
Family isn't an important thing. It's everything.
Michael J. Fox
www.YourPositiveOasis.com

MOTIVATION

LOGICAL
DATABASE DESIGN AND RELATIONAL MODEL

Module 4

Information Management

College of Computer Studies
Calapan City Campus

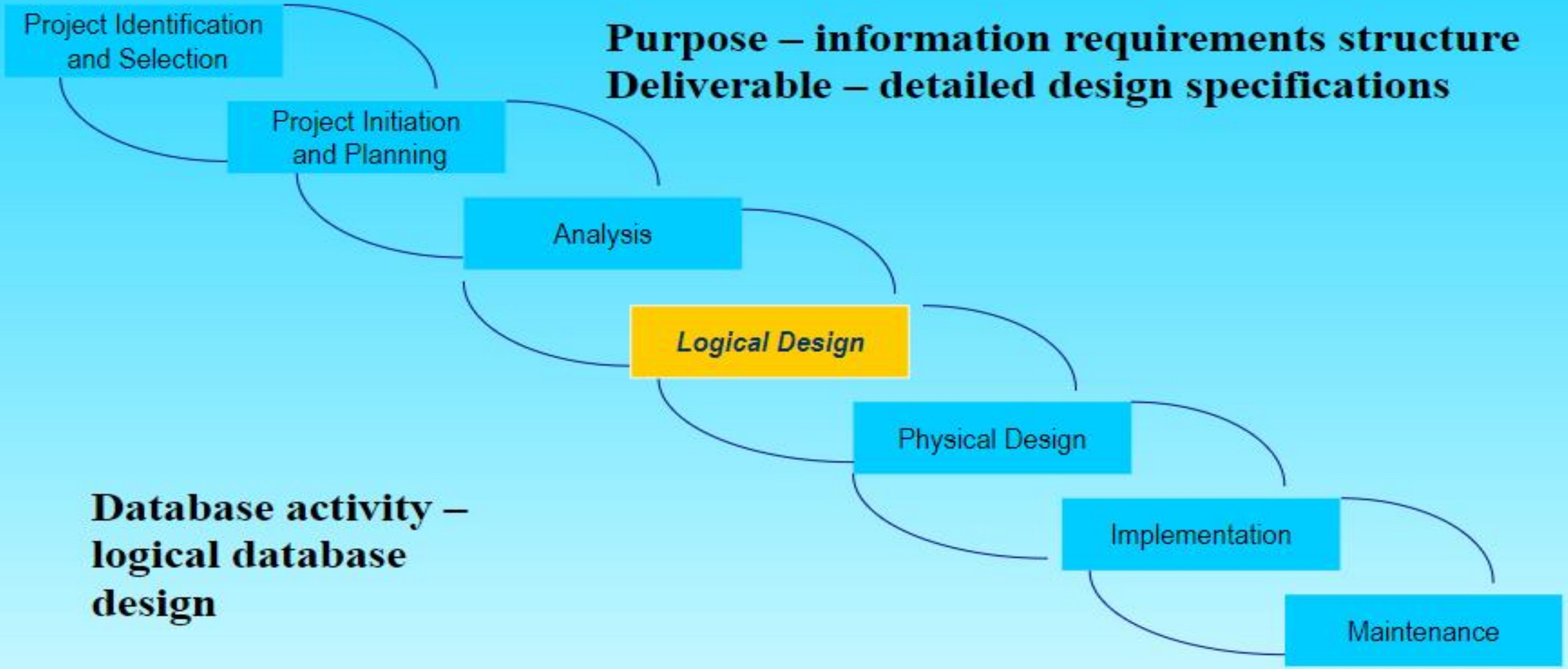
TODAY'S OBJECTIVES

At the end of this lesson, the students must have:

- Analysed and identified entities, properties, keys, relationships, normalization, and cardinalities based on problems
- Produced relational schema on a given problem
- Mapped ER or EER model to relational schema

The Physical Design Stage of SDLC (Figures 2-4, 2-5 revisited)

Purpose – information requirements structure
Deliverable – detailed design specifications



Database activity – logical database design

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 and rows must be irrelevant*

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

Relational Database Concepts

A **relational database** is a database whose logical structure is made up of nothing but a collection of relations.

- ✓ is the result of the work of one man—Edgar (E. F.) Codd.
- ✓ In mathematical set theory, a relation is the definition of a table with columns (**attributes**) and rows (**tuples**).
- ✓ The word "**table**" is used synonymously with "**relation**" in the relational data model. When you include rows of data, you have an **instance of a relation**.

Relation

Customer Number	First Name	Last Name	Phone
0001	Jane	Doe	(555) 555-1111
0002	John	Doe	(555) 555-2222
0003	Jane	Smith	(555) 555-3333
0004	John	Smith	(555) 555-4444

A simple customer relation

Columns And Column Characteristics

A column in a relation has the following properties:

- A **name that is unique** within the table: Two or more tables within the same relational database schema may have columns with the same names—in fact, as you will see shortly, in some circumstances this is highly desirable—but a single table must have unique column names.

- A **domain**: The values in a column are drawn from one and only one domain. As a result, relations are said to be column homogeneous.

Relation

There are **no “positional concepts**.” In other words, the columns can be viewed in any order without affecting the meaning of the data.

Rows and Row Characteristics

- In relational design theory, a row in a relation has the following properties:
- **Only one value** at the intersection of a column and row: A relation does not allow multivalued attributes.

Relation

- **Uniqueness**: There are no duplicate rows in a relation.
- A **primary key**: A primary key is a column or combination of columns with a value that uniquely identifies each row.
- There are **no positional concepts**. The rows can be viewed in any order without affecting the meaning of the data.

Relation

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)*

Types of Tables

A relational database works with two types of tables.

- ✓ **Base tables** are relations that are actually stored in the database. These are the tables that are described by your schema.
- ✓ However, relational operations on tables produce additional tables as their result. Such tables, which exist only in main memory, are known as **virtual tables**.

Relation

Notation for Relations

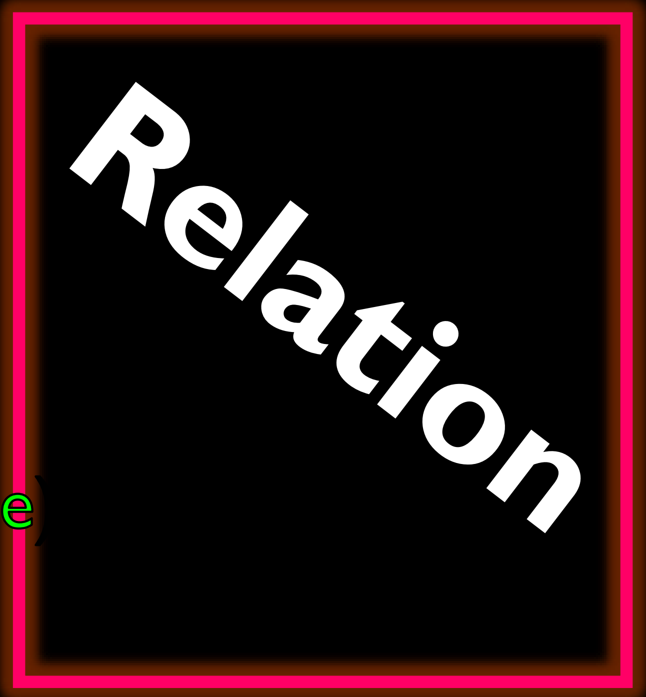
relation_name (primary_key, non_primary_key_column …)

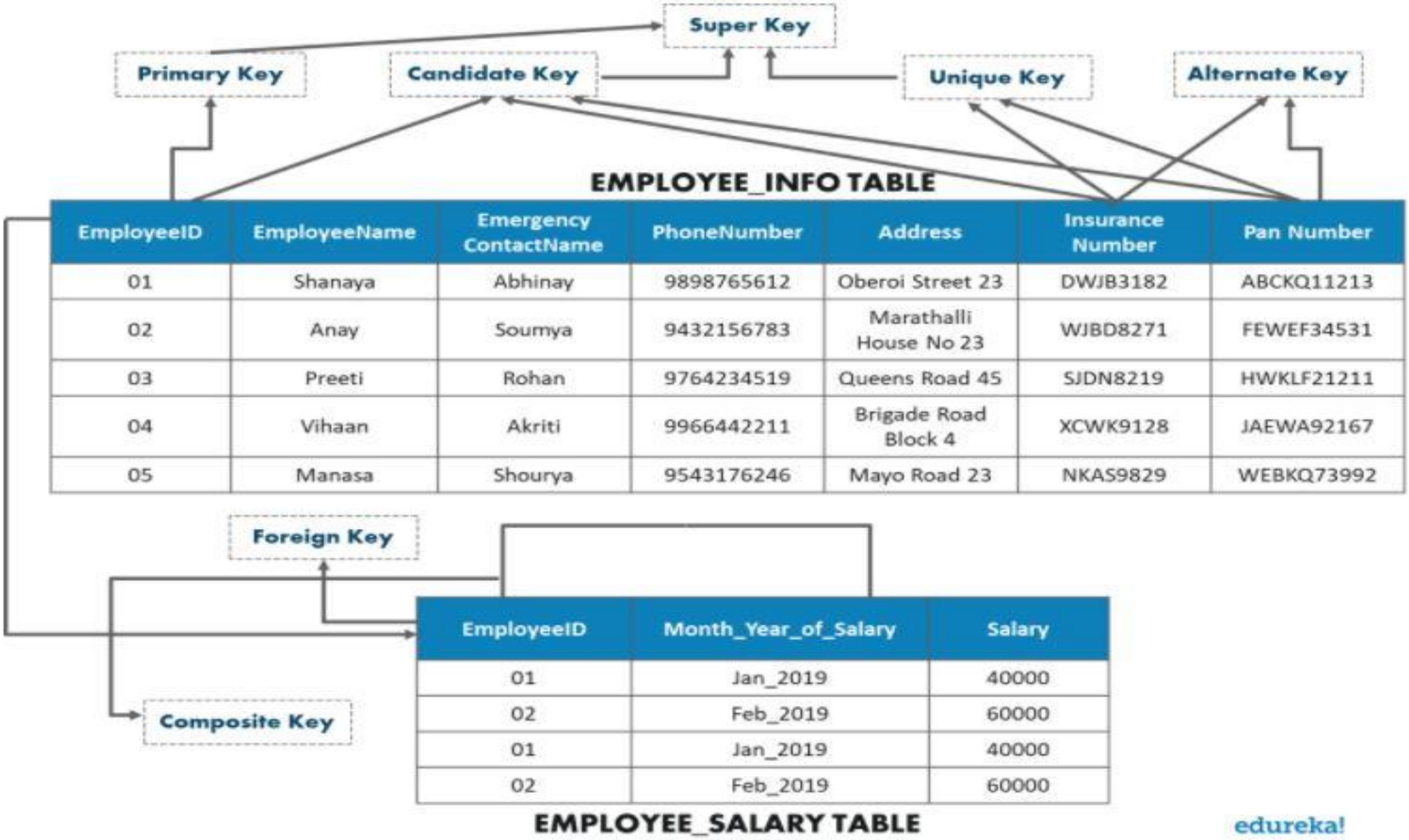
Example:

Customer (customer_number, first_name, last_name, phone)

Products (product_id, product_name, price)

Books (Book_id, Title, Description, Num_copies)





EMPLOYEE_INFO TABLE

EmployeeID	EmployeeName	Emergency ContactName	PhoneNumber	Address	Insurance Number	Pan Number
01	Shanaya	Abhinay	9898765612	Oberoi Street 23	DWJB3182	ABCKQ11213
02	Anay	Soumya	9432156783	Marathalli House No 23	WJBD8271	FEWEF34531
03	Preeti	Rohan	9764234519	Queens Road 45	SJDN8219	HWKLF21211
04	Vihaan	Akriti	9966442211	Brigade Road Block 4	XCWK9128	JAEWA92167
05	Manasa	Shourya	9543176246	Mayo Road 23	NKAS9829	WEBKQ73992

EmployeeID	Month_Year_of_Salary	Salary
01	Jan_2019	40000
02	Feb_2019	60000
01	Jan_2019	40000
02	Feb_2019	60000

EMPLOYEE_SALARY TABLE

CORRESPONDENCE WITH E-R MODEL

- Keys are special fields that serve two main purposes:
    - ➢ ***PRIMARY KEYS*** are unique identifiers of the relation in question. Examples include employee numbers, social security numbers, etc. *This is how we can guarantee that all rows are unique*
    - ➢ ***FOREIGN KEYS*** are identifiers that enable a dependent relation (on the many side of a relationship) to refer to its parent relation (on the one side of the relationship)
- Keys can be ***simple*** (a single field) or ***composite*** (more than one field)
- Keys usually are used as indexes to speed up the response to user queries

Key Fields

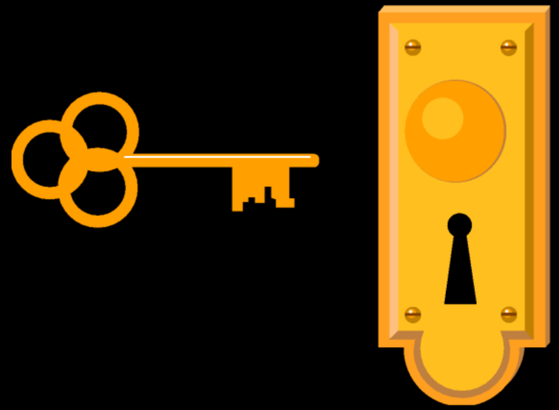


Figure 5-3 Schema for four relations (Pine Valley Furniture Company)

CUSTOMER

Customer_ID	Customer_Name	Customer_Address	City*	State*	Postal_Code*

Primary Key

ORDER

Order_ID	Order_Date	Customer_ID

Foreign Key
(implements 1:N relationship between customer and order)

ORDER LINE

Order_ID	Product_ID	Ordered_Quantity

Combined, these are a *composite primary key* (uniquely identifies the order line)…individually they are *foreign keys* (implement M:N relationship between order and product)

PRODUCT

Product_ID	Product_Description	Product_Finish	Standard_Price	Product_Line_ID

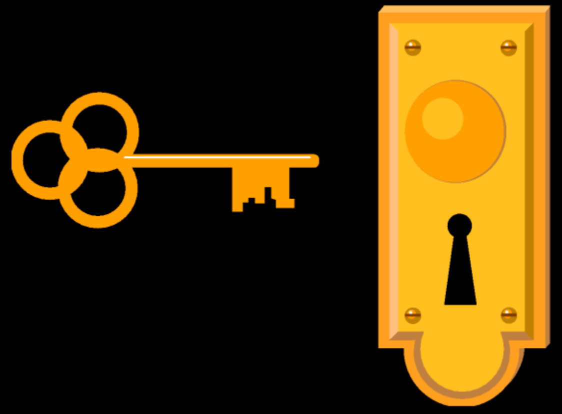
* Not in Figure 3-22 for simplicity.

ALTERNATE KEY – Alternate Keys are the candidate keys, which are not chosen as a Primary key. From the above example, the alternate keys are PanNumber and Insurance Number.

UNIQUE KEY – The unique key is similar to the primary key, but allows one NULL value in the column. Here the Insurance Number and the Pan Number can be considered as unique keys.

COMPOSITE KEY – A composite key is a combination of two or more columns that identify each tuple uniquely. Here, the Employee_ID and Month-Year_Of_Salary can be grouped together to uniquely identify every tuple in the table.

Key Fields



- **Domain Constraints**
  - Allowable values for an attribute.
- **Entity Integrity**
  - No primary key attribute may be null. All primary key fields **MUST** have data

Integrity Constraints

ENTITY INTEGRITY

- the mechanism that provides to maintain primary keys.
- it ensures two properties for primary keys:
  - The primary key for a row is unique; it does not match the primary key of any other row in the table.

Entity Integrity

- The primary key is not null, no component of the primary key may be set to null.

- The uniqueness property ensures that the primary key of each row uniquely identifies it; there are no duplicates. The second property ensures that the primary key has meaning, has a value; no component of the key is missing.

- **The system enforces Entity Integrity by not allowing operations (INSERT, UPDATE) to produce an invalid primary key. Any operation that creates a duplicate primary key or one containing nulls is rejected.**

Entity Integrity

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

REFERENTIAL INTEGRITY

- **Referential Integrity** is the mechanism the system provides to maintain ***foreign keys***.
- The definition of a foreign key must specify the table whose primary key is being referenced. Referential Integrity ensures only one property for foreign keys:

Referential Integrity

Referential Integrity

- While the Referential Integrity property looks simpler than those for Entity Integrity, the consequences are more complex since both primary and foreign keys are involved. The rule for foreign keys is:
  - No operation (INSERT, UPDATE) can create a non-null foreign key unless a corresponding primary key exists.
  - Any operation that produces a non-null foreign key value without a matching primary key value is rejected. Primary keys are also constrained by Referential Integrity:
  - No operation (UPDATE, DELETE) can remove or change a primary key while a referencing foreign keys exist.

Primary Table

CompanyId	CompanyName
1	Apple
2	Samsung

?

