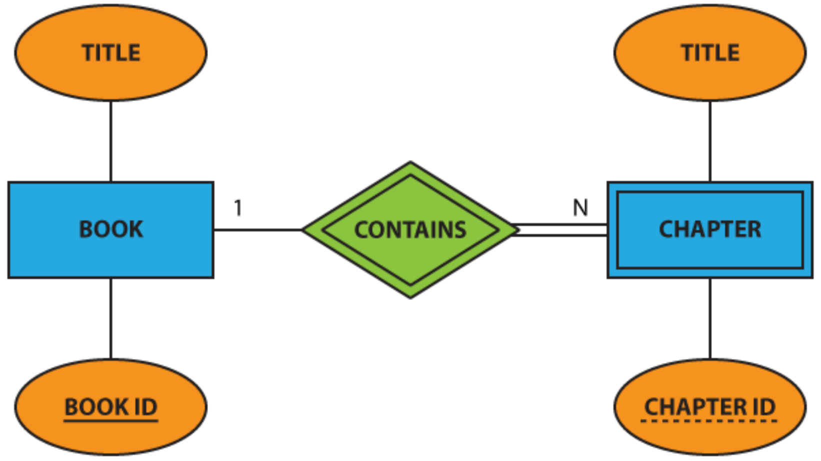
Data Modeling Using the Entity-Relationship (ER) Model



Modified from Chapter 3 and 4 by Ramez Elmasri and Shamkant B. Navathe, Fundamentals of Database Systems,

Seventh Edition,

Pearson Education, 2017

Chapter Outline

- Example Database Application (COMPANY)
- ER Model Concepts (traditional approach): high-level conceptual data model
  - Entities and Attributes
  - Entity Types, Value Sets, and Key Attributes
  - Relationships and Relationship Types
  - Weak Entity Types
  - Roles and Attributes in Relationship Types
- ER Diagrams - Notation
- ER Diagram for COMPANY Schema
- Alternative Notations – UML class diagrams, others

The description of a database is called the database schema, which is specified during database design and is not expected to change frequently.

Database application

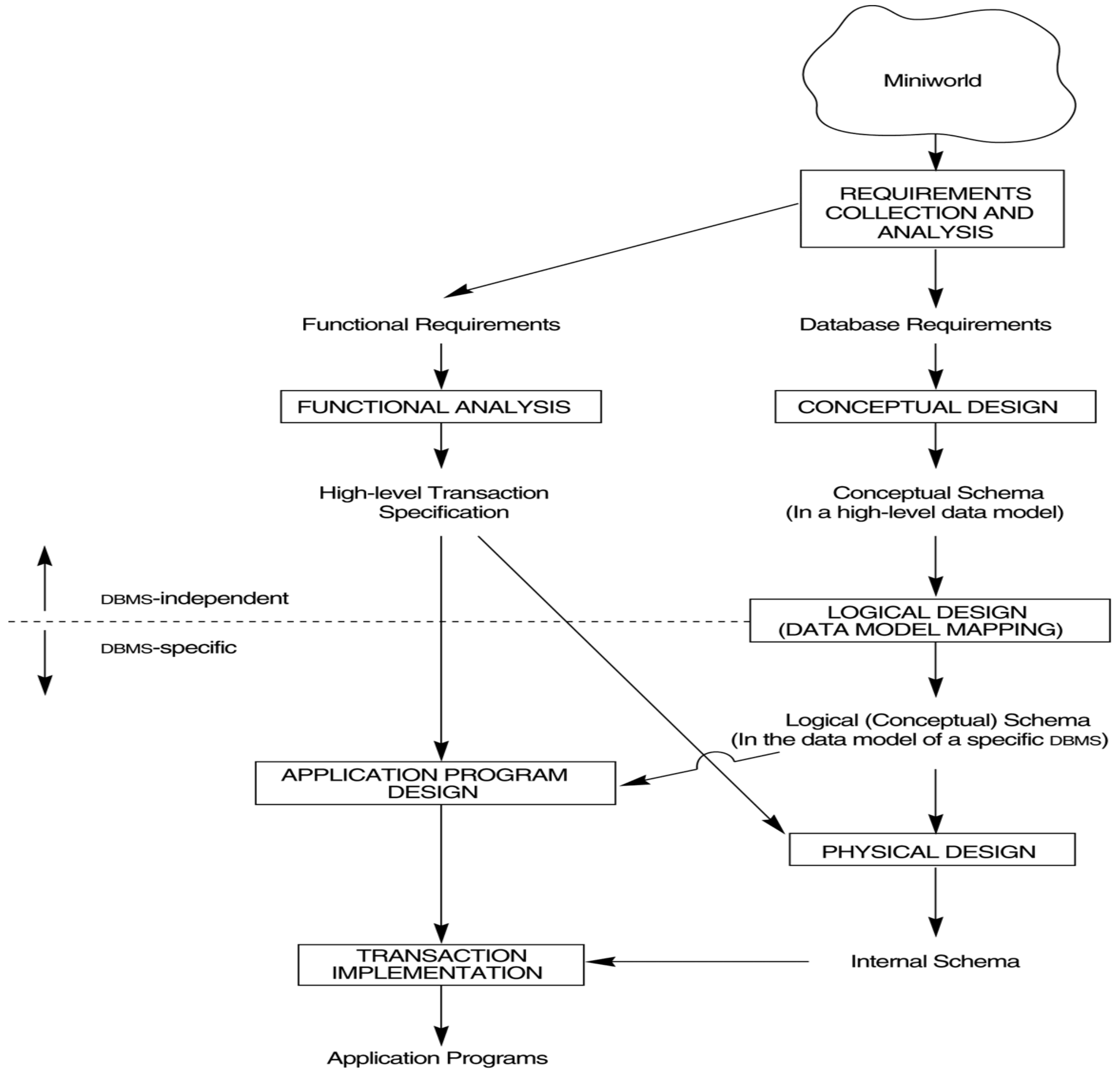
Database application refers to a particular database and the associated programs that implement the database queries and updates

