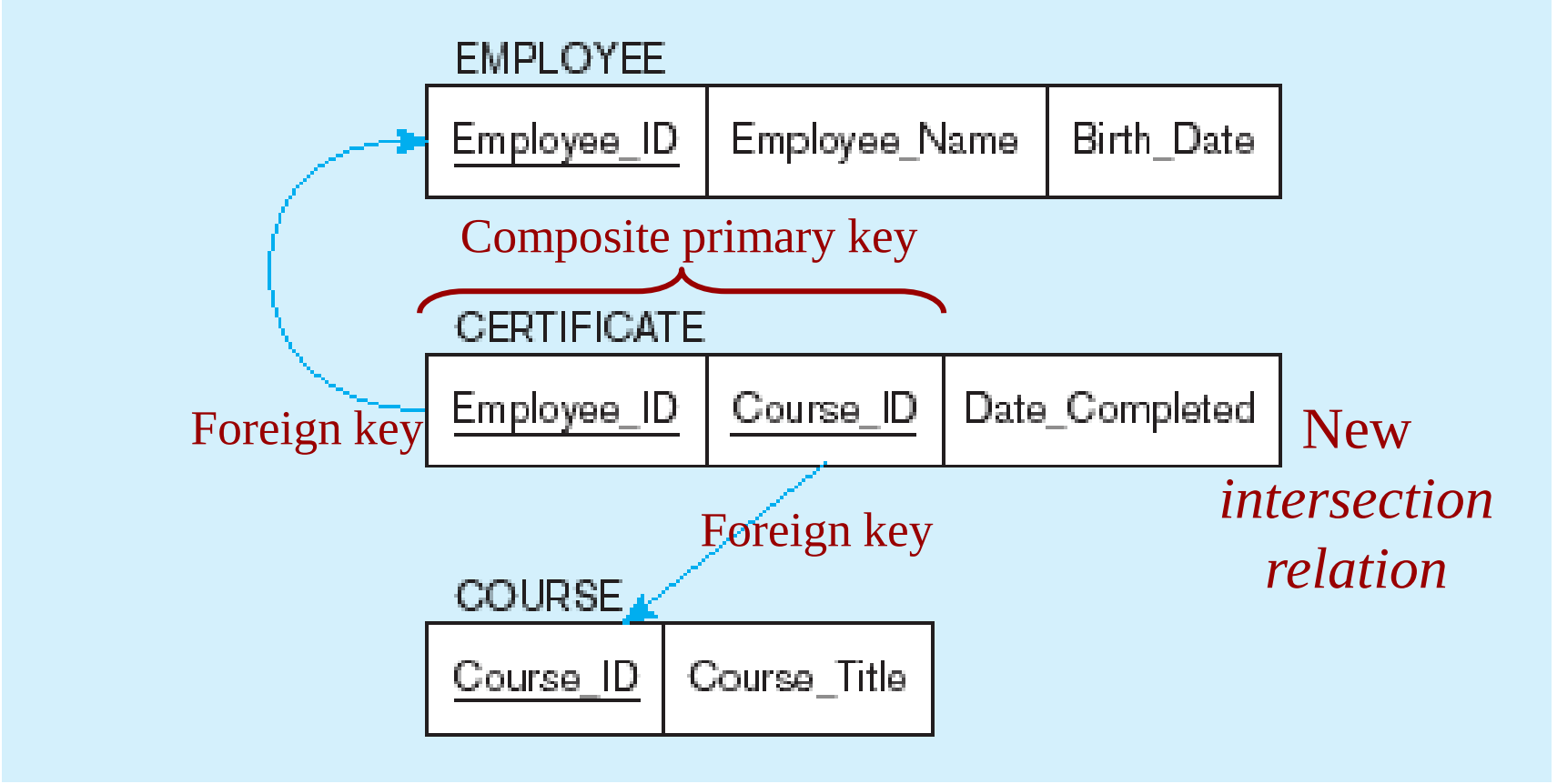
a) Completes relationship (M:N)