Related Table

CompanyId	ProductId	ProductName
1	1	iPhone
15	2	Mustang

Associated Record ✔

Orphaned Record ✘

Here, the related table contains a foreign key value that doesn't exist in the primary key field of the primary table (i.e. the "CompanyId" field). This has resulted in an "orphaned record".

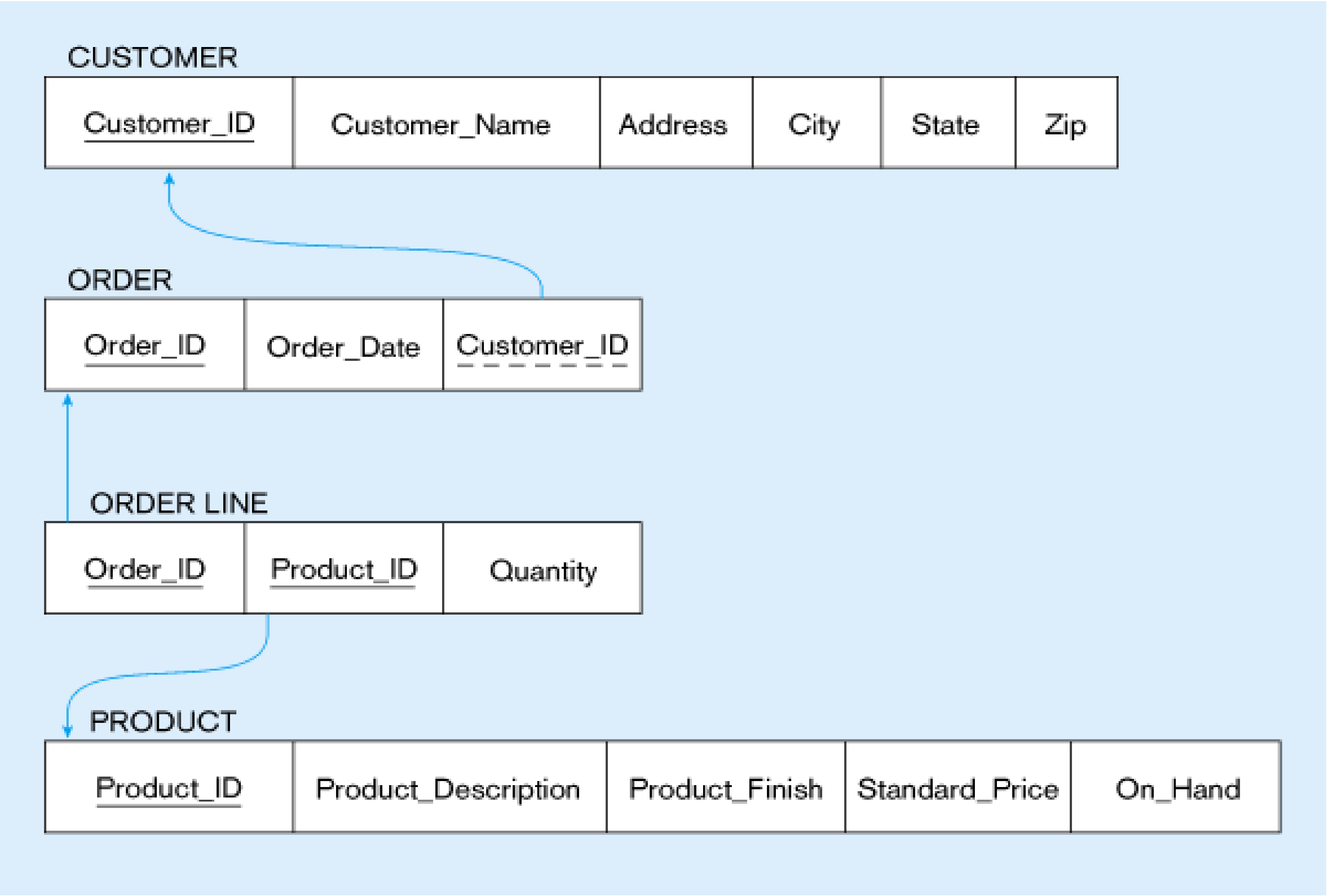
Referential Integrity has a rule that states that any foreign key value (on the relation of the many side) MUST match a primary key value in the relation of the one side. (Or the foreign key can be null)

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 are not allowed for weak entities

Integrity Constraints

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



Integrity Constraints

Figure 5-6 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

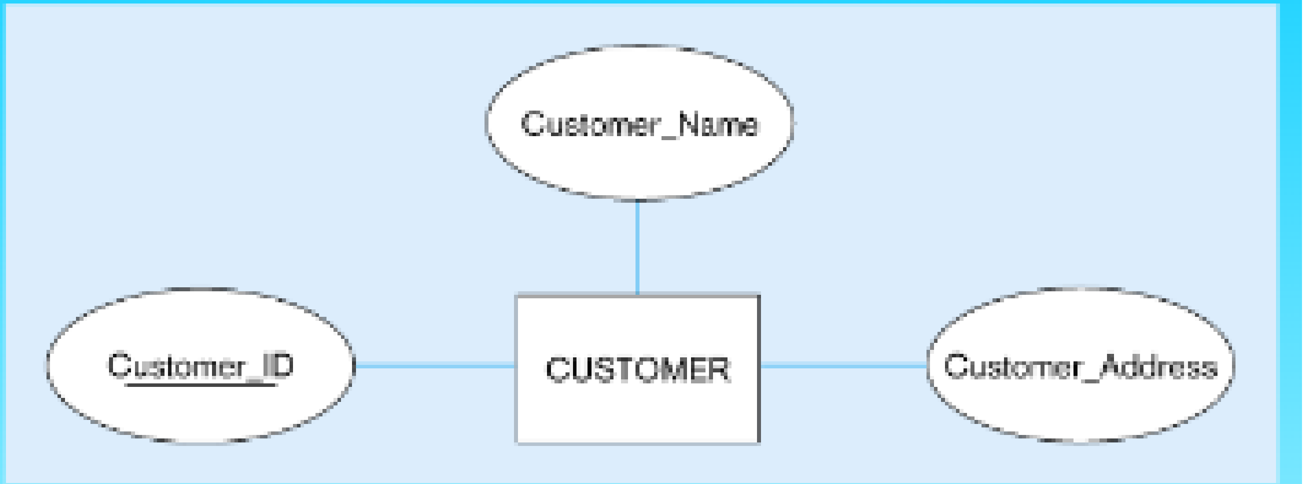
Integrity Constraints

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

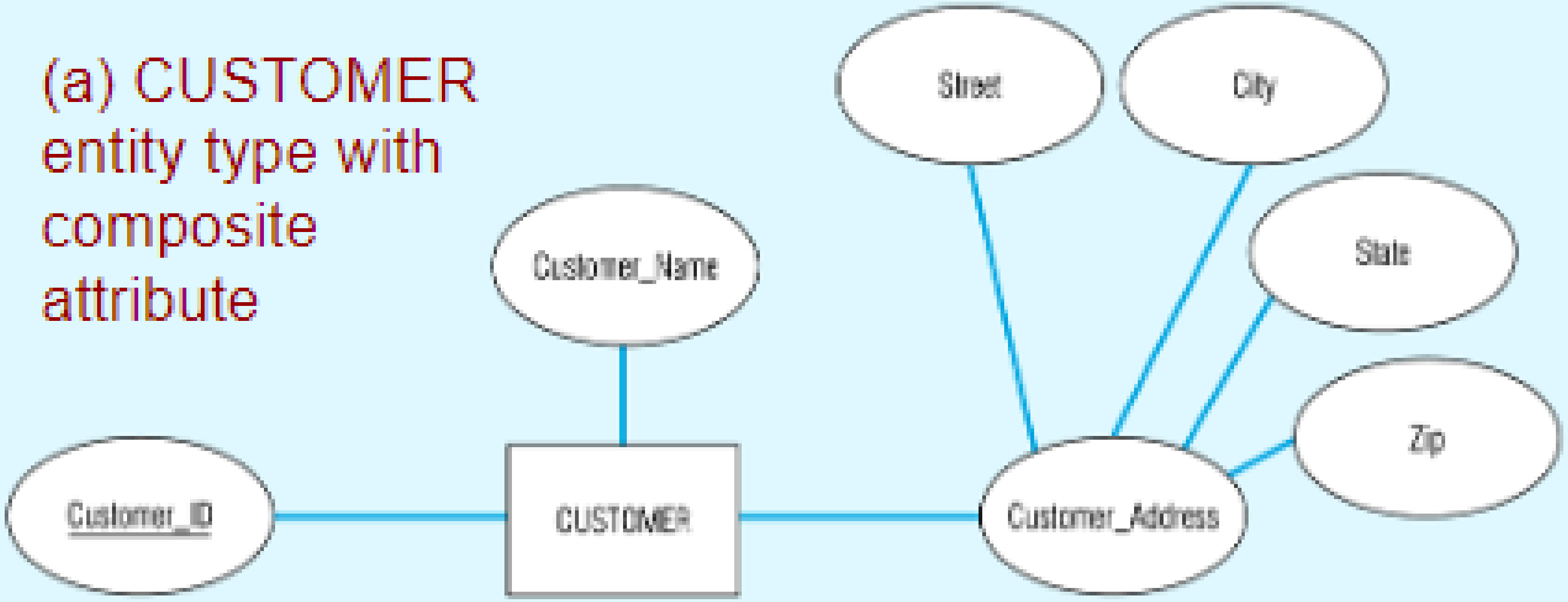
CUSTOMER

Customer_ID	Customer_Name	Customer_Address

Transforming EER Diagrams to Relations

Figure 5-9: Mapping a composite attribute

(a) CUSTOMER entity type with composite attribute

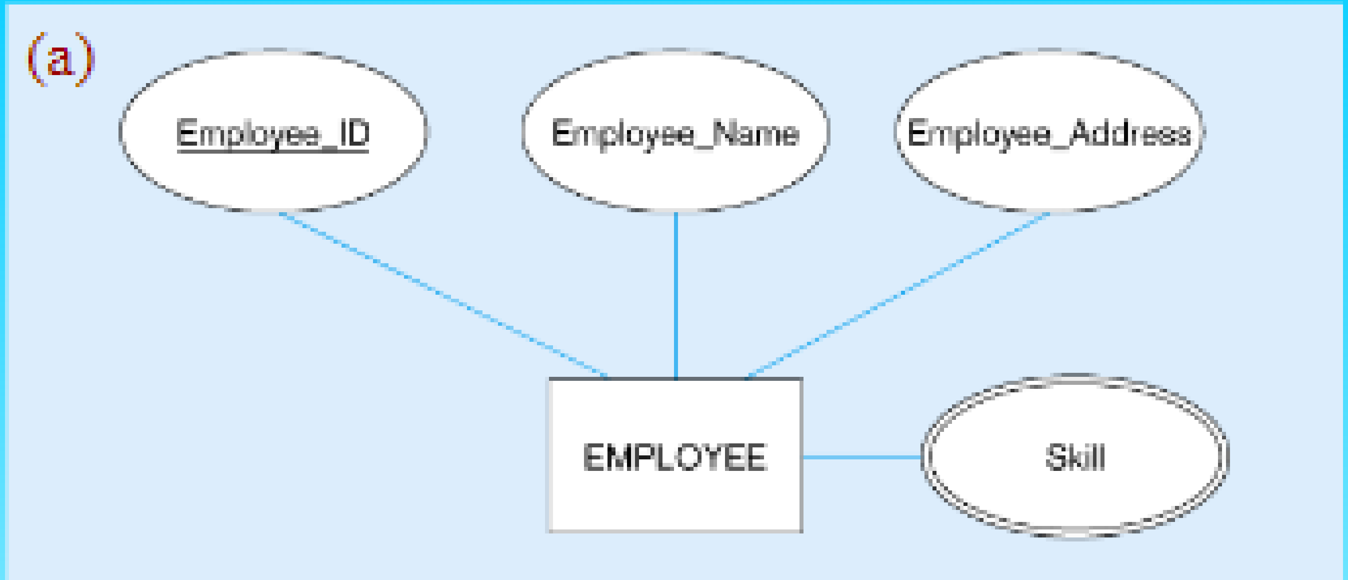


CUSTOMER (b) CUSTOMER relation with address detail

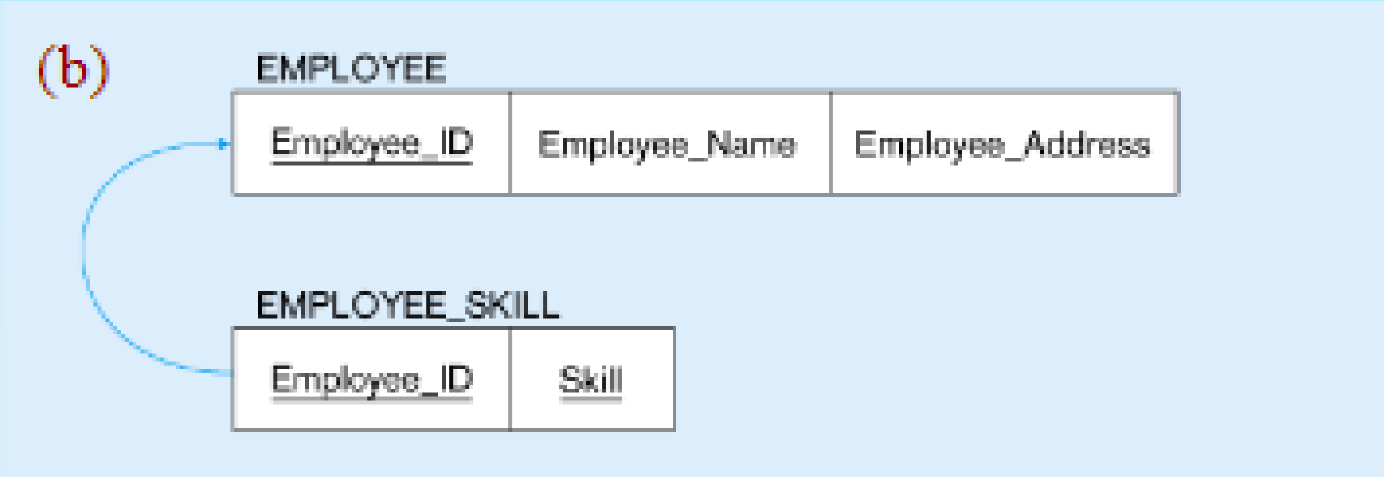
Customer_ID	Customer_Name	Street	City	State	Zip

Transforming EER Diagrams to Relations

Figure 5-10: Mapping a multivalued attribute



Multivalued attribute becomes a separate relation with foreign key



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

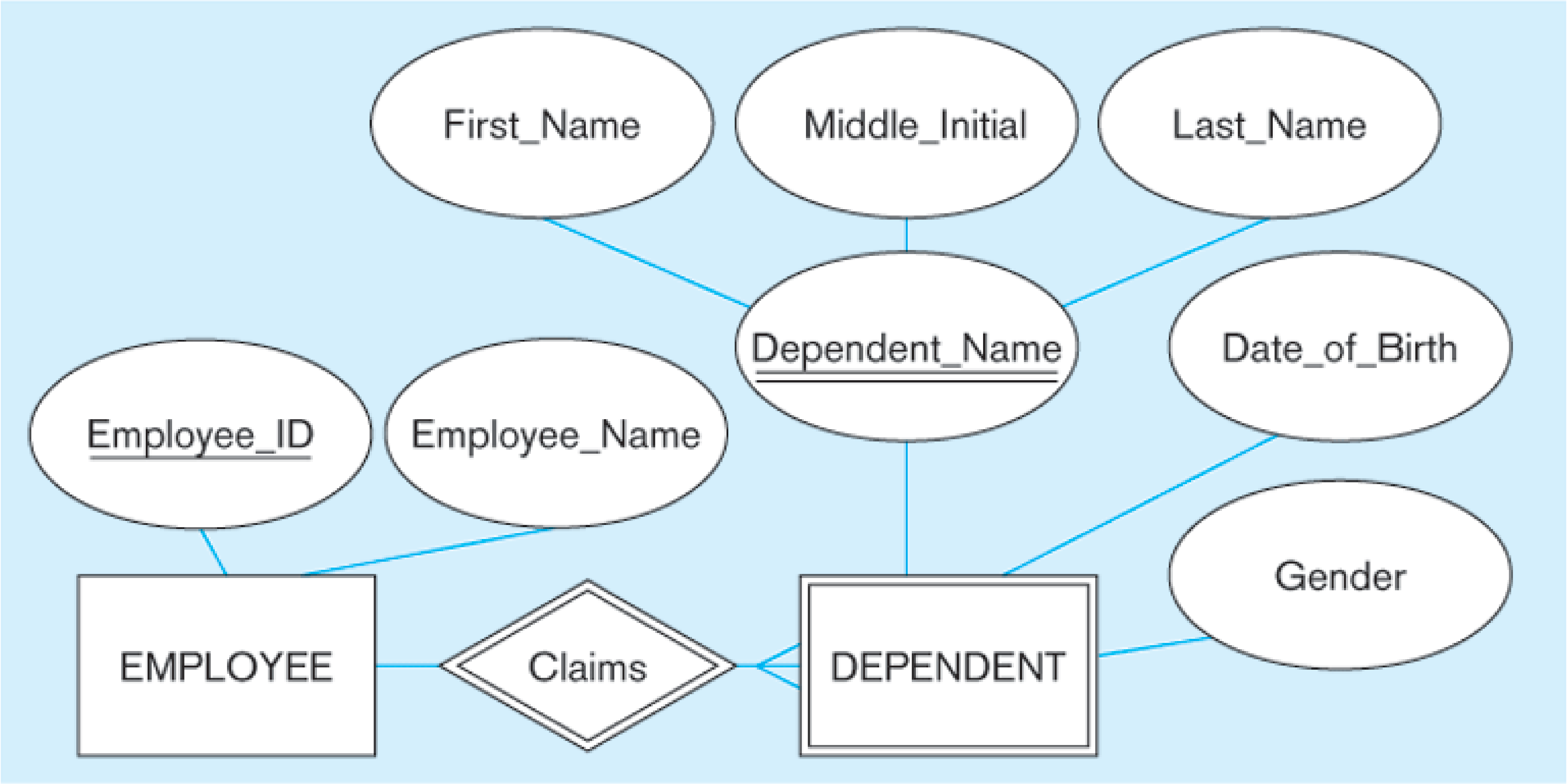
Transforming EER Diagrams to Relations

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)