Ex. a BANK application that keeps track of customer accounts would include programs that keeps track customer accounts for making deposits and withdrawals

Database design methodologies & Software engineering methodologies

- **Database design methodologies** include more concepts for specifying operations on database objects
- Software engineering methodologies specify in more detail the structure of the databases that software programs will use and access
- Both are strongly related

FIGURE 3.1
A simplified diagram to illustrate the main phases of database design.


Miniworld
REQUIREMENTS COLLECTION AND ANALYSIS
Functional Requirements
Database Requirements
FUNCTIONAL ANALYSIS
CONCEPTUAL DESIGN
High-level Transaction Specification
Conceptual Schema (In a high-level data model)
DBMS-independent
DBMS-specific
LOGICAL DESIGN (DATA MODEL MAPPING)
Logical (Conceptual) Schema (In the data model of a specific DBMS)
APPLICATION PROGRAM DESIGN
PHYSICAL DESIGN
TRANSACTION IMPLEMENTATION
Internal Schema
Application Programs

Three Schema Architecture

External level

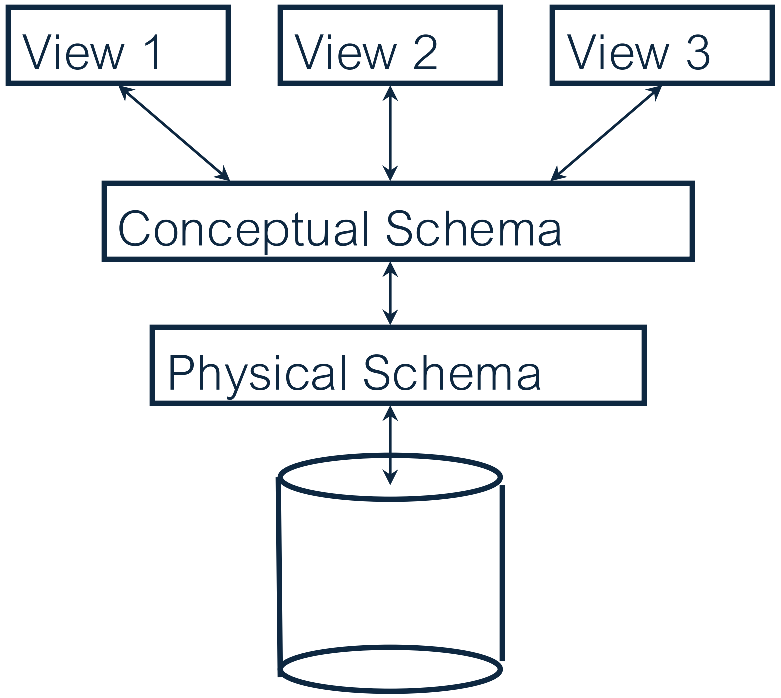
- Views describe how users see the data

Conceptual level

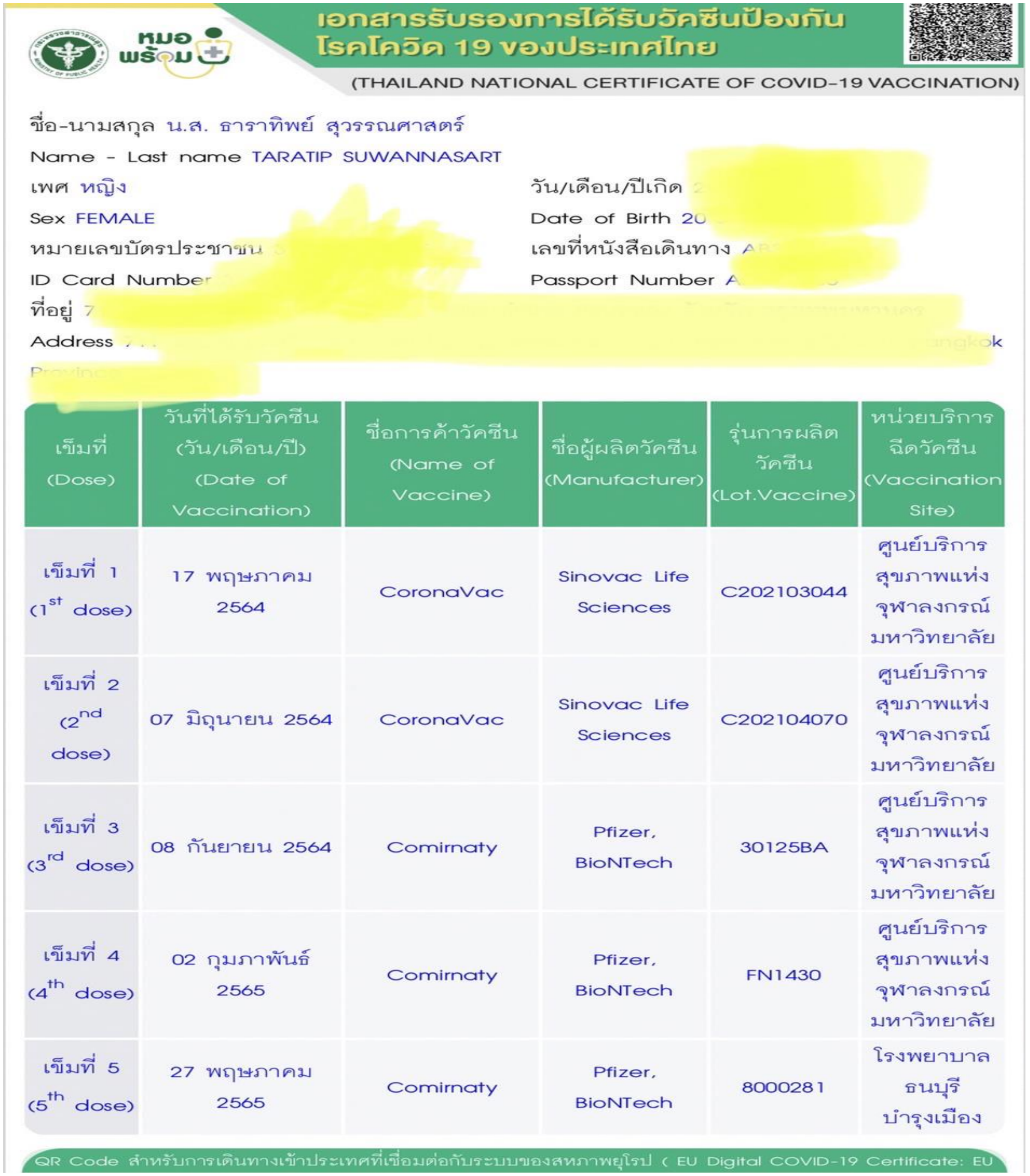
- Conceptual schema defines logical structure

Internal level

- Physical schema describes the files and indexes used.