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

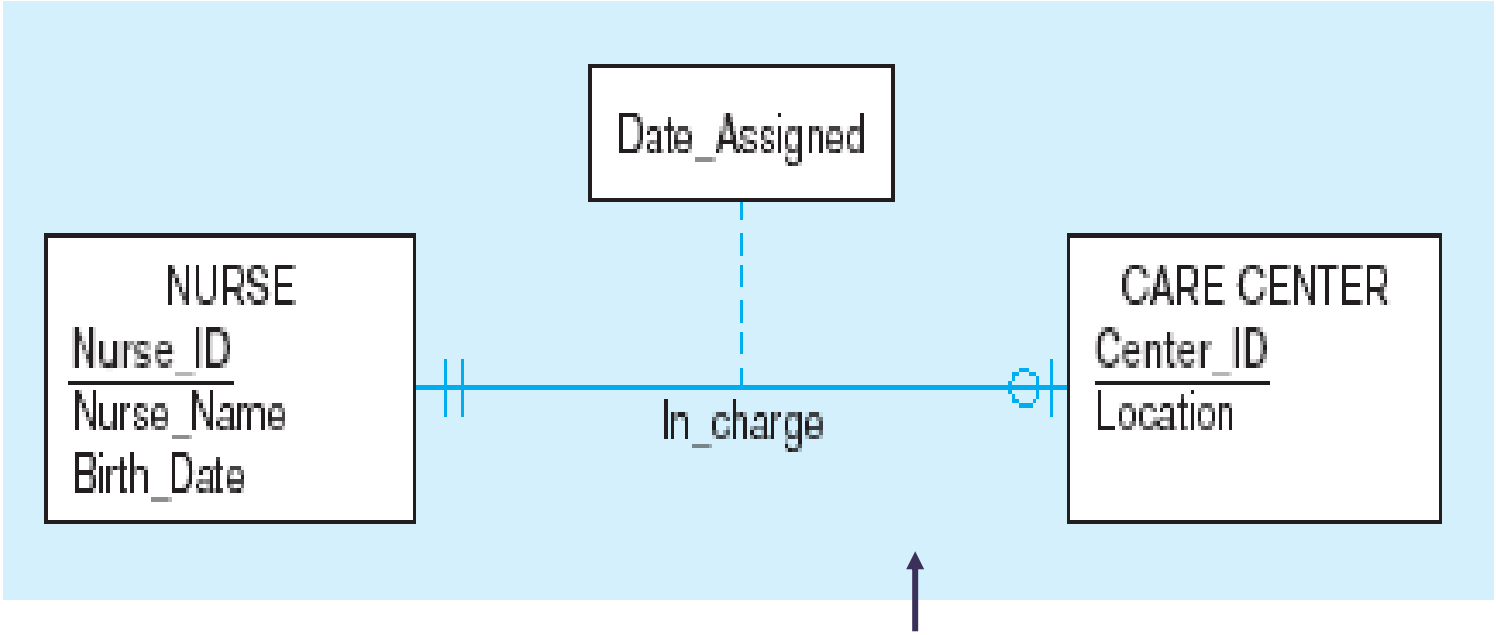
Example of mapping an M:N relationship (cont.)

b) Three resulting relations



Example of mapping a binary 1:1 relationship

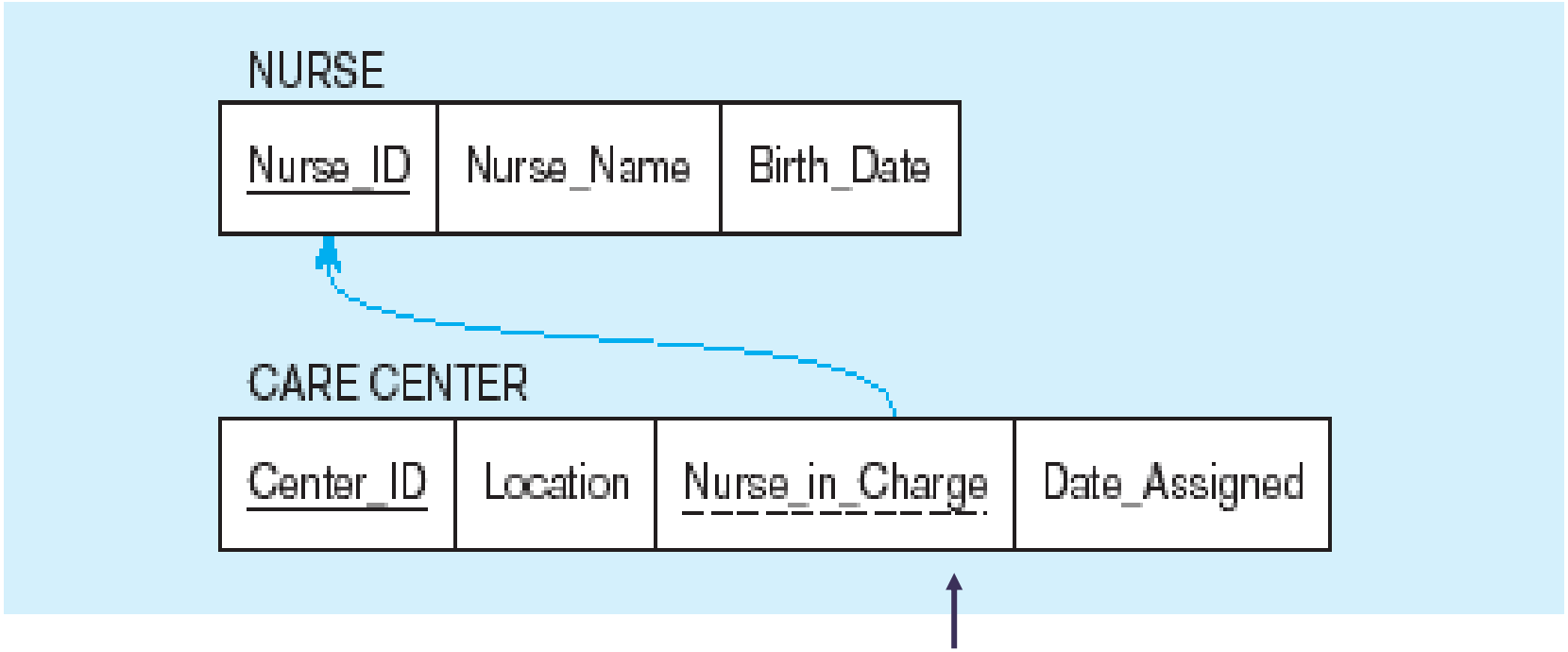
a) In_charge relationship (1:1)