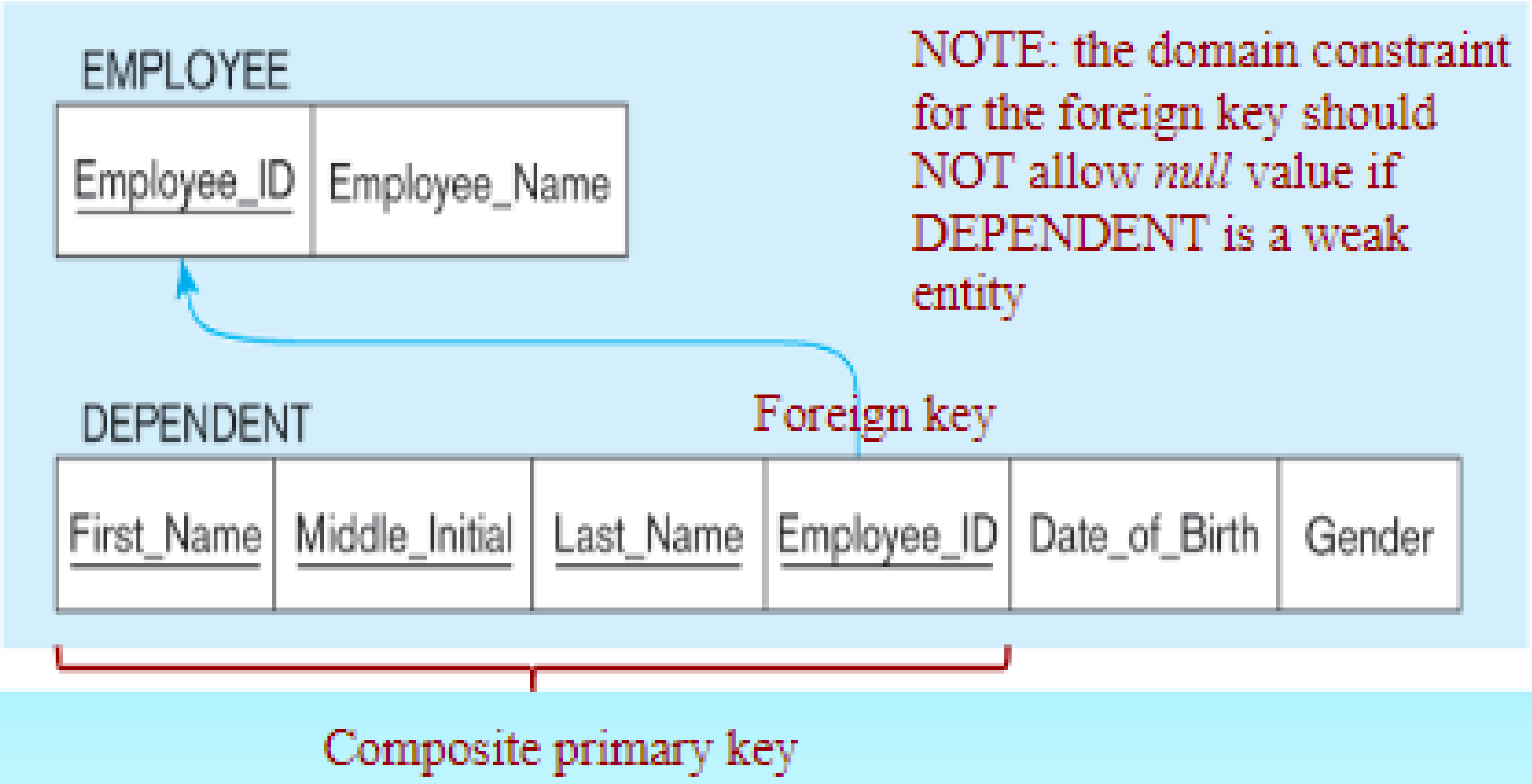
Transforming EER Diagrams to Relations

Figure 5-11a Example of mapping a weak entity - Weak entity DEPENDENT



Transforming EER Diagrams to Relations

Figure 5-11b Example of mapping a weak entity - Relations resulting from weak entity

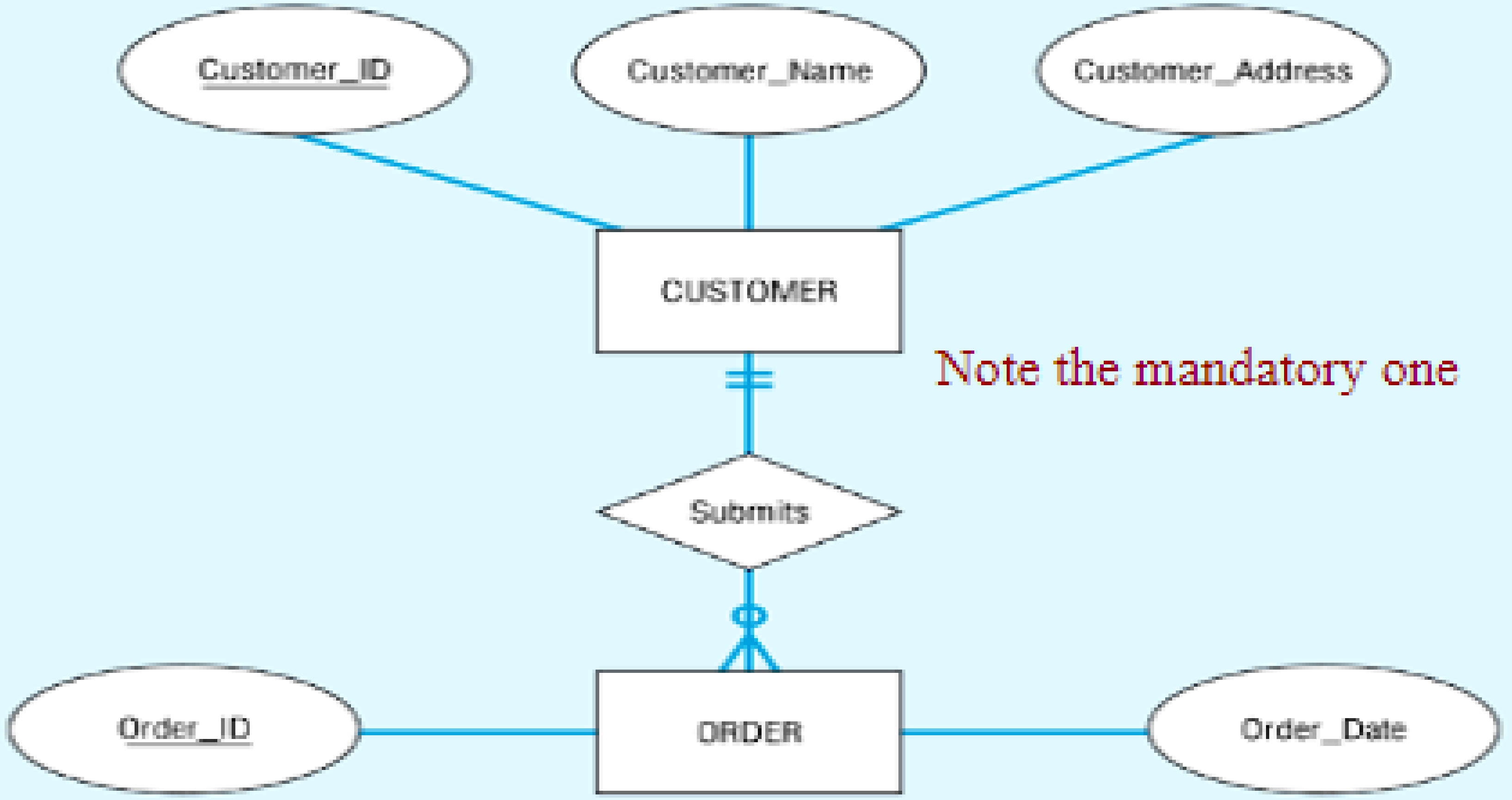


Transforming EER Diagrams to Relations

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

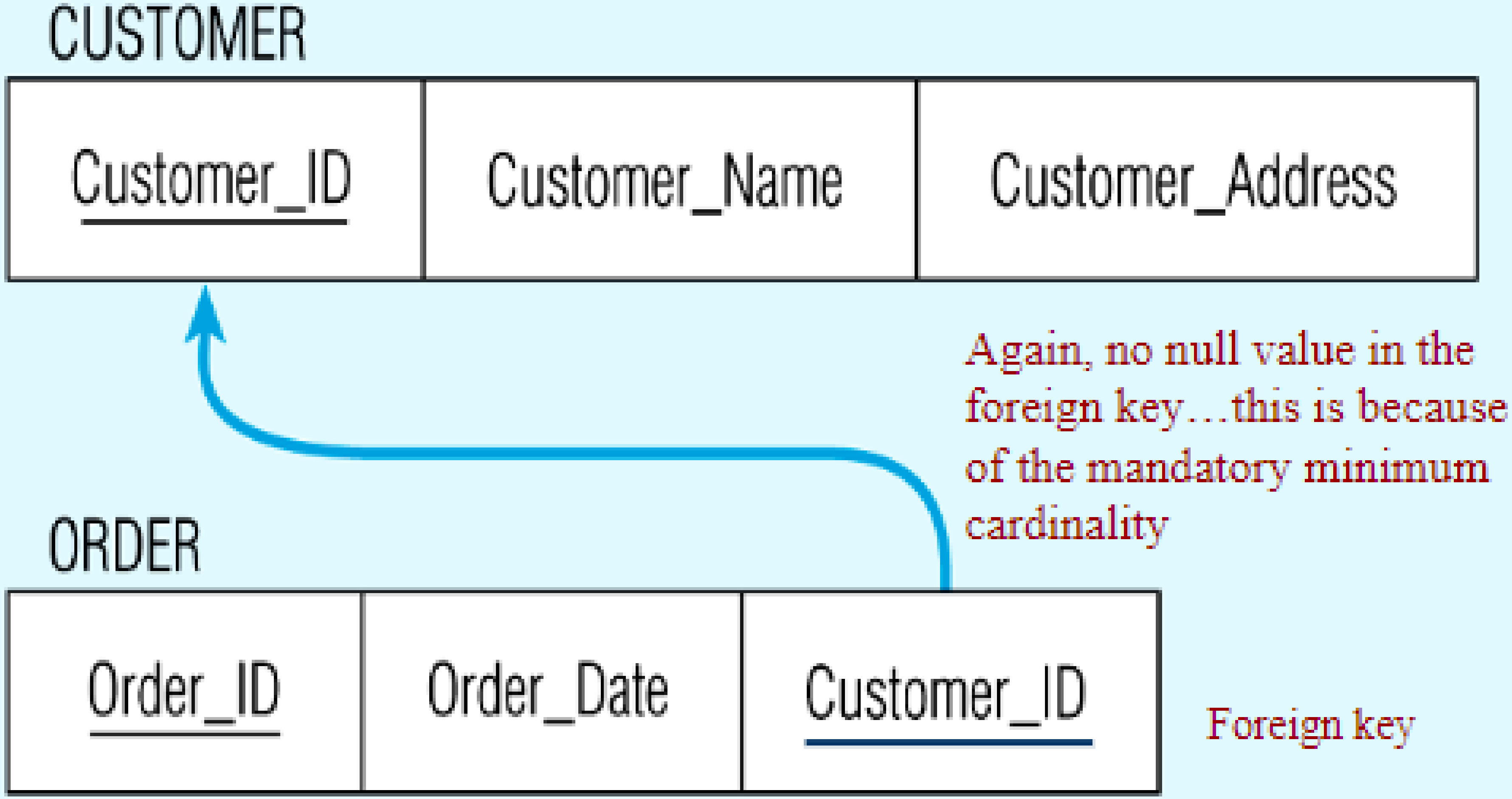
Figure 5-12a: Example of mapping a 1:M relationship
Relationship between customers and orders