 Schemas are defined using DDL; data is modified/queried using DML.

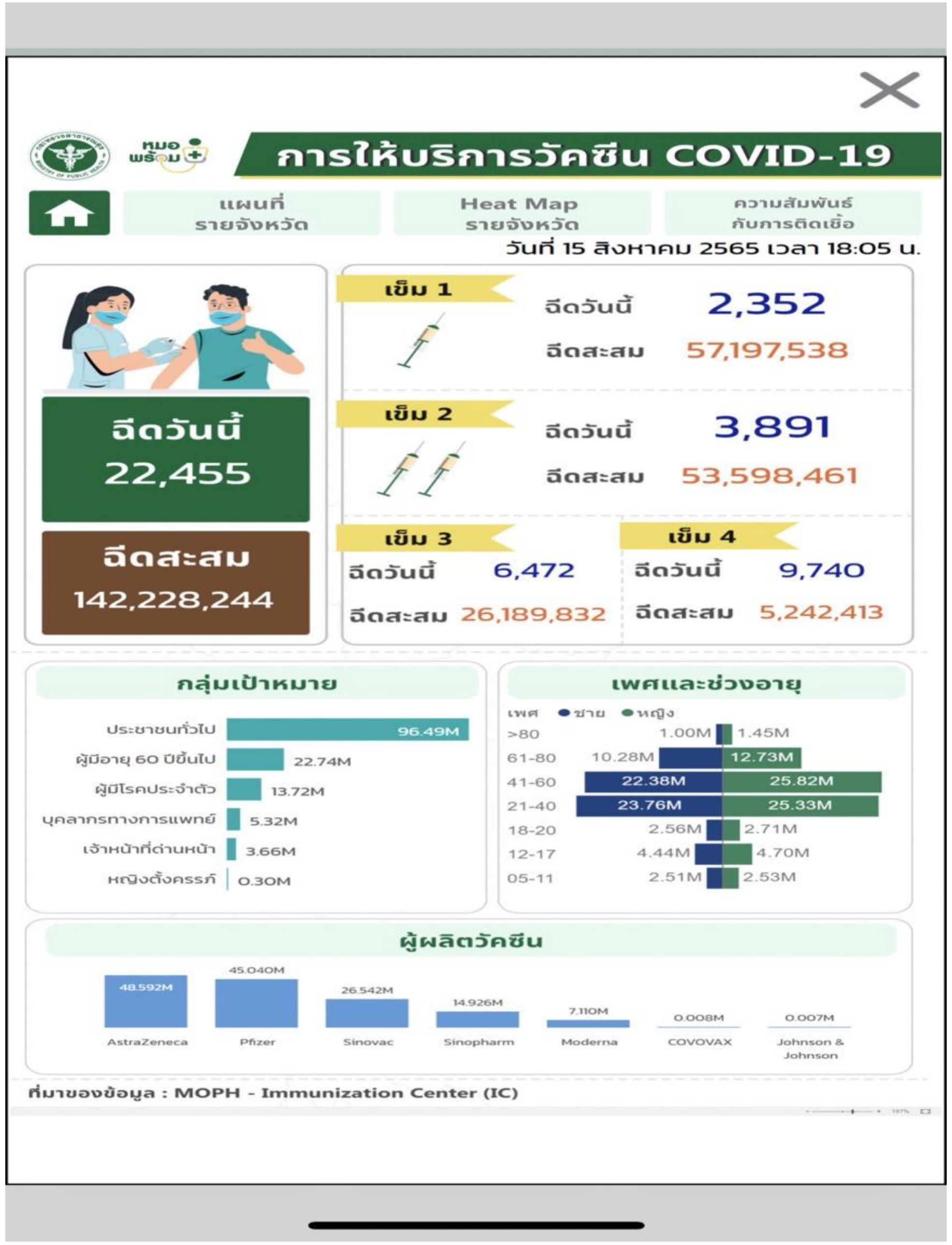


เอกสารรับรองการได้รับวัคซีนป้องกันโรคโควิด 19 ของประเทศไทย
(THAILAND NATIONAL CERTIFICATE OF COVID-19 VACCINATION)