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

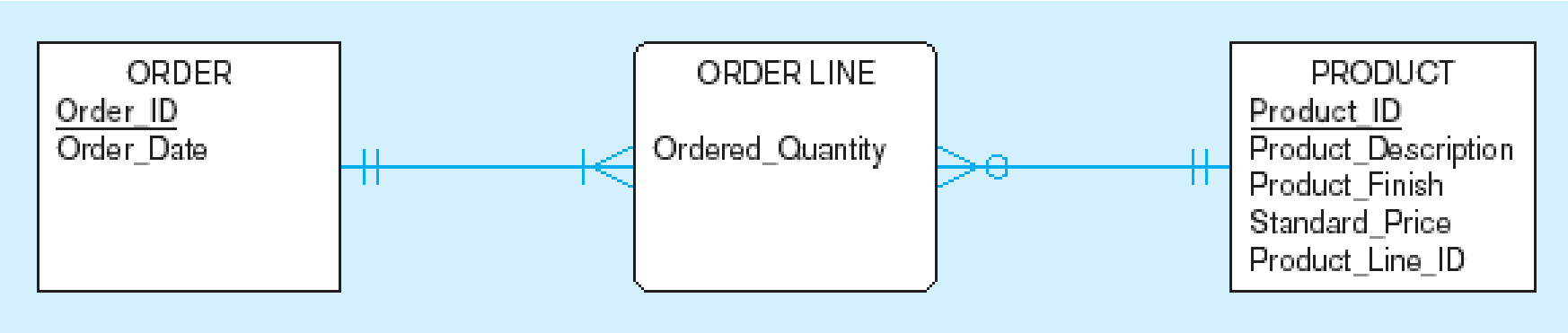
Transforming EER Diagrams into Relations (cont.)

Mapping Associative Entities

- Identifier Not Assigned
  - Default primary key for the association relation is composed of the primary keys of the two entities (as in M:N relationship)
- Identifier Assigned
  - It is natural and familiar to end-users
  - Default identifier may not be unique

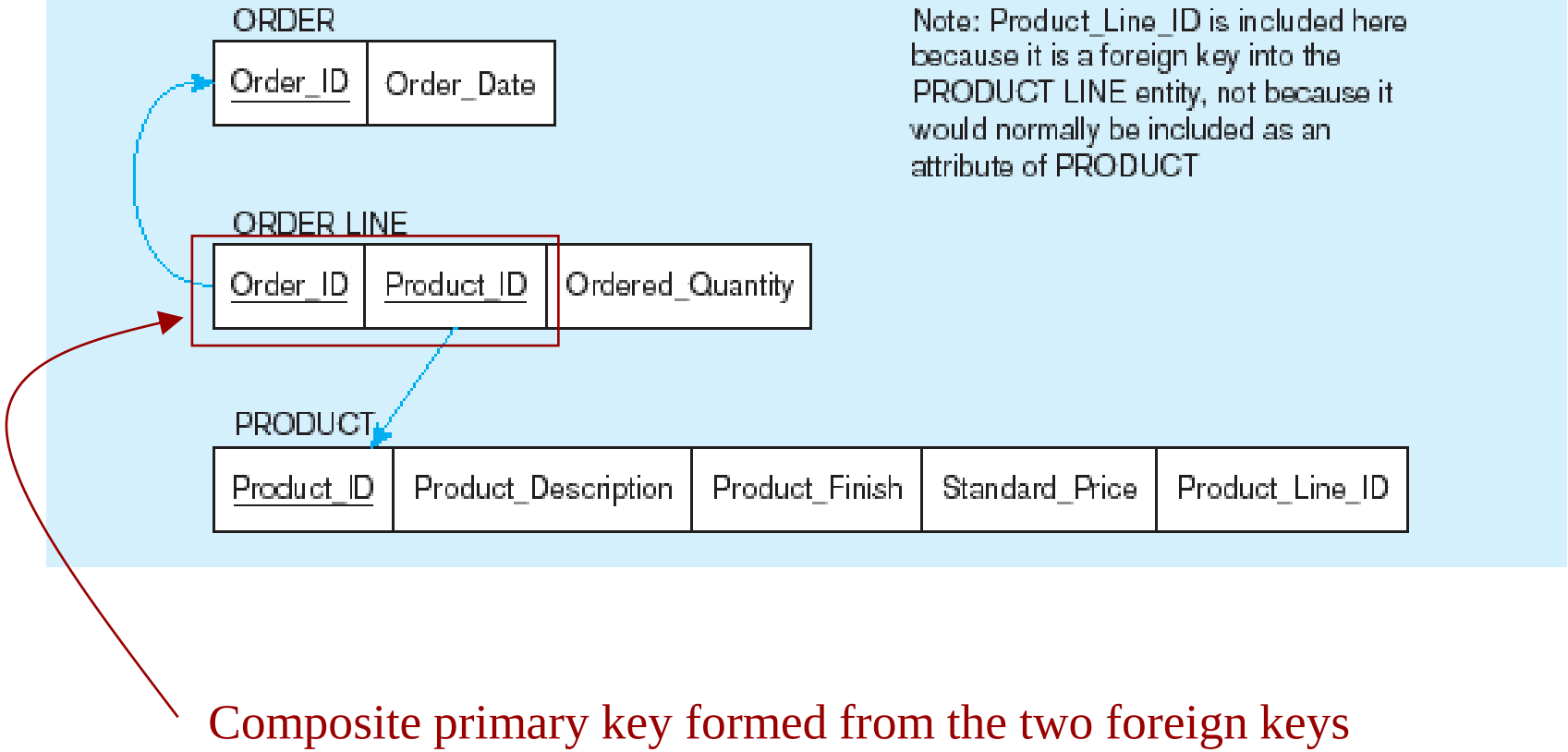
Example of mapping an associative entity