Note the mandatory one

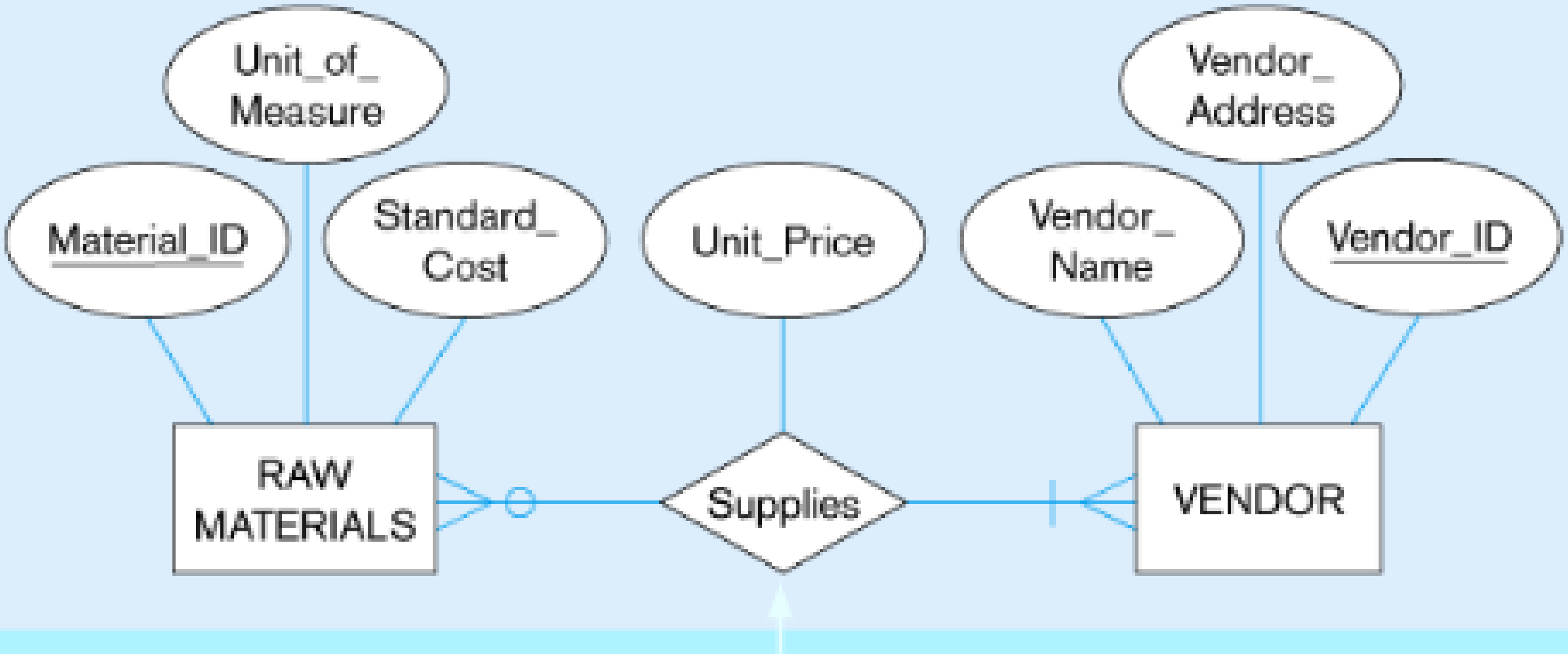
Transforming EER Diagrams to Relations

Figure 5-12b Mapping the relationship



Transforming EER Diagrams to Relations

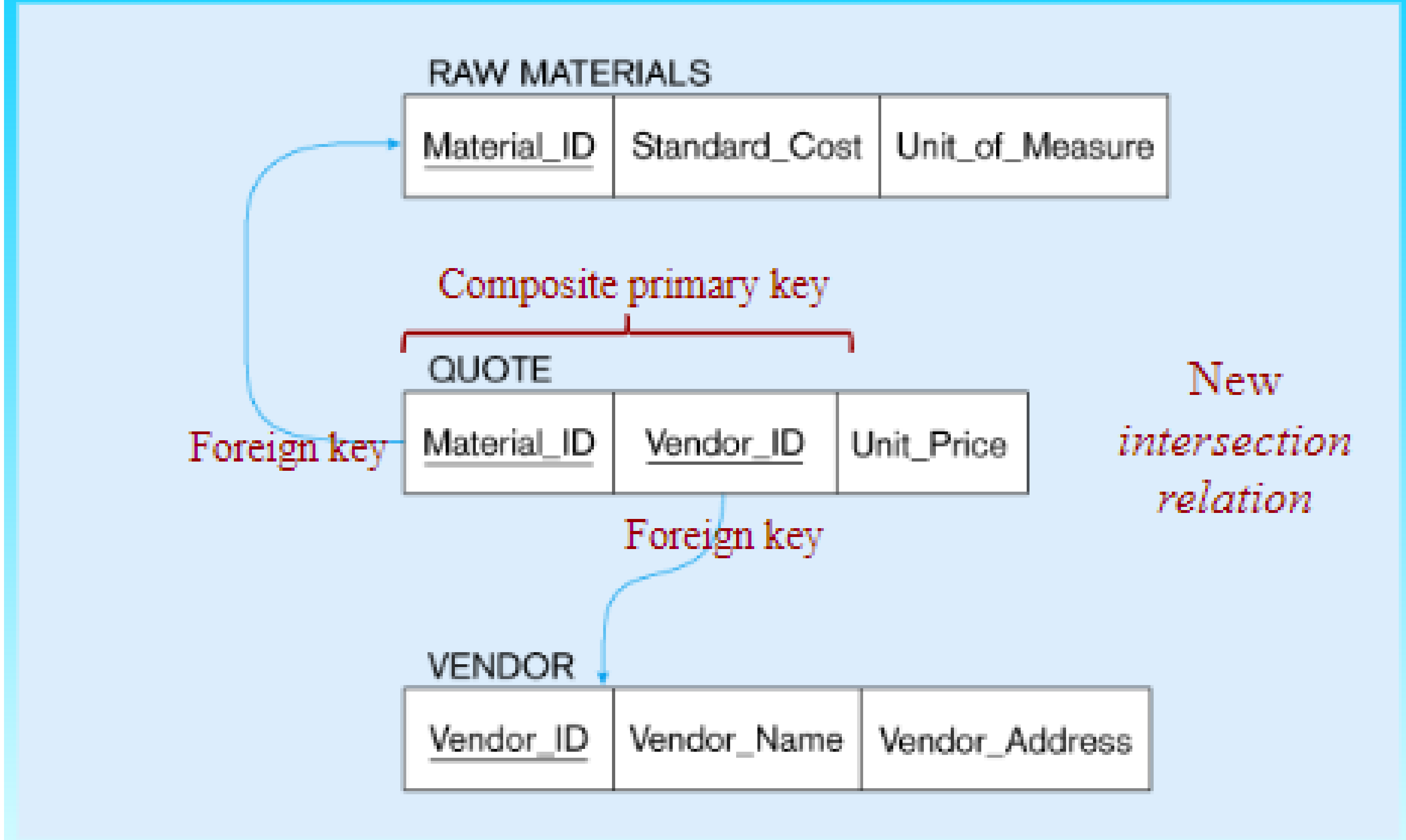
Figure 5-13a: Example of mapping an M:N relationship
E-R diagram (M:N)



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

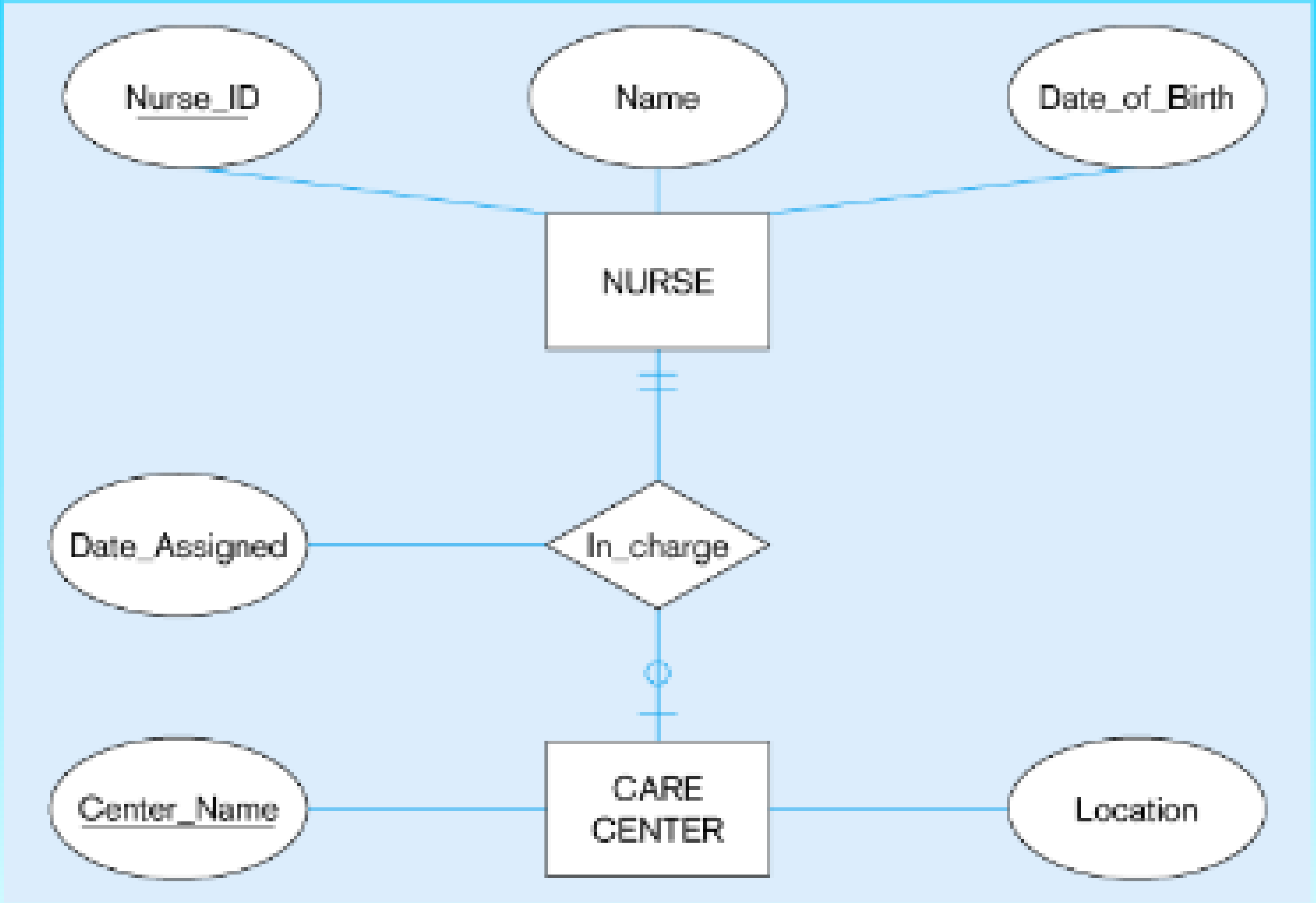
Transforming EER Diagrams to Relations

Figure 5-13b Three resulting relations



Transforming EER Diagrams to Relations

Figure 5-14a: Mapping a binary 1:1 relationship
In_charge relationship



Transforming EER Diagrams to Relations

Figure 5-14b Resulting relations

NURSE

Nurse_ID	Name	Date_of_Birth

CARE CENTER

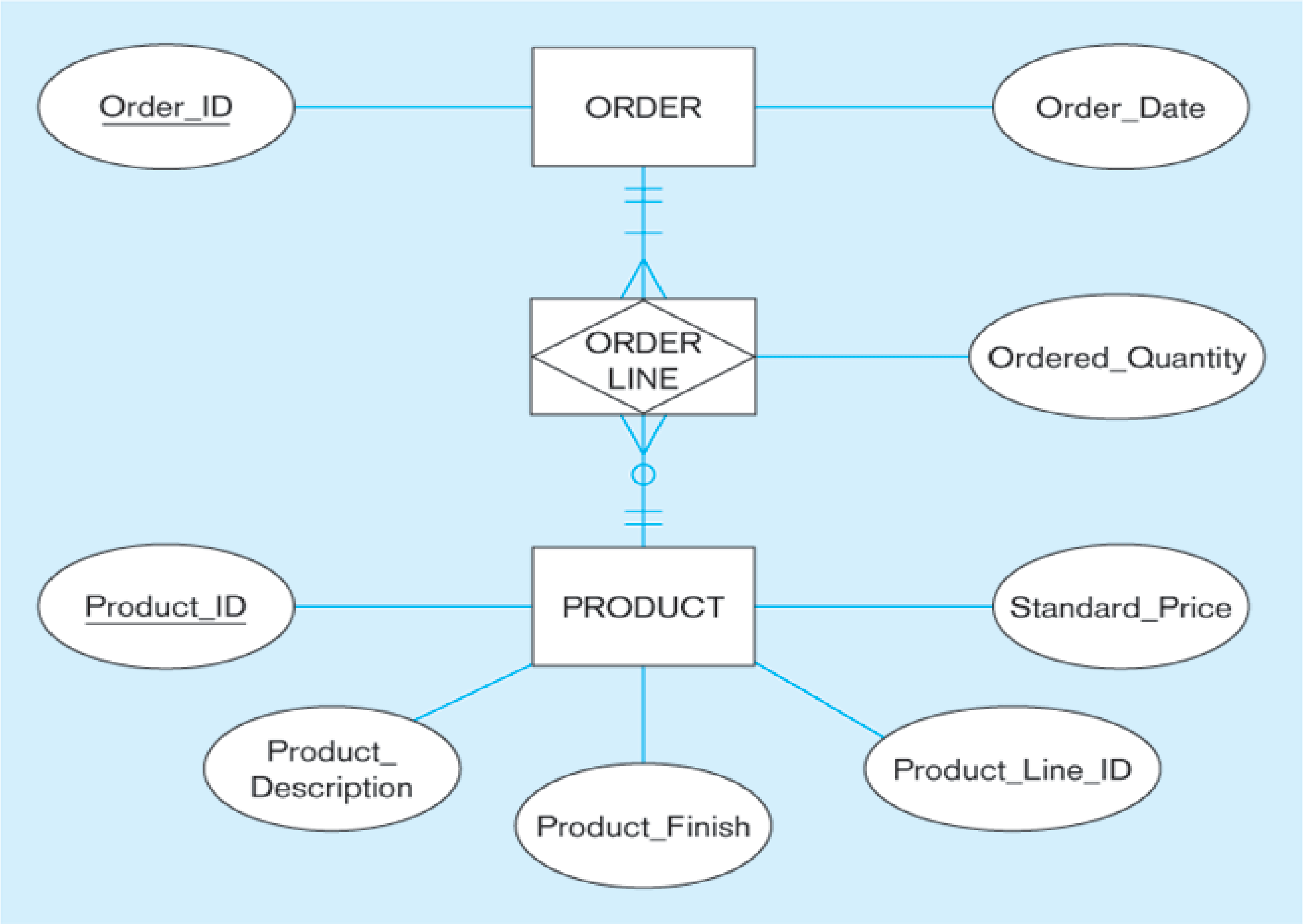
Center_Name	Location	Nurse_in_Charge	Date_Assigned

Transforming EER Diagrams to Relations

Mapping Associative Entities

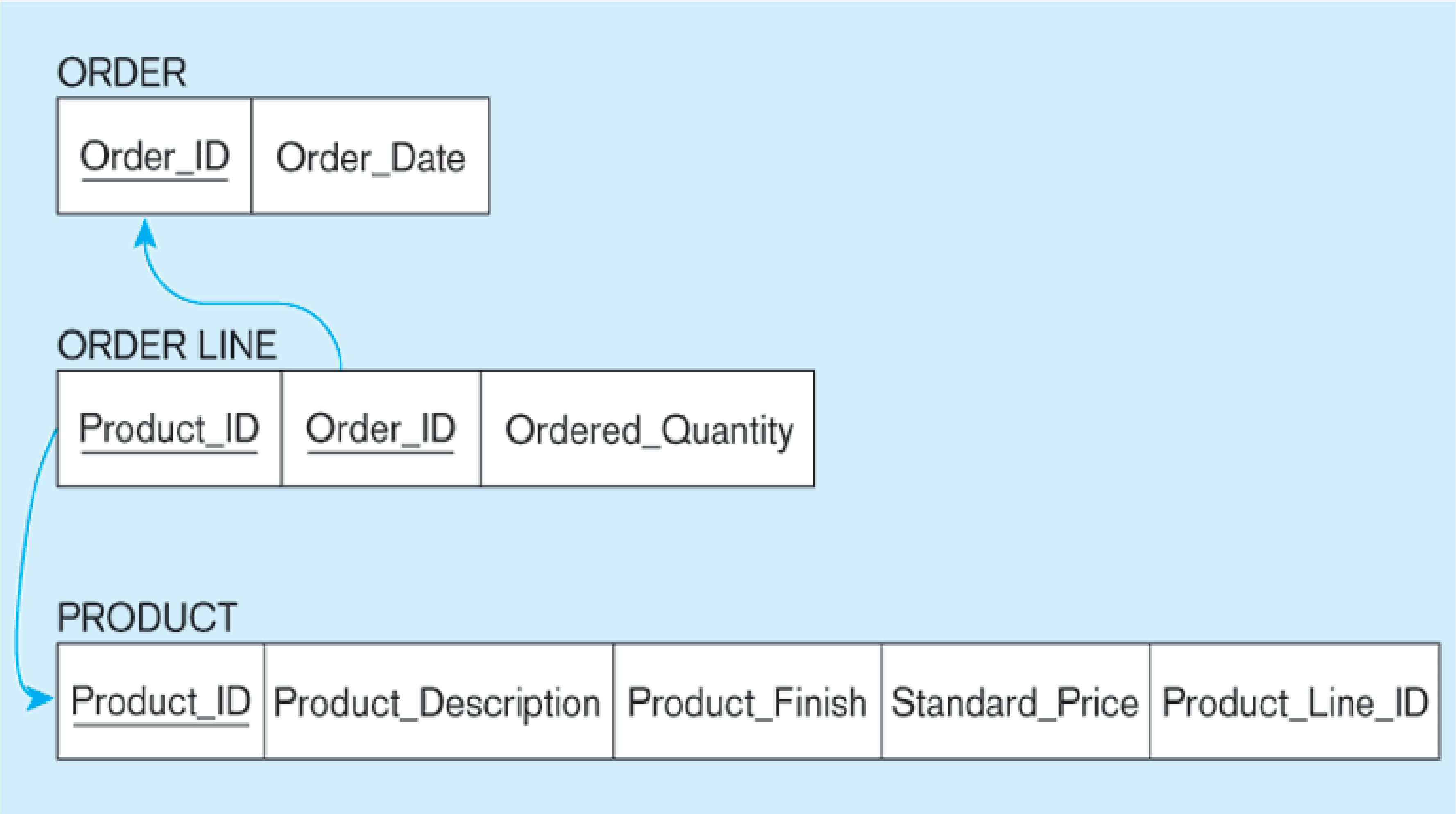
- **Identifier Not Assigned**
  - Default primary key for the association relation is composed of the primary keys of the two entities (as in M:N relationship)
- **Identifier Assigned**
  - It is natural and familiar to end-users
  - Default identifier may not be unique