ชื่อ-นามสกุล น.ส. ธาราทิพย์ สุวรรณศาสตร์
Name - Last name TARATIP SUWANNASART
เพศ หญิง วัน/เดือน/ปีเกิด [redacted]
Sex FEMALE Date of Birth [redacted]
หมายเลขบัตรประชาชน [redacted] เลขที่หนังสือเดินทาง [redacted]
ID Card Number [redacted] Passport Number [redacted]
ที่อยู่ [redacted]
Address [redacted]

เข็มที่ (Dose)	วันที่ได้รับวัคซีน (วัน/เดือน/ปี) (Date of Vaccination)	ชื่อการค้าวัคซีน (Name of Vaccine)	ชื่อผู้ผลิตวัคซีน (Manufacturer)	รุ่นการผลิตวัคซีน (Lot.Vaccine)	หน่วยบริการฉีดวัคซีน (Vaccination Site)
เข็มที่ 1 (1st dose)	17 พฤษภาคม 2564	CoronaVac	Sinovac Life Sciences	C202103044	ศูนย์บริการสุขภาพแห่งจุฬาลงกรณ์มหาวิทยาลัย
เข็มที่ 2 (2nd dose)	07 มิถุนายน 2564	CoronaVac	Sinovac Life Sciences	C202104070	ศูนย์บริการสุขภาพแห่งจุฬาลงกรณ์มหาวิทยาลัย
เข็มที่ 3 (3rd dose)	08 กันยายน 2564	Comirnaty	Pfizer, BioNTech	30125BA	ศูนย์บริการสุขภาพแห่งจุฬาลงกรณ์มหาวิทยาลัย
เข็มที่ 4 (4th dose)	02 กุมภาพันธ์ 2565	Comirnaty	Pfizer, BioNTech	FN1430	ศูนย์บริการสุขภาพแห่งจุฬาลงกรณ์มหาวิทยาลัย
เข็มที่ 5 (5th dose)	27 พฤษภาคม 2565	Comirnaty	Pfizer, BioNTech	8000281	โรงพยาบาลธนบุรี บำรุงเมือง

QR Code สำหรับการเดินทางเข้าประเทศที่เชื่อมต่อกับระบบของสหภาพยุโรป (EU Digital COVID-19 Certificate: EU



การให้บริการวัคซีน COVID-19

วันที่ 15 สิงหาคม 2565 เวลา 18:05 น.

ฉีดวันนี้ 22,455
ฉีดสะสม 142,228,244

	ฉีดวันนี้	ฉีดสะสม
เข็ม 1	2,352	57,197,538
เข็ม 2	3,891	53,598,461
เข็ม 3	6,472	26,189,832
เข็ม 4	9,740	5,242,413

กลุ่มเป้าหมาย

ประชาชนทั่วไป	96.49M
ผู้มีอายุ 60 ปีขึ้นไป	22.74M
ผู้มีโรคประจำตัว	13.72M
บุคลากรทางการแพทย์	5.32M
เจ้าหน้าที่ด่านหน้า	3.66M
หญิงตั้งครรภ์	0.30M

เพศและช่วงอายุ

เพศ	● ชาย	● หญิง
>80	1.00M	1.45M
61-80	10.28M	12.73M
41-60	22.38M	25.82M
21-40	23.76M	25.33M
18-20	2.56M	2.71M
12-17	4.44M	4.70M
05-11	2.51M	2.53M

ผู้ผลิตวัคซีน

AstraZeneca	Pfizer	Sinovac	Sinopharm	Moderna	COVOVAX	Johnson & Johnson
48.592M	45.040M	26.542M	14.926M	7.110M	0.008M	0.007M

ที่มาของข้อมูล : MOPH - Immunization Center (IC)

Data Independence

Logical data independence: Protection from changes in *logical* structure of data.