a) An associative entity



Example of mapping an associative entity (cont.)

b) Three resulting relations

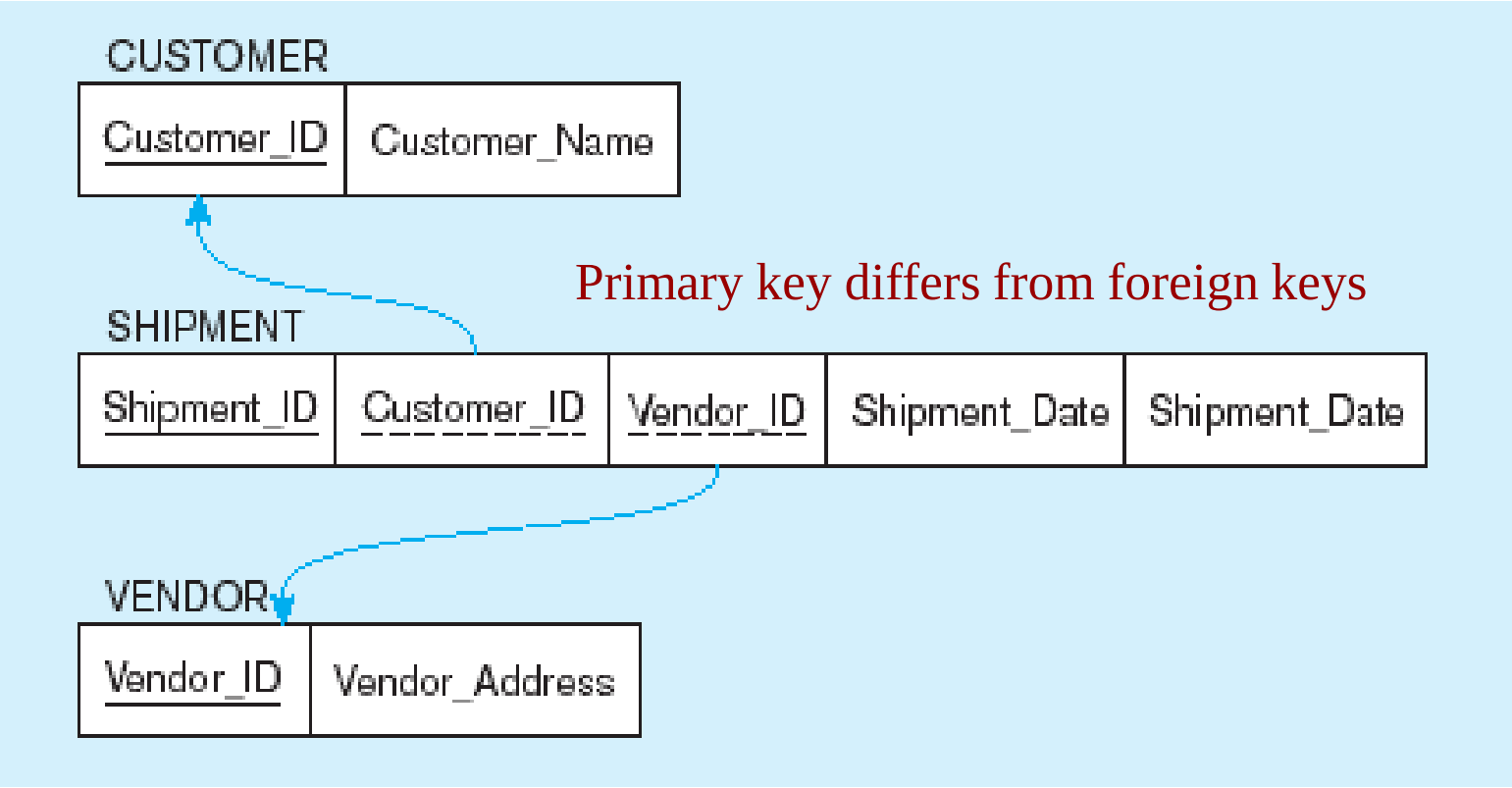


a) SHIPMENT associative entity

Example of mapping an associative entity with an identifier

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

b) Three resulting relations



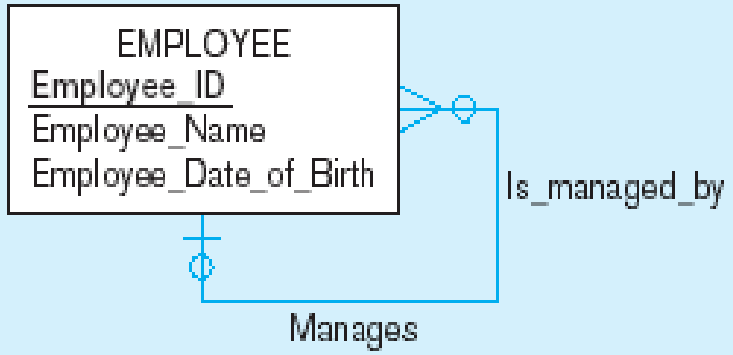
Transforming EER Diagrams into Relations (cont.)