Figure 5-15a Mapping an associative entity - An associative entity



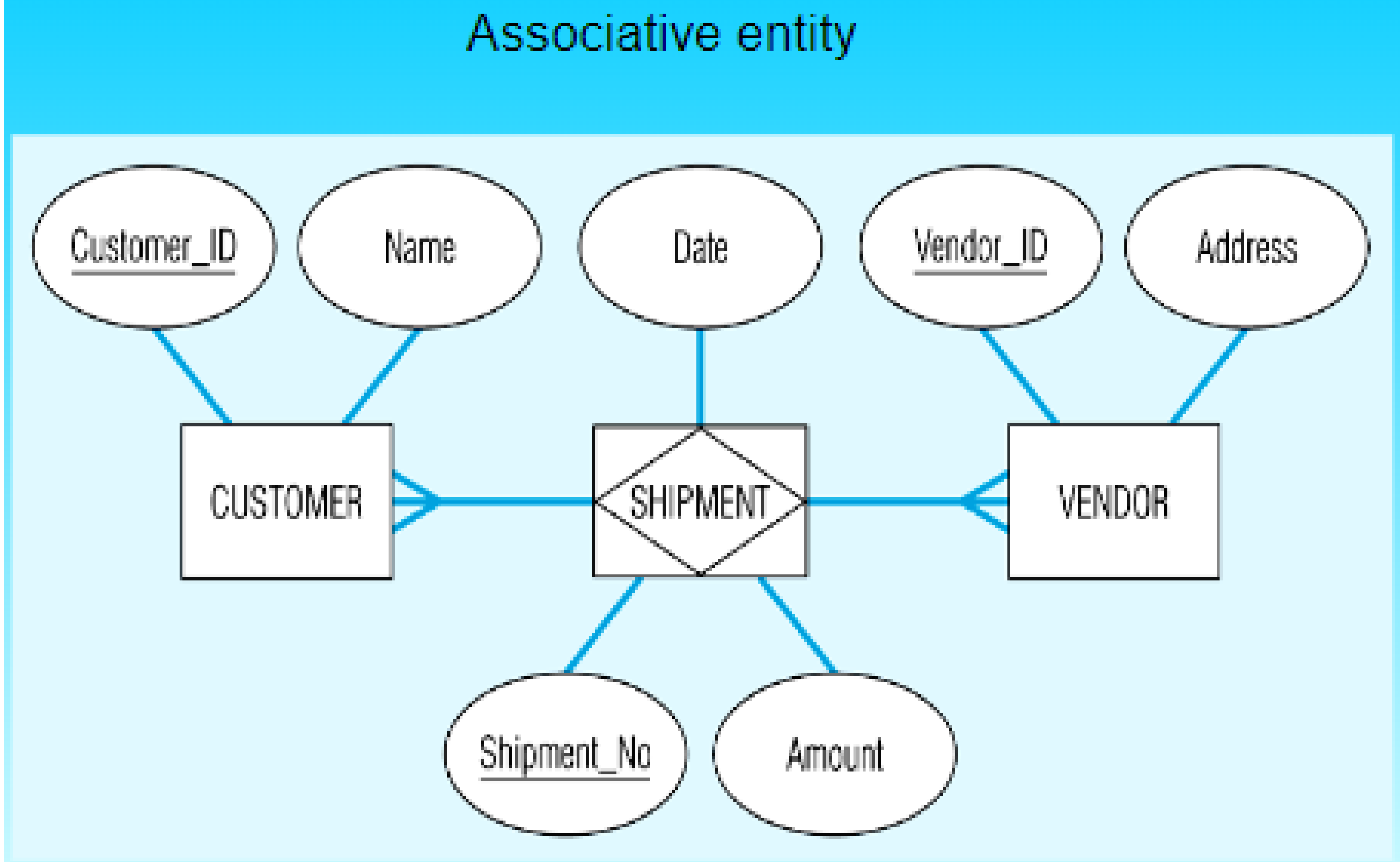
Transforming EER Diagrams to Relations

Figure 5-15b Mapping an associative entity - Three resulting relations



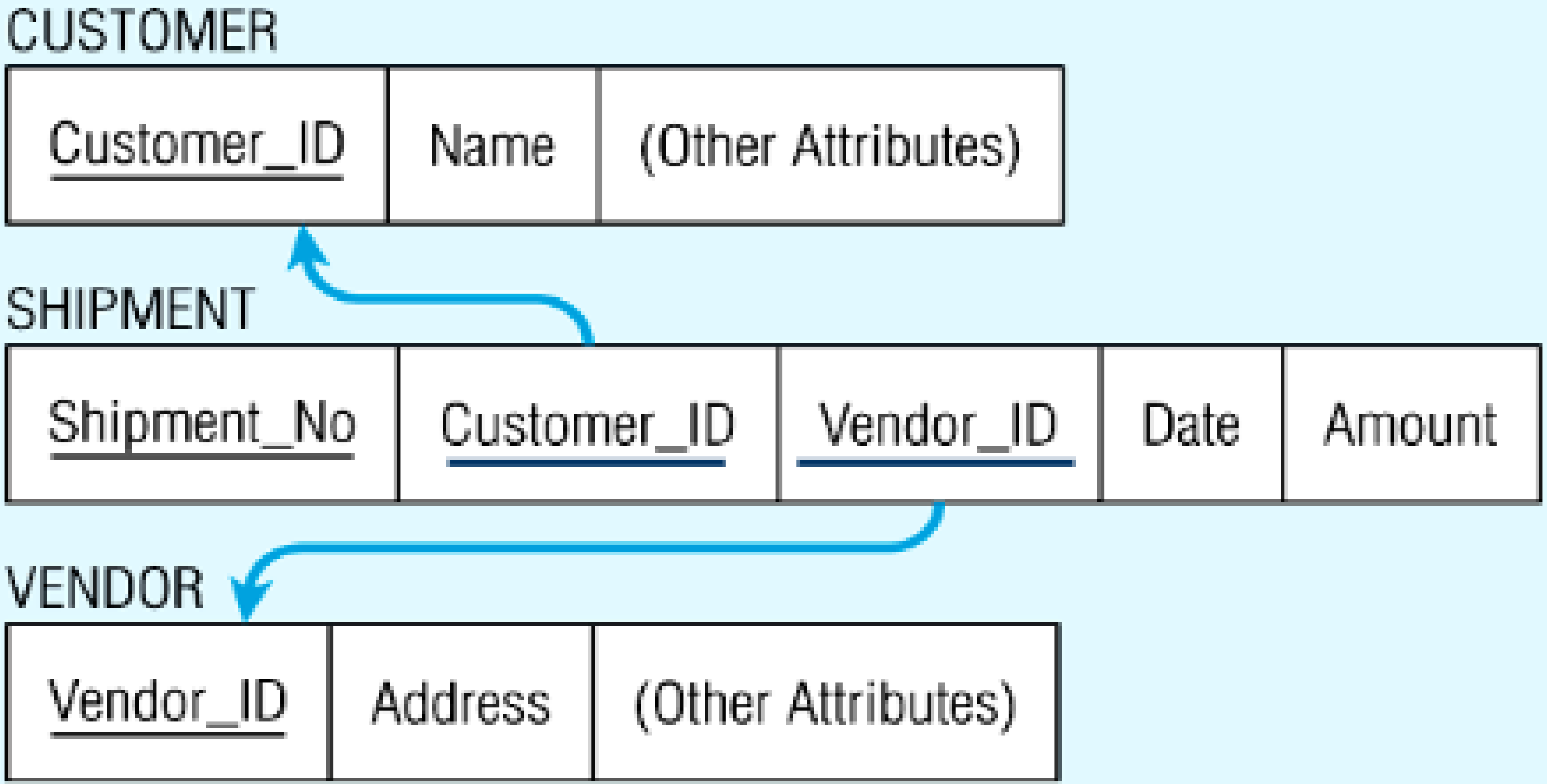
Transforming EER Diagrams to Relations

Figure 5-16a: Mapping an associative entity with an identifier



Transforming EER Diagrams to Relations

Figure 5-16b Three resulting relations



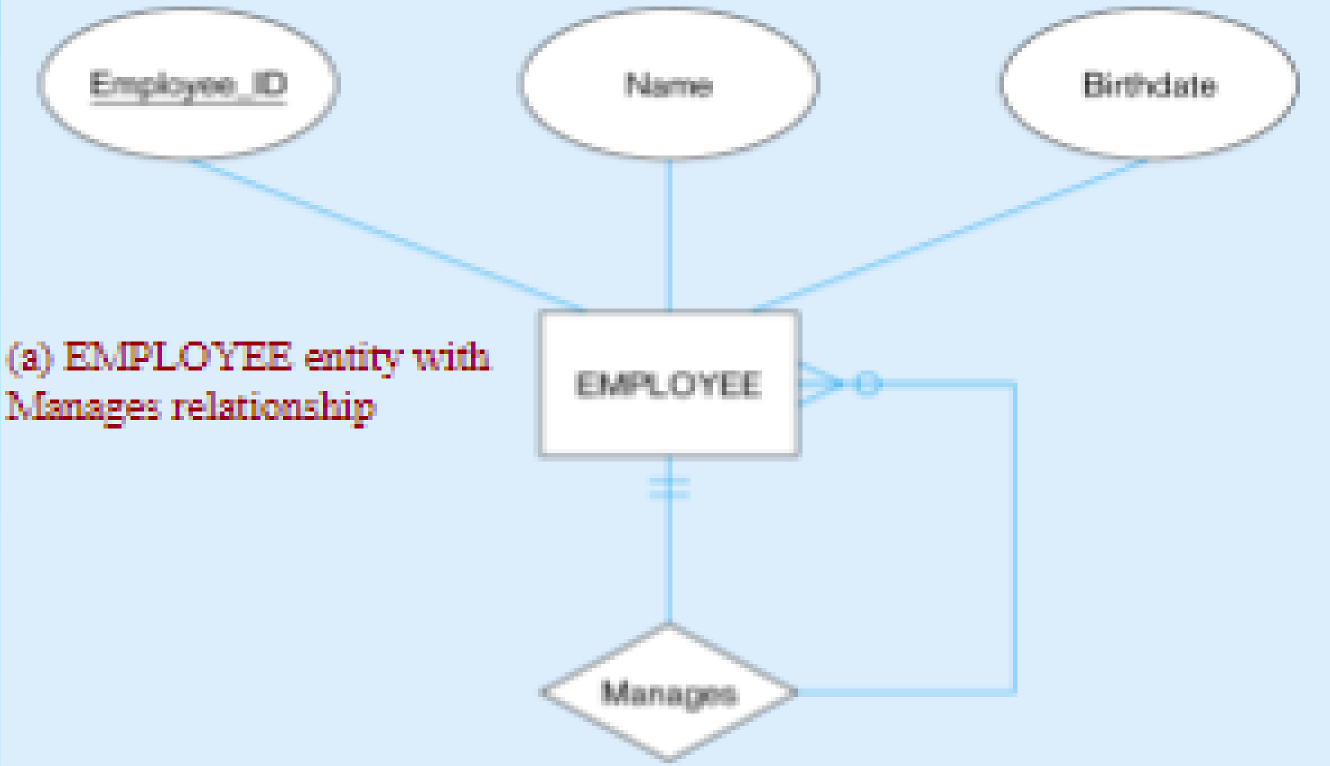
Transforming EER Diagrams to Relations

Mapping Unary Relationships

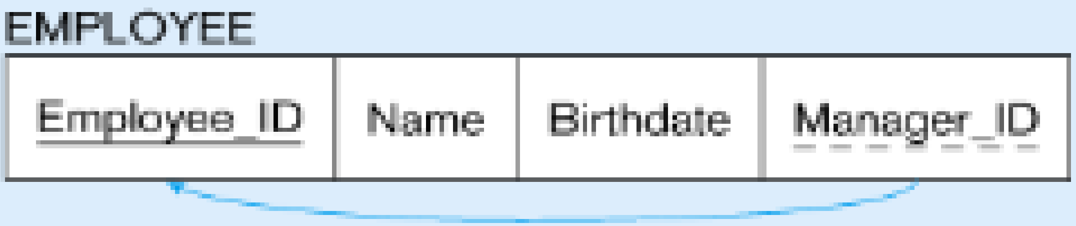
- **One-to-Many** - Recursive foreign key in the same relation
- **Many-to-Many** - Two relations:
  - One for the entity type
  - One for an associative relation in which the primary key has two attributes, both taken from the primary key of the entity