Physical data independence: Protection from changes in *physical* structure of data.

 One of the most important benefits of using a DBMS!

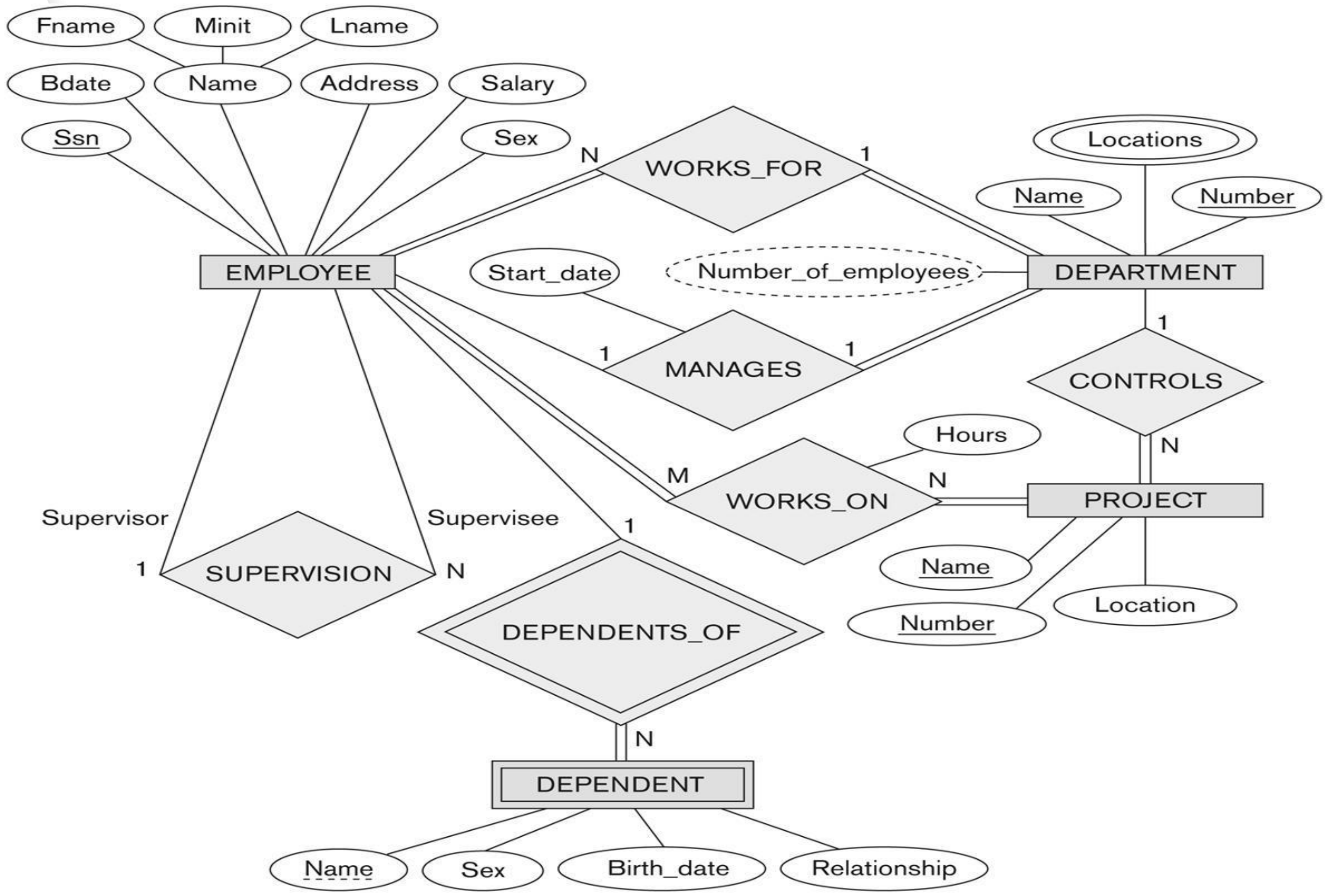
Example COMPANY Database

- Requirements of the Company (oversimplified for illustrative purposes)
  - The company is organized into **DEPARTMENTs.**
  - Each department has a unique name, unique number and a particular employee who *manages* the department.
  - We keep track of the start date of the department manager. A department may have several locations
  - Each department *controls* a number of **PROJECTs.**
  - Each project has a unique name, unique number and is located at a single location.