Mapping Unary Relationships

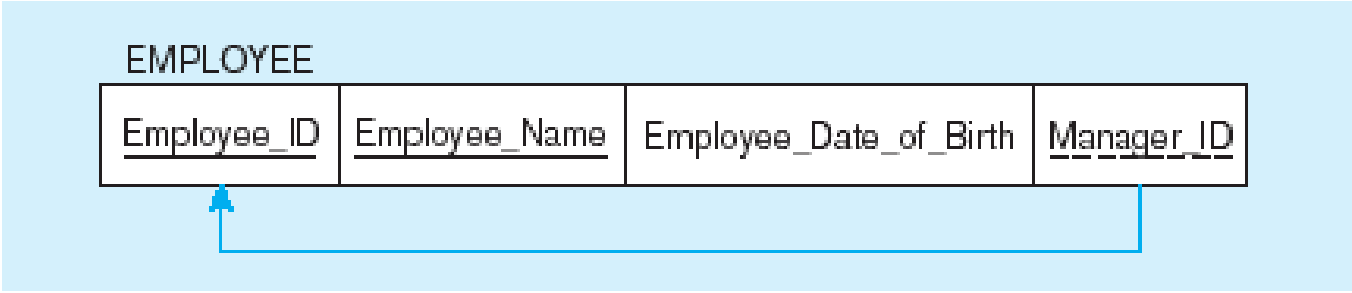
- One-to-Many–Recursive foreign key in the same relation
- Many-to-Many–Two relations:
  - One for the entity type
  - One for an associative relation in which the primary key has two attributes, both taken from the primary key of the entity

Mapping a unary 1:N relationship

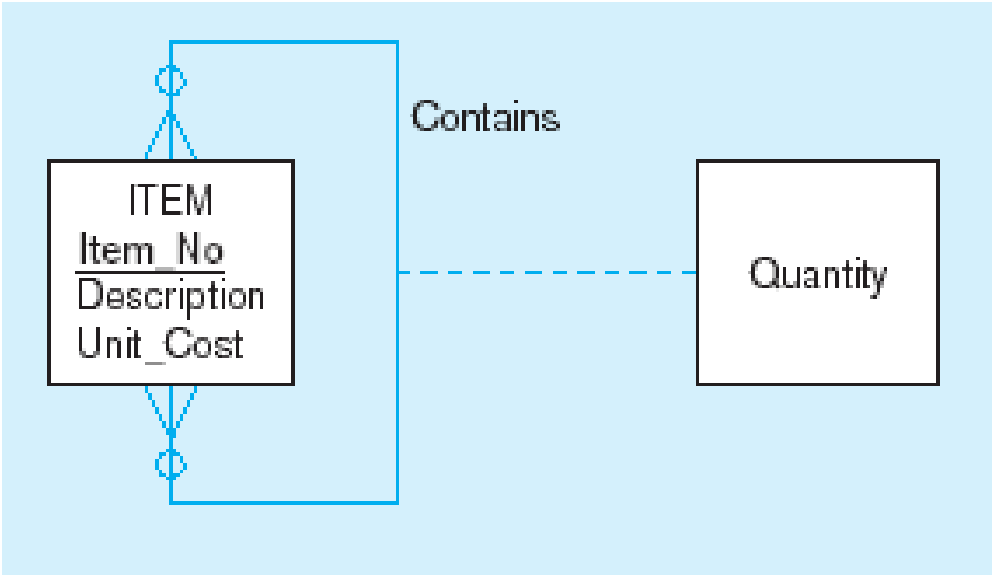
(a) EMPLOYEE entity with unary relationship