Figure 5-17: Mapping a unary 1:N relationship

(a) EMPLOYEE entity with Manages relationship

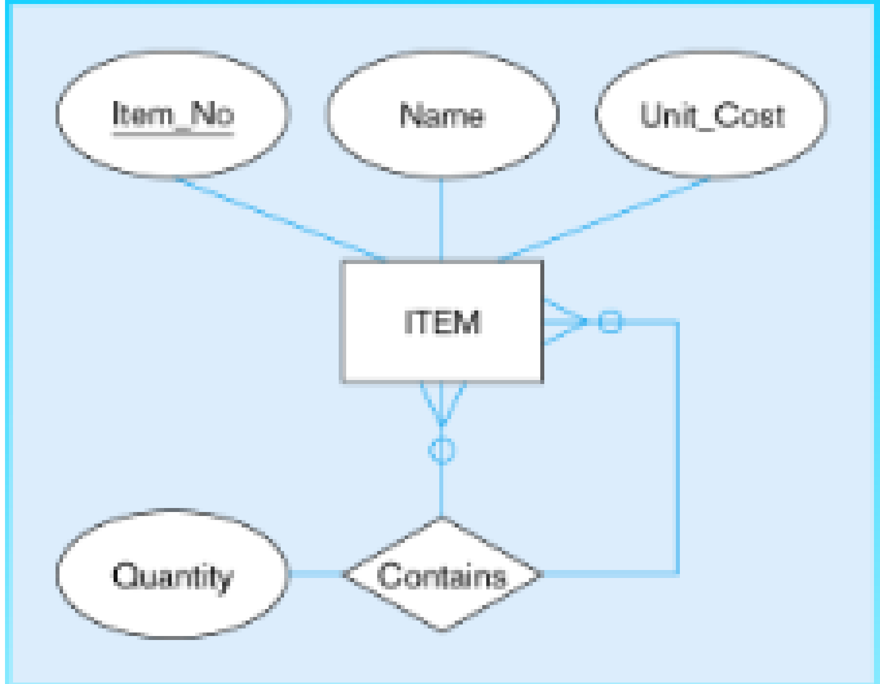


(b) EMPLOYEE relation with recursive foreign key



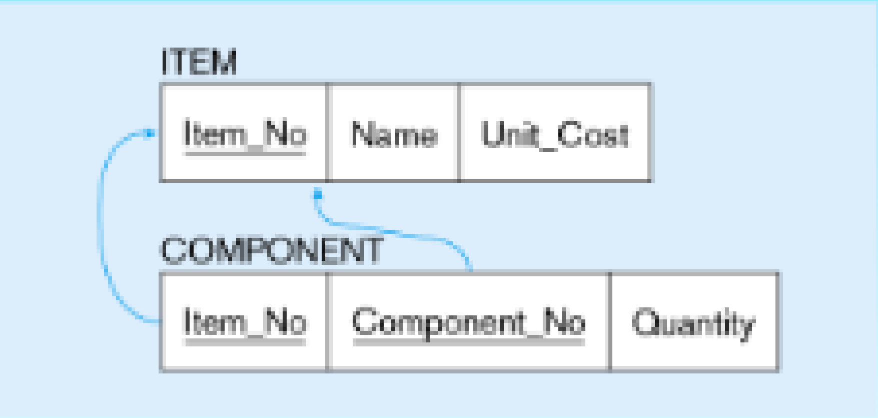
Transforming EER Diagrams to Relations

Figure 5-18: Mapping a unary M:N relationship



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

(b) ITEM and COMPONENT relations



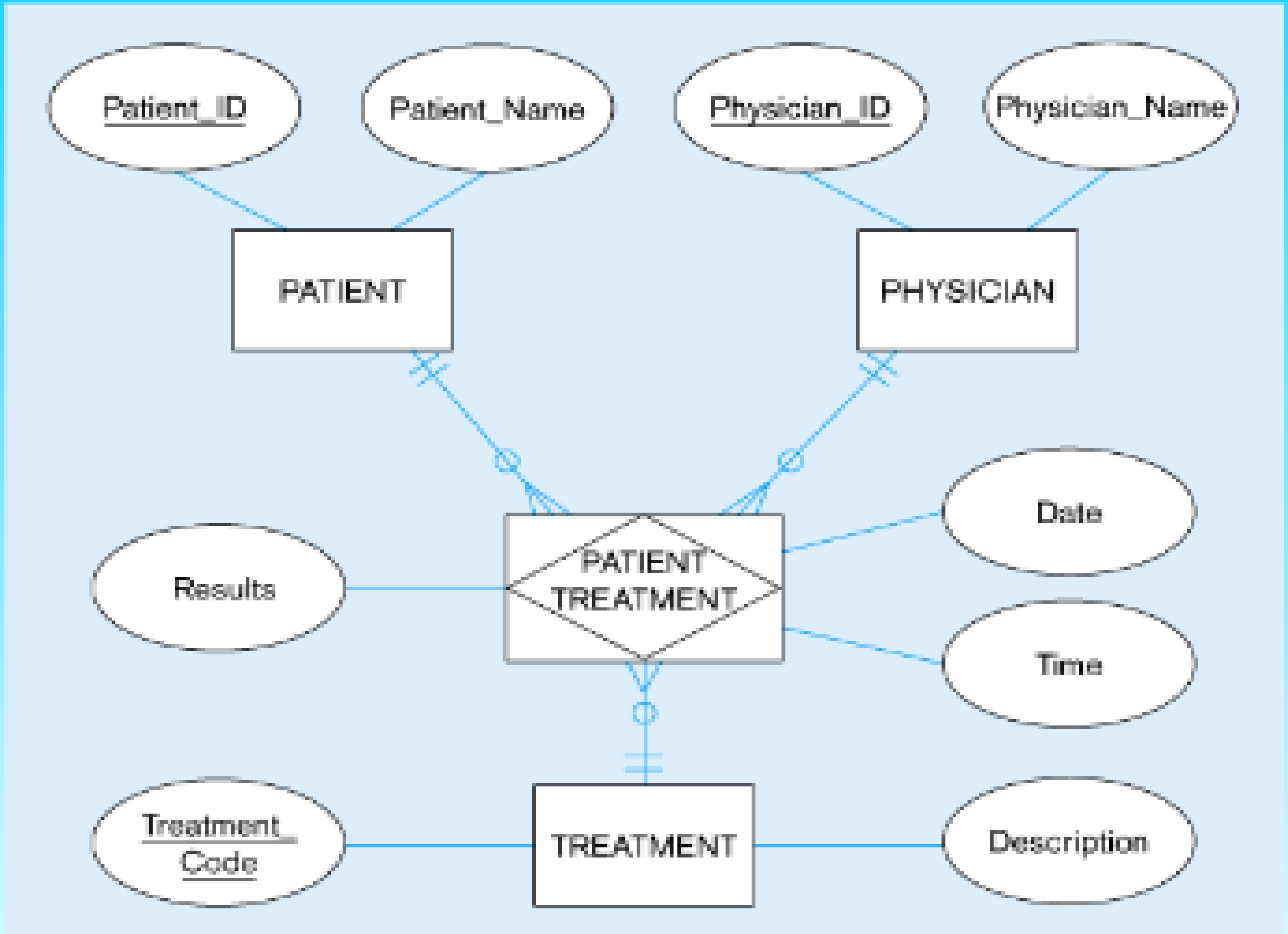
Transforming EER Diagrams to Relations

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

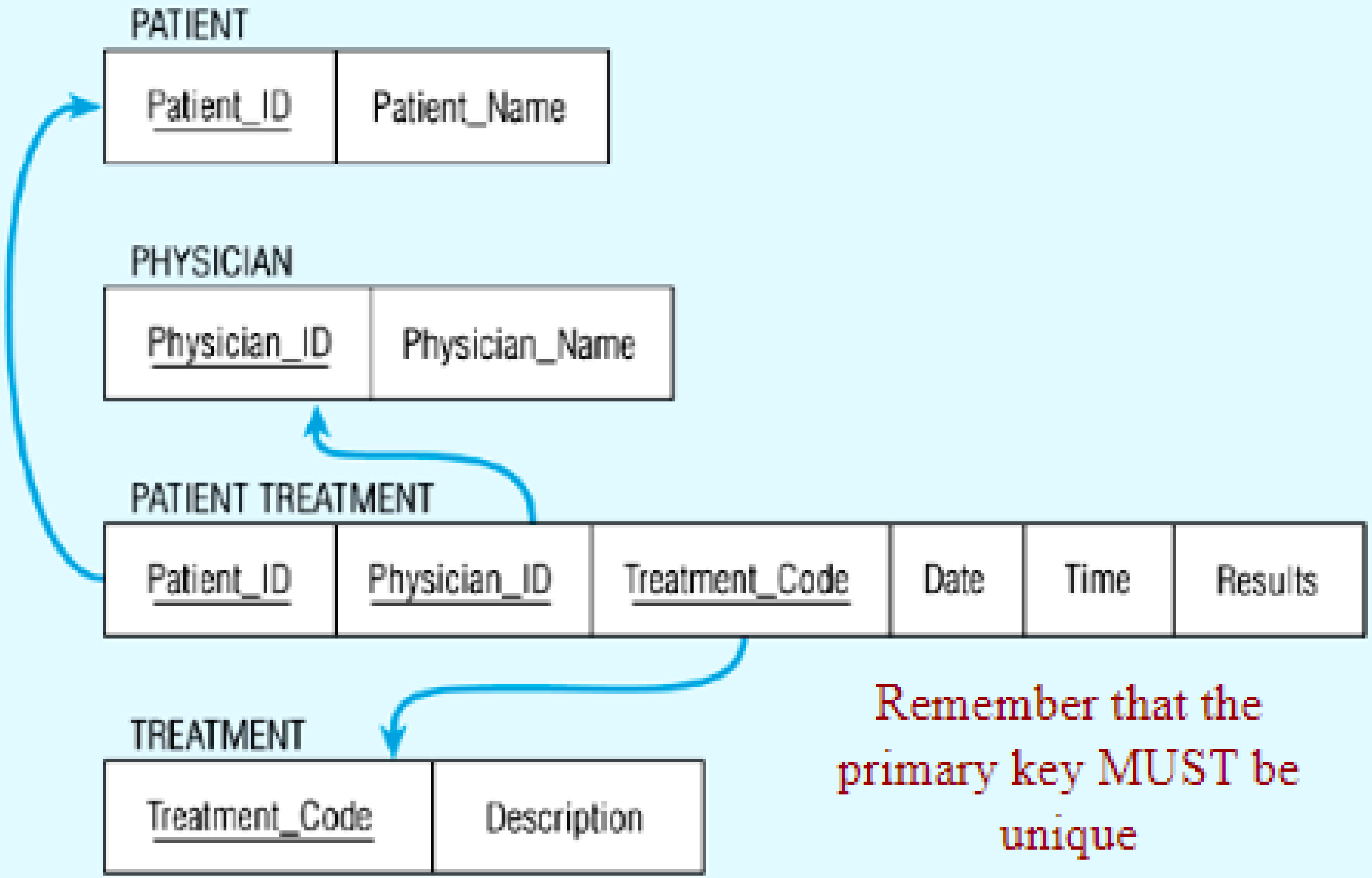
Transforming EER Diagrams to Relations

Figure 5-19a: Mapping a ternary relationship
Ternary relationship with associative entity



Transforming EER Diagrams to Relations

Figure 5-19b Mapping the ternary relationship



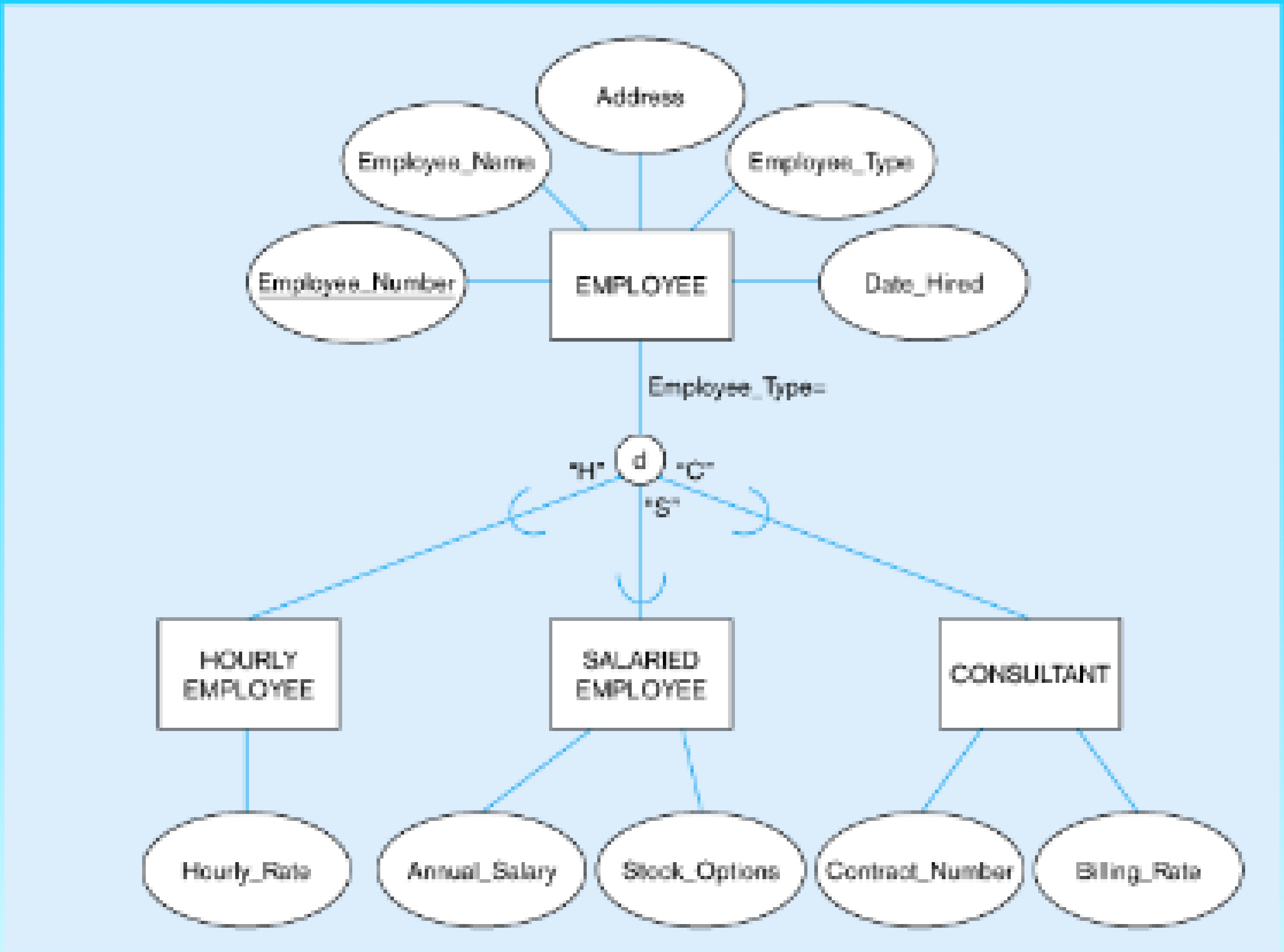
Remember that the primary key MUST be unique

Transforming EER Diagrams to Relations

Mapping Supertype/Subtype Relationships

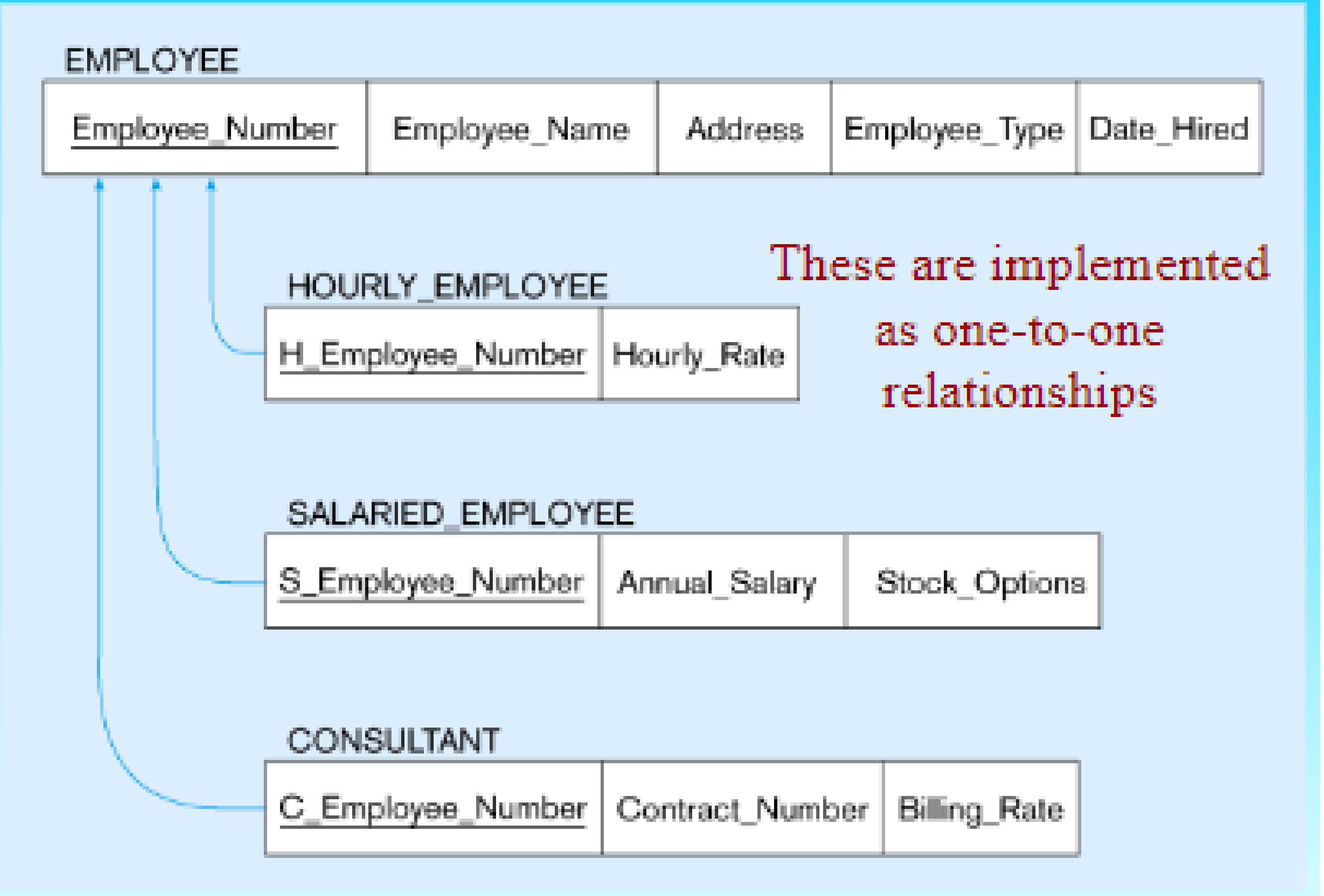
- One relation for supertype and for each subtype
- Supertype attributes (including identifier and subtype discriminator) go into supertype relation
- Subtype attributes go into each subtype; primary key of supertype relation also becomes primary key of subtype relation
- 1:1 relationship established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

Figure 5-20: Supertype/subtype relationships



Transforming EER Diagrams to Relations

Figure 5-21:
Mapping Supertype/subtype relationships to relations



Transforming EER Diagrams to Relations

CONVERTING ERM TO RELATIONAL MODEL

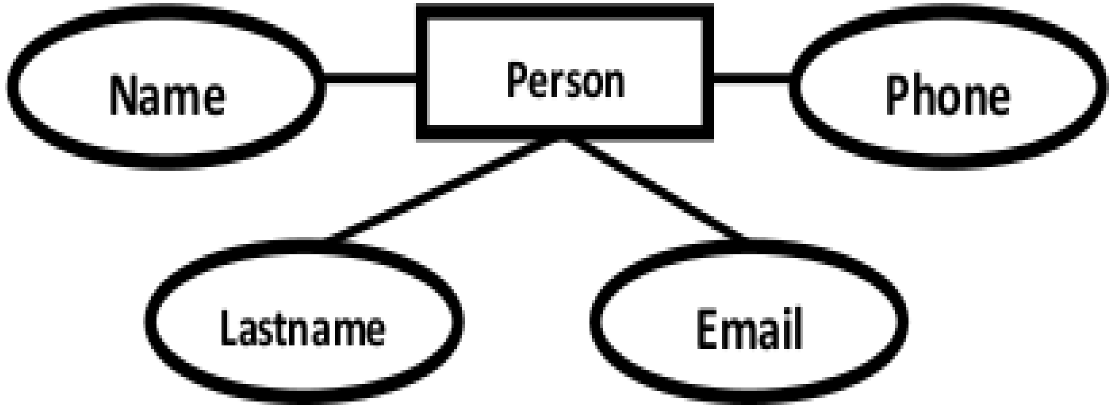
continuation

ER Model, when conceptualized into diagrams, gives a good overview of entity-relationship, which is easier to understand. ER diagrams can be mapped to relational schema, that is, it is possible to create relational schema using ER diagram.

We cannot import all the ER constraints into relational model, but an approximate schema can be generated.

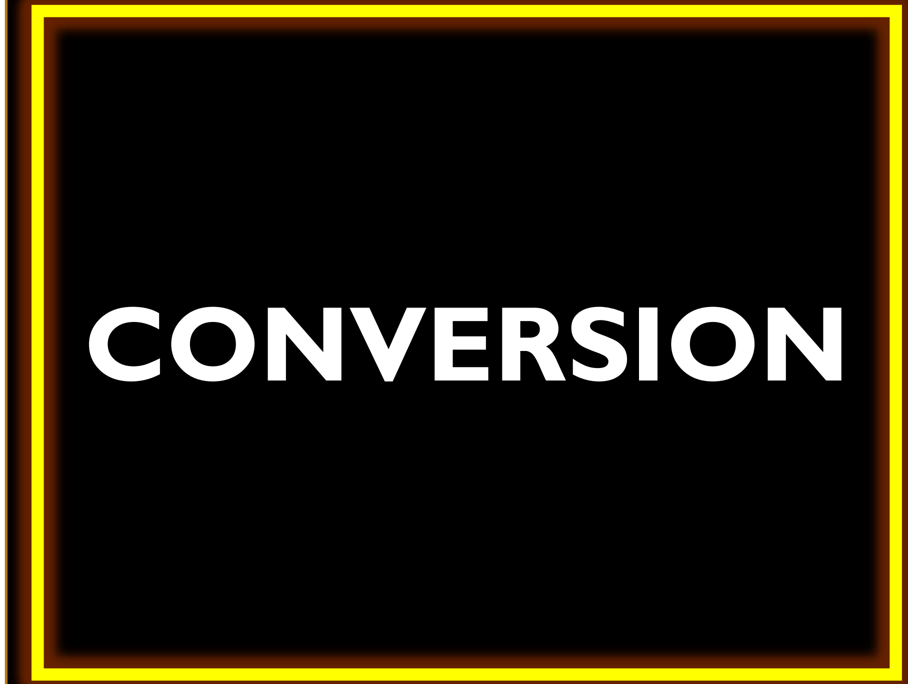
CONVERSION

Mapping #1:
Entities and Simple Attributes

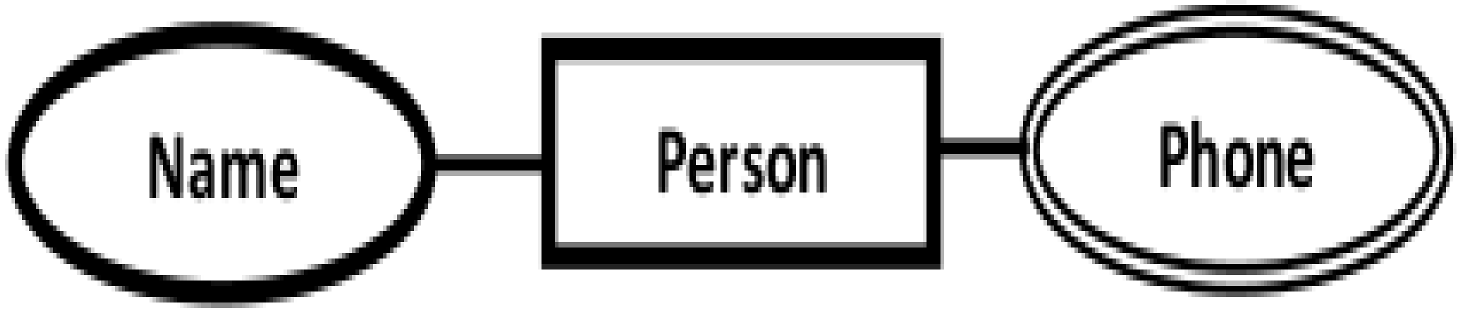


Relational Schema:

Person (***<u>personid</u>***, name, lastname, email, phone)



Mapping #2:
Multi-Valued Attributes



Relational Schema:

Person (**<u>personid</u>**, name)

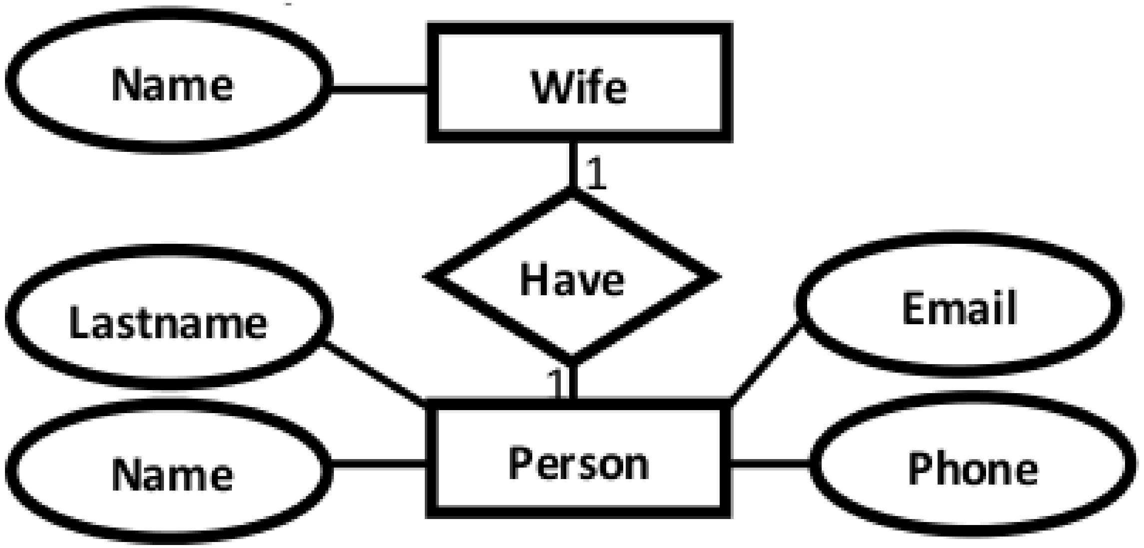
Phone (**<u>phoneid</u>**, personid, phone)

CONVERSION

If you have a multi-valued attribute, take the attribute and turn it into a new entity or table of its own. Then make a 1:N relationship between the new entity and the existing one. In simple words. 1. Create a table for the attribute. 2. Add the primary (id) column of the parent entity as a foreign key within the new table as shown below:

CONVERSION

Mapping #3:
1:1 Relationship



Relational Schema:

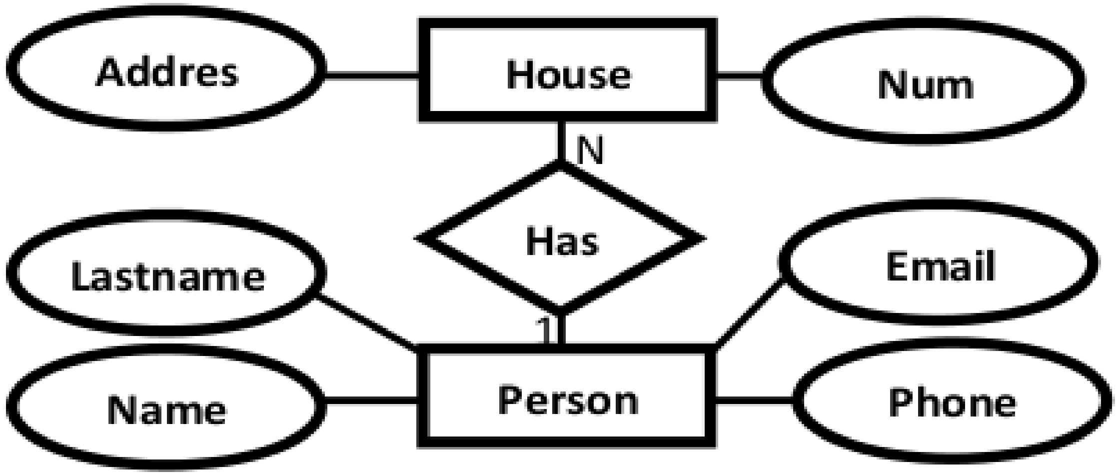
Persons (<u>**personid**</u> , name, lastname, email, phone, *wifeid*)
Wife (<u>**wifeid**</u> , name)

Or

Persons (<u>**personid**</u> , name, lastname, email, phone)
Wife (<u>**wifeid**</u> , name, *personid*)

CONVERSION

Mapping #4:
1:N Relationship



Relational Schema:

Person (**<u>personid</u>**, name, lastname, email, phone)

House (**<u>house_id</u>**, num, address, personid)

Note: Many side contains the foreign key.

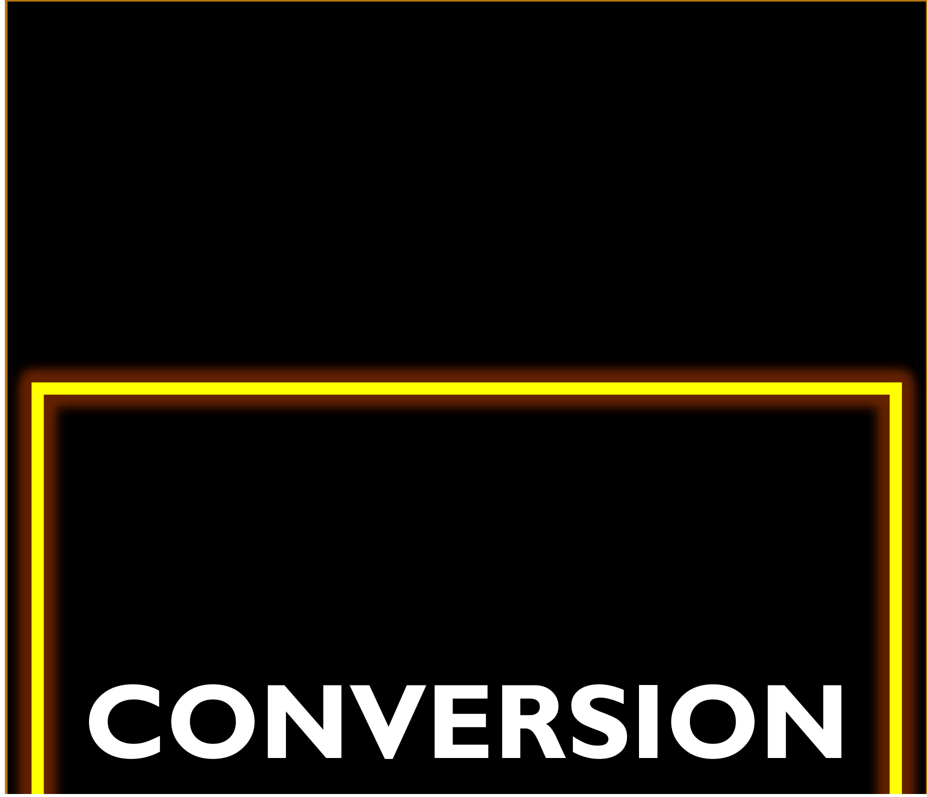
CONVERSION

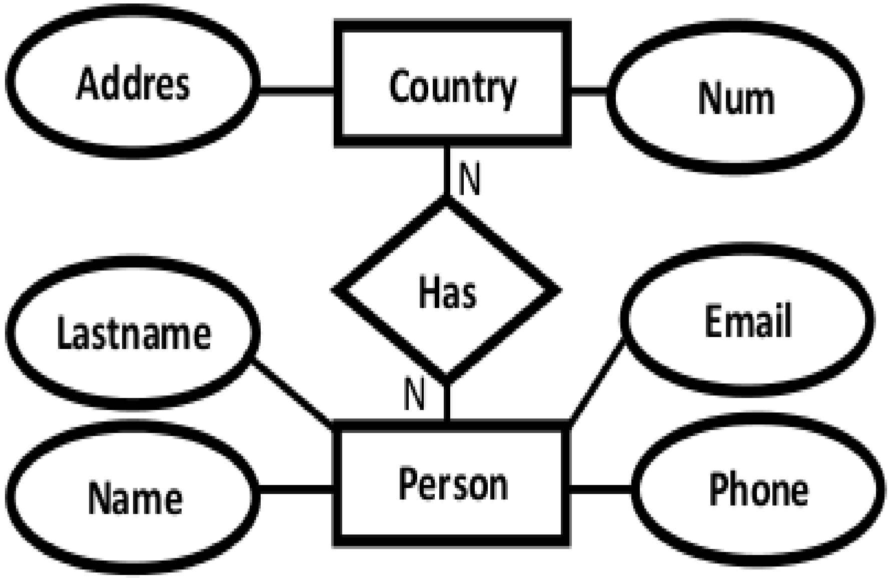
Mapping #5:
N:N Relationship

We normally use tables to express such type of relationship. This is the same for N−ary relationship of ER diagrams. For instance, The Person can live or work in many countries. Also, a country can have many people. To express this relationship within a relational schema we use a separate table as shown below:

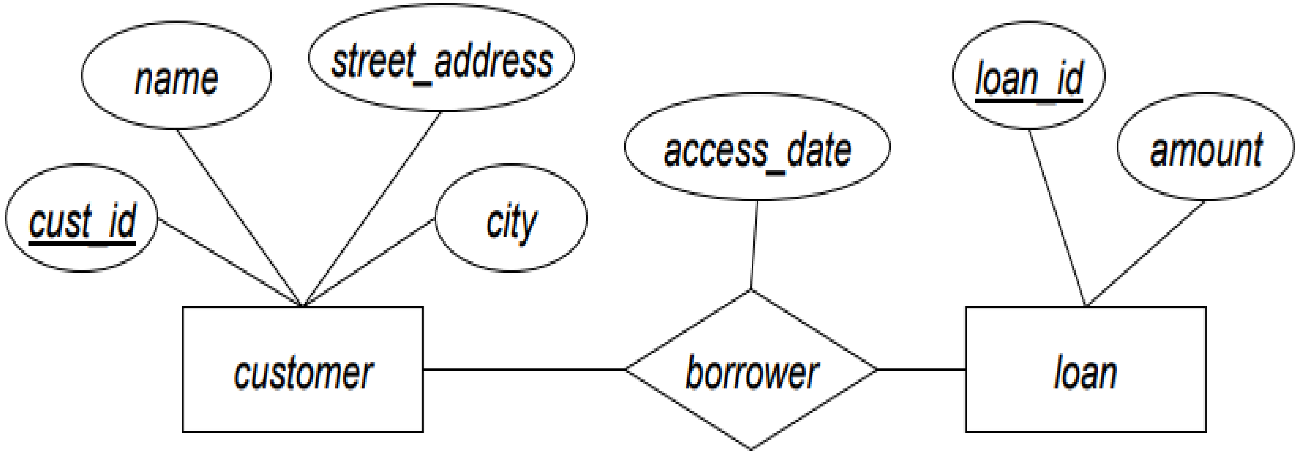
Relational Schema:

Person (**personid**, name, lastname, email, phone)
Country (**country_id**, address, num)
HasRela (**rela_id**, personid, country_id)





Mapping #5:
N:N Relationship



Relational Schema:

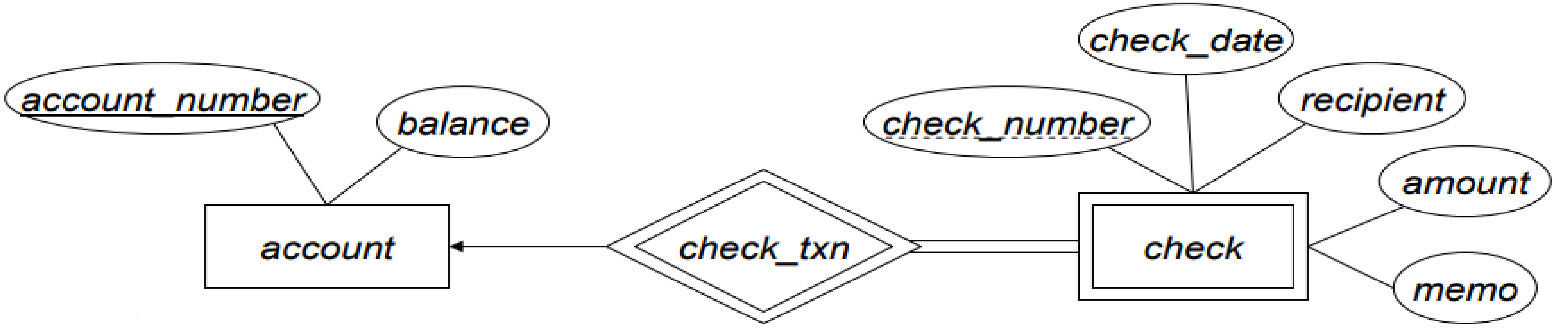
Customer (**<u>cust_id</u>**, name, street_address, city)

Loan (**<u>loan_id</u>**, amount)

Borrower (**<u>cust_id</u>**, **<u>loan_id</u>**, access_date)

CONVERSION

Mapping #6: Relationship including weak entity

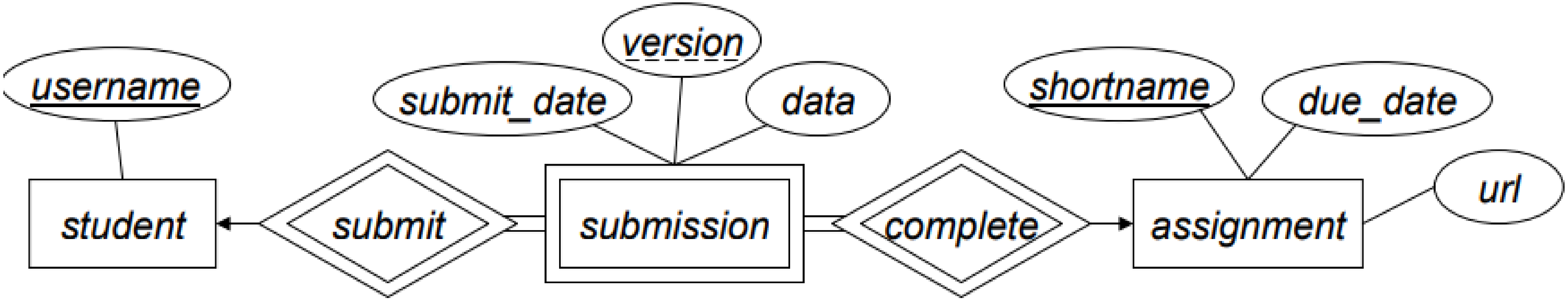


Relational Schema:

Account (<u>**account_number**</u>, balance)

Check (<u>**account_number, check_number**</u>, check_date, receipient, amount, memo)