(b) EMPLOYEE relation with recursive foreign key

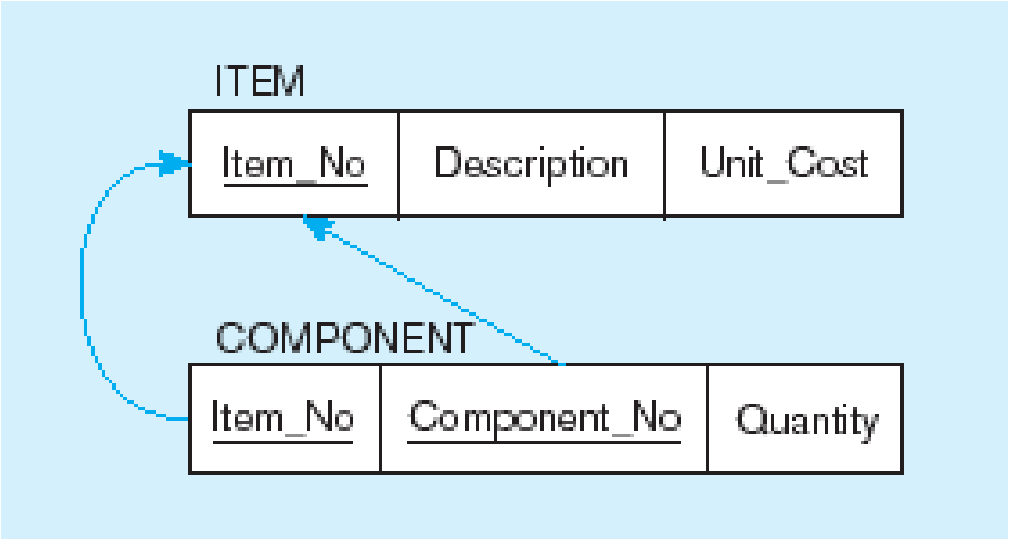


Mapping a unary M:N relationship



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

(b) ITEM and COMPONENT relations



Transforming EER Diagrams into Relations (cont.)

Mapping Ternary (and n-ary) Relationships

- 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

a) PATIENT TREATMENT Ternary relationship with associative entity

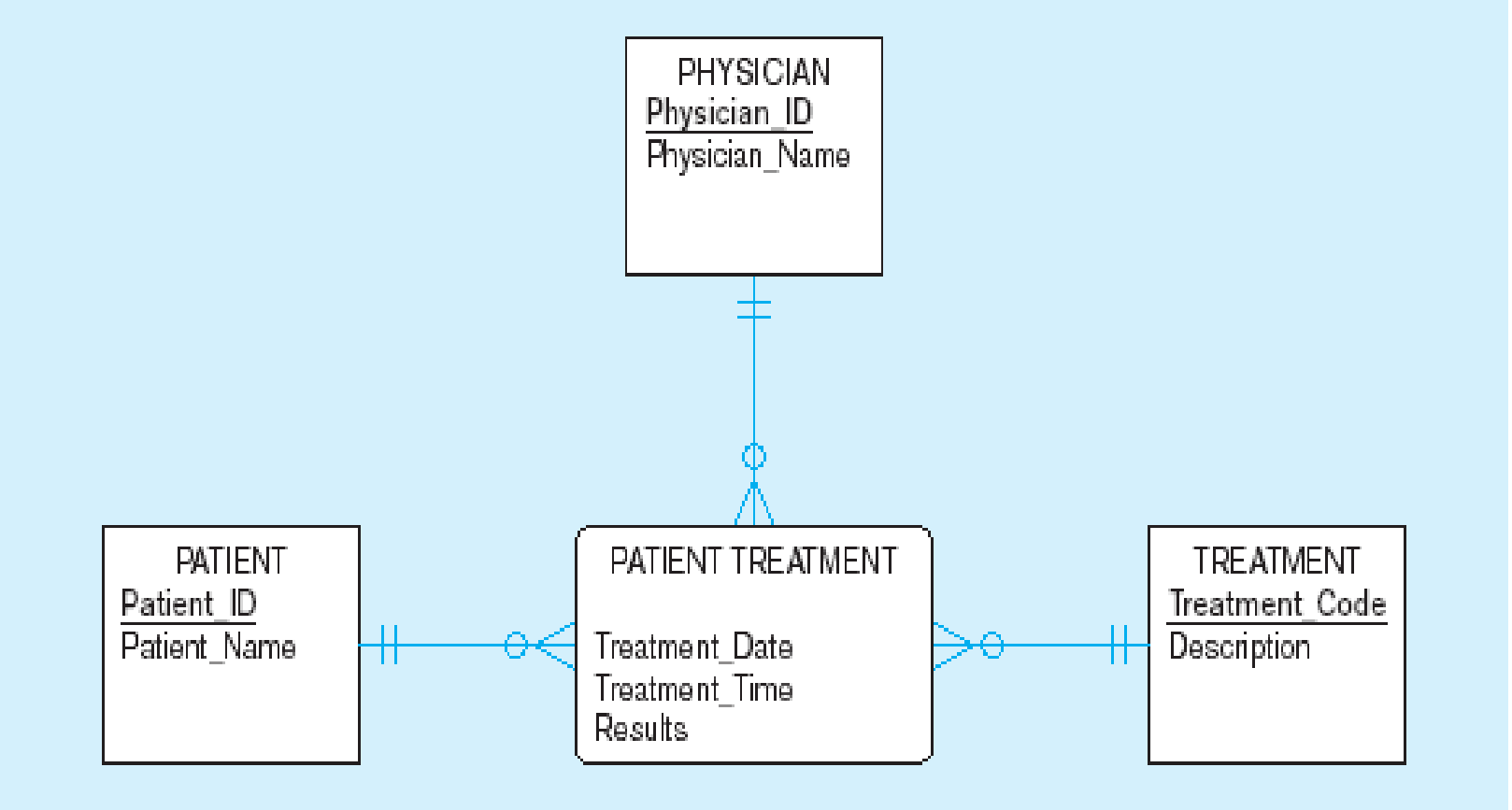
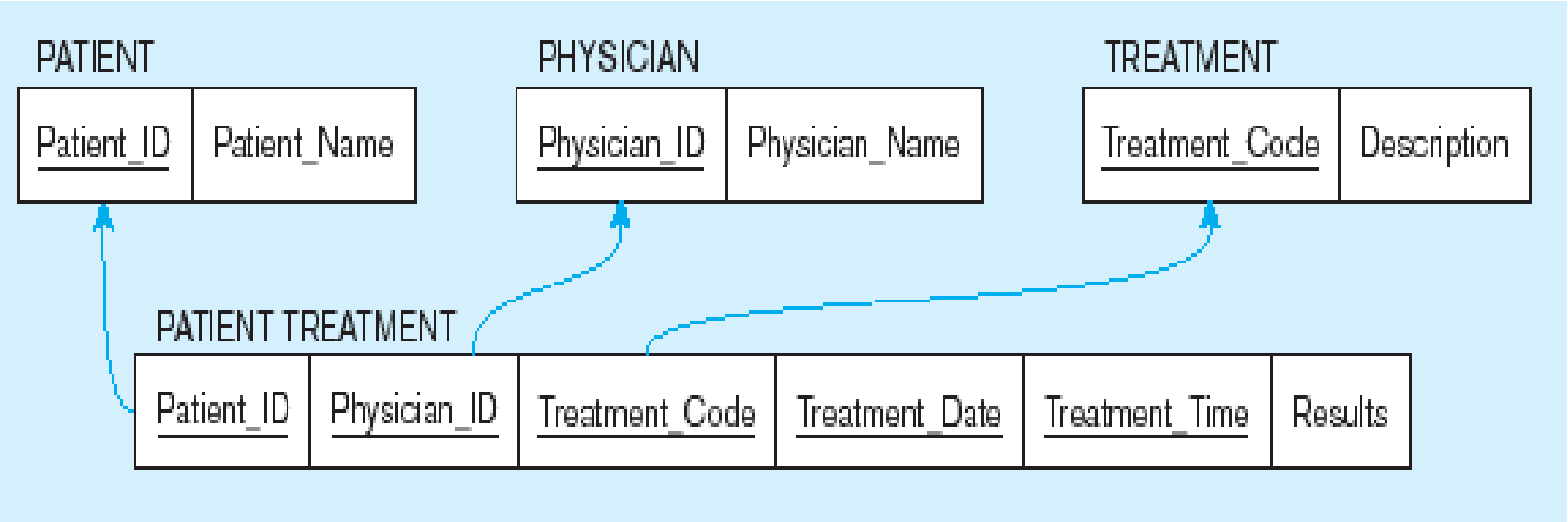


Figure 5-19 Mapping a ternary relationship (cont.)

b) Mapping the ternary relationship PATIENT TREATMENT



Remember that the primary key MUST be unique

Therefore, treatment date and time are included in the composite primary key

But this makes a very awkward key…

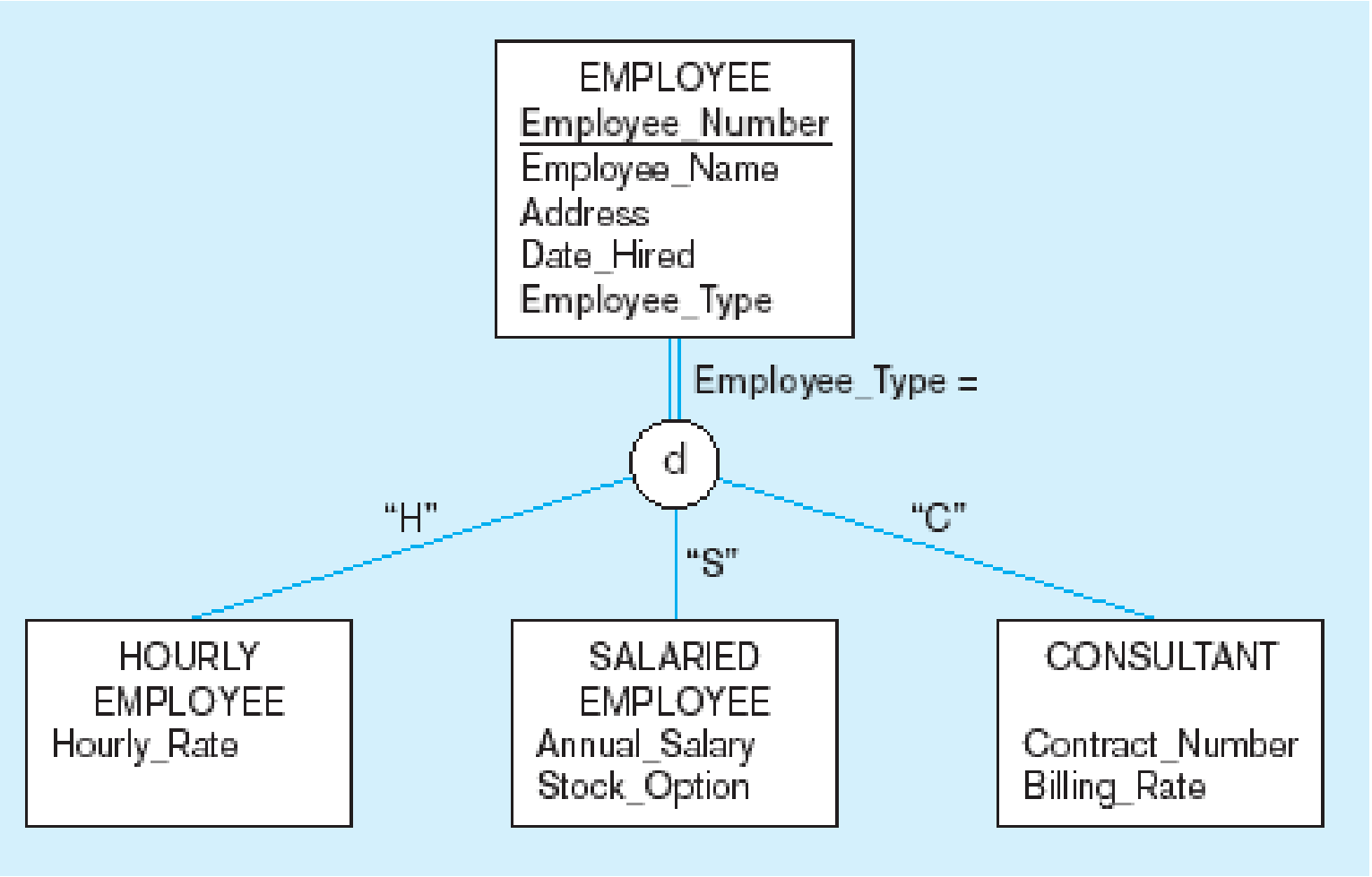
It would be better to create a surrogate key like Treatment#