Mapping #6: Relationship including weak entity



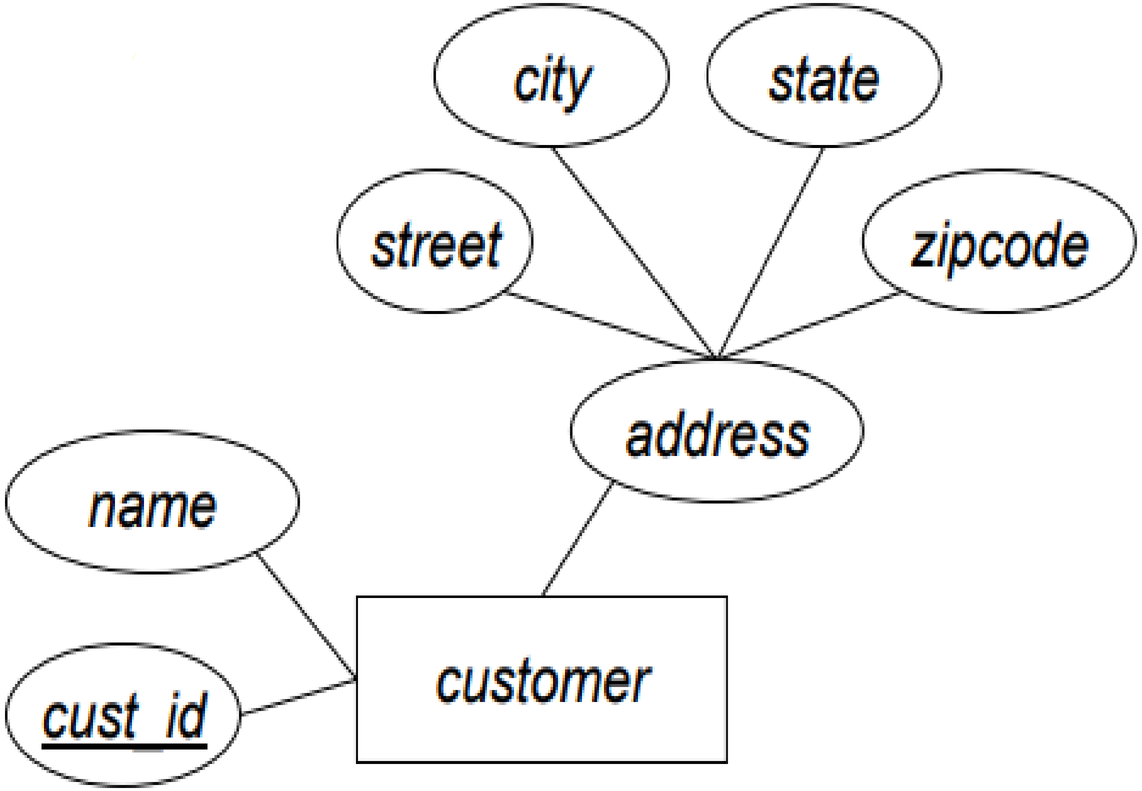
Relational Schema:

Student (**username**)

Assignment (**shortname**, due_date, url)

Submission (**username, shortname, version**, submit_date, date)

Mapping #7:
Composite Attributes



Relational Schema:

Customer (**<u>cust_id</u>**, name, street, city, state, zipcode)

Data Normalization

- Primarily a tool to validate and improve a logical design so that it satisfies certain constraints that ***avoid unnecessary duplication of data***
- The process of decomposing relations with anomalies to produce smaller, ***well-structured*** relations

DATA Normalization

Well-Structured Relations

A relation that contains minimal data redundancy and allows users to insert, delete, and update rows without causing data inconsistencies

Goal is to avoid anomalies

- **Insertion Anomaly** – adding new rows forces user to create duplicate data
- **Deletion Anomaly** – deleting rows may cause a loss of data that would be needed for other future rows
- **Modification Anomaly** – changing data in a row forces changes to other rows because of duplication

General rule of thumb: a table should not pertain to more than one entity type

Example – Figure 5.2b

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	SPSS	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

Is this a relation?

Answer: YES, unique rows and no multivalued attributes

What's the primary key?

Answer: Composite: Emp_ID, Course_Title

ATA

alization

Anomalies in this table

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	SPSS	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

ATA
alization

- **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

Anomalies in this table

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	SPSS	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

Why do these anomalies exist?

Because there are two themes (entity types) into one relation. This results in duplication, and an unnecessary dependency between the entities

ATA

alization

Functional Dependencies and Keys

- Functional Dependency: The value of one attribute (the ***determinant***) determines the value of another attribute
- 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 field is functionally dependent on every candidate key

DATA Normalization

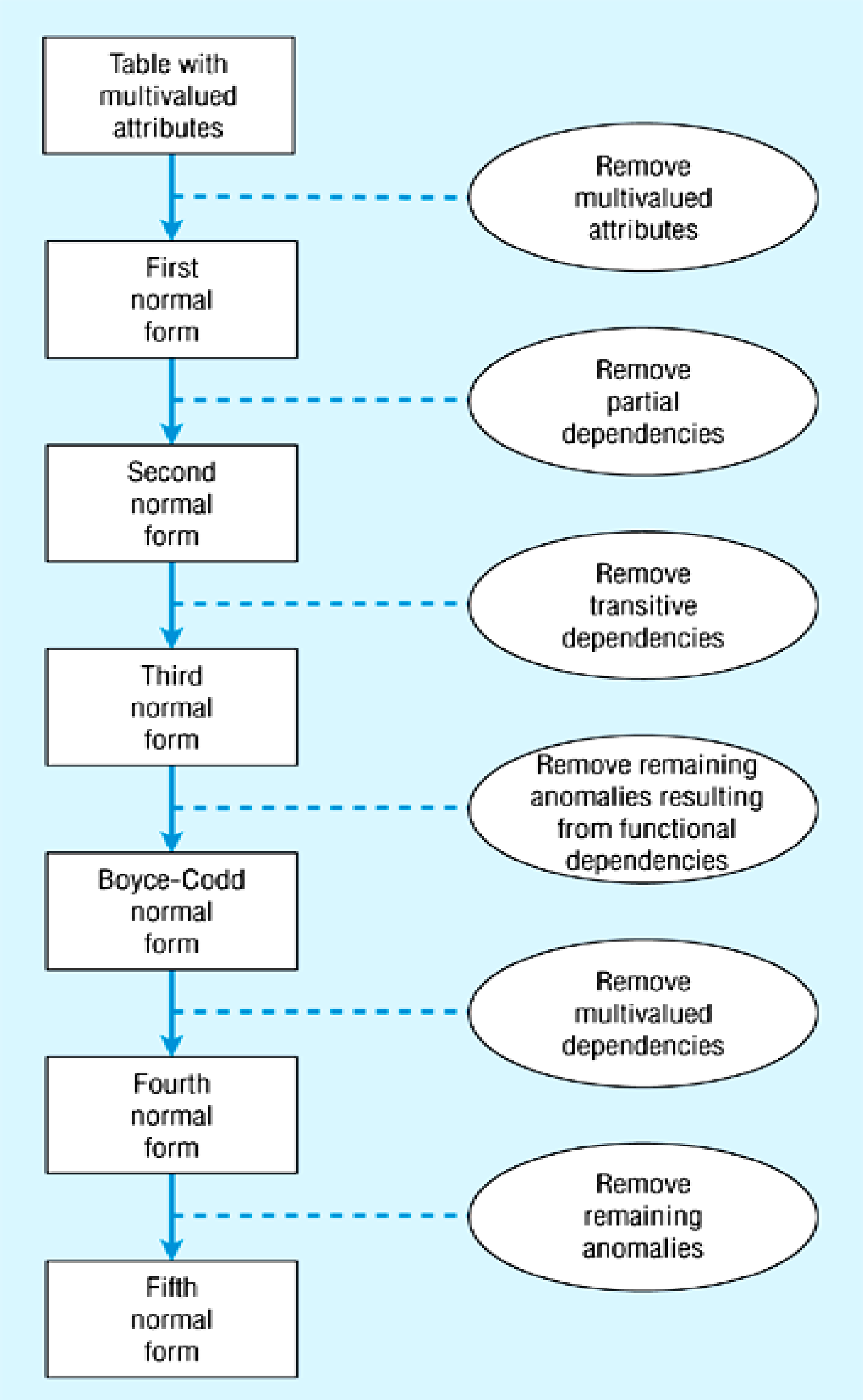


Figure 5.22 -Steps in normalization

DATA Normalization

First Normal Form

- No multivalued attributes
- Every attribute value is atomic
- Fig. 5-25 *is not* in 1st Normal Form (multivalued attributes), it is not a relation
- Fig. 5-26 *is* in 1st Normal form
- *All relations* are in 1st Normal Form

1st

DATA Normalization

First Normal Form

Table with multivalued attributes, not in 1st normal form

Figure 5-25 INVOICE data (Pine Valley Furniture Company)

Order_ID	Order_Date	Customer_ID	Customer_Name	Customer_Address	Product_ID	Product_Description	Product_Finish	Unit_Price	Ordered_Quantity
1006	10/24/2004	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	10/25/2004	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

First Normal Form

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

Figure 5-26 INVOICE relation (1NF) (Pine Valley Furniture Company)

Order_ID	Order_Date	Customer_ID	Customer_Name	Customer_Address	Product_ID	Product_Description	Product_Finish	Unit_Price	Ordered_Quantity
1006	10/24/2004	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
1006	10/24/2004	2	Value Furniture	Plano, TX	5	Writer's Desk	Cherry	325.00	2
1006	10/24/2004	2	Value Furniture	Plano, TX	4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2004	6	Furniture Gallery	Boulder, CO	11	4–Dr Dresser	Oak	500.00	4
1007	10/25/2004	6	Furniture Gallery	Boulder, CO	4	Entertainment Center	Natural Maple	650.00	3

Note: this is relation, but not a well-structured one

First Normal Form

Anomalies in this table

- **Insertion** – if new product is ordered for order 1007 of existing customer, customer data must be re-entered, causing duplication
- **Deletion** – if we delete the Dining Table from Order 1006, we lose information concerning this item's finish and price
- **Update** – changing the price of product ID 4 requires update in several records

Why do these anomalies exist?

Because there are multiple themes (entity types) into one relation. This results in duplication, and an unnecessary dependency between the entities

1st

DATA Normalization

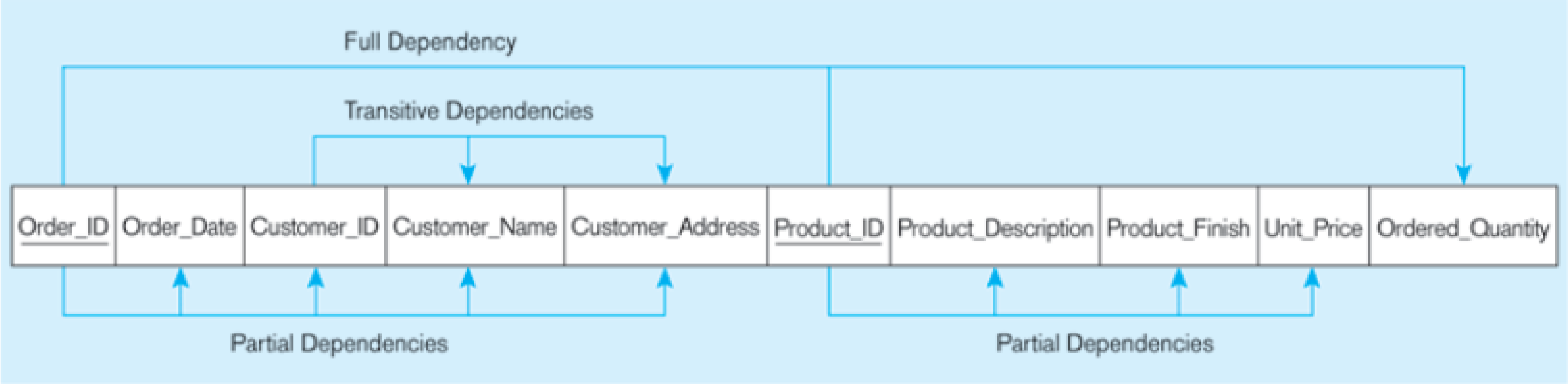
Second Normal Form

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

- Every non-key attribute must be defined by the entire key, not by only part of the key
- No partial functional dependencies

Second Normal Form

Figure 5-27 Functional dependency diagram for INVOICE



Order_ID → Order_Date, Customer_ID, Customer_Name, Customer_Address

Customer_ID → Customer_Name, Customer_Address

Product_ID → Product_Description, Product_Finish, Unit_Price

Order_ID, Product_ID → Order_Quantity

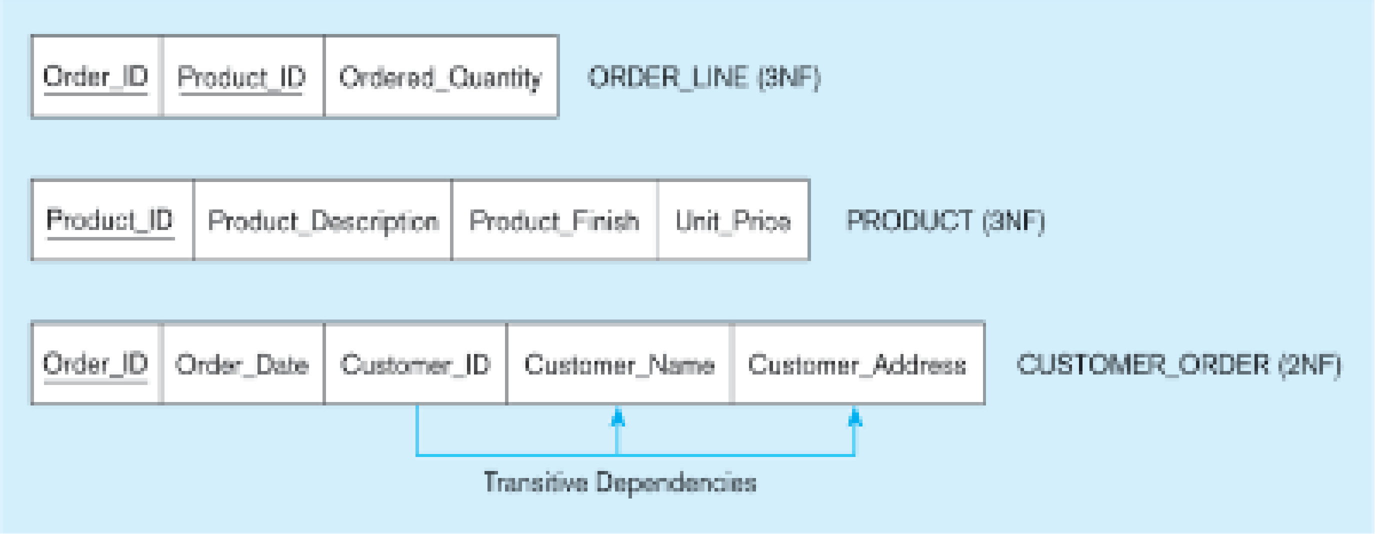
Therefore, NOT in 2nd Normal Form

tion

Second Normal Form

Getting it into Second Normal Form

Figure 5-28 Removing partial dependencies



Partial Dependencies are removed, but there are still transitive dependencies

Third Normal Form

- **2NF + *no transitive dependencies* (functional dependencies on non-primary-key attributes)**
- Note: this is called **transitive**, because *the primary key is a determinant for another attribute, which in turn is a determinant for a third*
- **Solution**: non-key determinant with transitive dependencies go into a **new table**; non-key determinant becomes primary key in the new table and stays as foreign key in the old table

Third Normal Form

Getting it into 3rd Normal Form

Figure 5-29 Removing transitive dependencies

Order_ID	Order_Date	Customer_ID	ORDER (3NF)

Customer_ID	Customer_Name	Customer_Address	CUSTOMER (3NF)

Transitive dependencies are removed

Merging Relations

- **View Integration – Combining entities from multiple ER models into common relations**
- **Issues to watch out for when merging entities from different ER models:**
  - **Synonyms** – two or more attributes with different names but same meaning
  - **Homonyms** – attributes with same name but different meanings
  - **Transitive dependencies** – even if relations are in 3NF prior to merging, they may not be after merging
  - **Supertype/subtype relationships** – may be hidden prior to merging

DATA Normalization

Enterprise Keys

- Primary keys that are unique in the whole database, not just within a single relation

- Corresponds with the concept of an object ID in object-oriented systems

DATA Normalization

Figure 5-31a Enterprise key - Relations with enterprise key

OBJECT (OID, Object_Type)
EMPLOYEE (OID, Emp_ID, Emp_Name, Dept_Name, Salary)
CUSTOMER (OID, Cust_ID, Cust_Name, Address)

Figure 5-31b Enterprise key - Sample data with enterprise key

OBJECT

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

EMPLOYEE

OID	Emp_ID	Emp_Name	Dept_Name	Salary
1	100	Jennings, Fred	Marketing	50000
4	101	Hopkins, Dan	Purchasing	45000
5	102	Huber, Ike	Accounting	45000

CUSTOMER

OID	Cust_ID	Cust_Name	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

REFERENCES

- https://www.tutorialride.com/dbms/enhanced-entity-relationship-model-eer-model.htm
- Elmasri, Ramez & Navathe, Shamkant (2016). Fundamentals of Database Systems 7th ed., Pearson.

END
OF TODAY'S
LESSON