Transforming EER Diagrams into Relations (cont.)

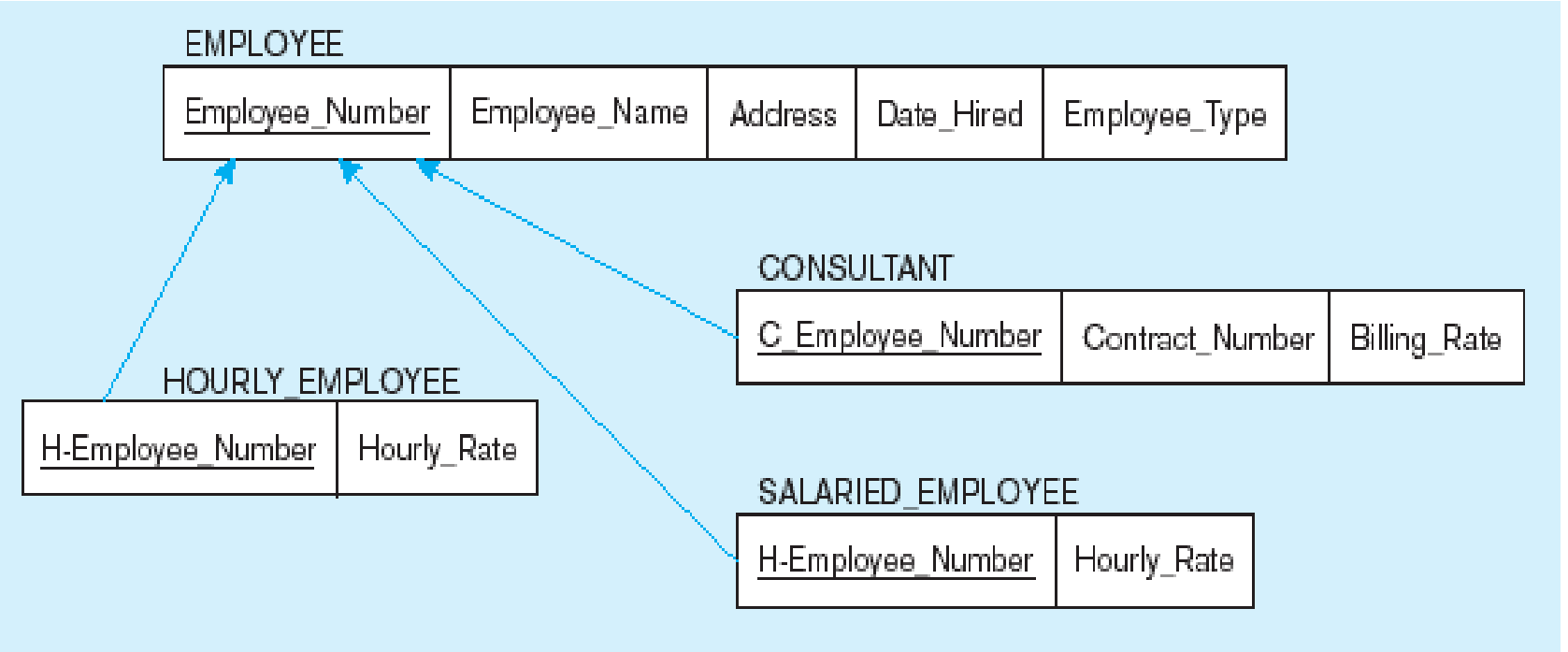
Mapping Supertype/Subtype Relationships

- 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

Supertype/subtype relationships



Mapping Supertype/subtype relationships to relations



These are implemented as one-to-one relationships

Relational Languages

- Two main languages:
  - SQL (Structured Query Language), standardized by ISO.
  - QBE (Query-by-Example), alternative graphical “point-and-click” way of querying database.