Example COMPANY Database (Cont.)

- We store each **EMPLOYEE**’s social security number, address, salary, sex, and birthdate.
- Each employee *works for* one department but may *work on* several projects.
- We keep track of the number of hours per week that an employee currently works on each project.
- We also keep track of the *direct supervisor* of each employee.
- Each employee may *have* a number of **DEPENDENTs**.
- For each dependent, we keep track of their name, sex, birthdate, and relationship to employee.

FIGURE 3.2
An ER schema diagram for the COMPANY database.



ER Model Concepts

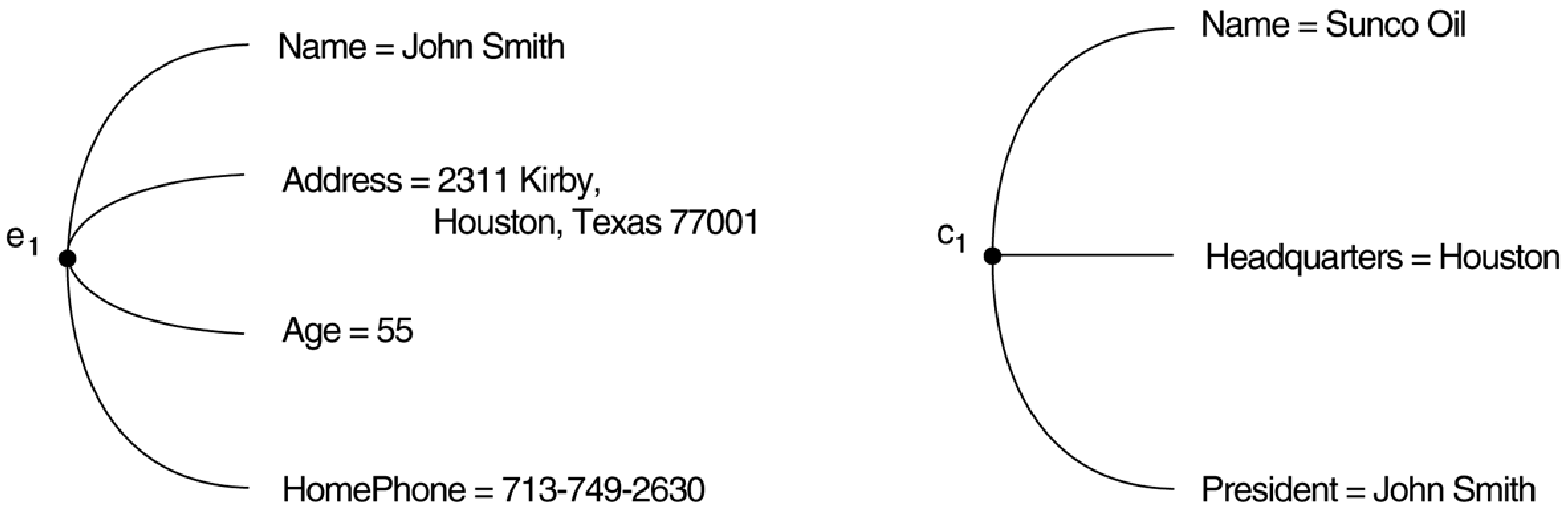
- **Entities are specific objects or things in the mini-world that are represented in the database.**
  - For example the EMPLOYEE John Smith, the Research DEPARTMENT, the ProductX PROJECT

ER Model Concepts (cont.)

- **Attributes are properties used to describe an entity.**
  - For example an EMPLOYEE entity may have a Name, SSN, Address, Sex, BirthDate
  - A specific entity will have a value for each of its attributes.
    - For example a specific employee entity may have Name='John Smith', SSN='123456789', Address ='731, Fondren, Houston, TX', Sex='M', BirthDate='09-JAN-55'
  - Each attribute has a *value set* (or data type) associated with it
    - e.g. integer, string, subrange, enumerated type, …

FIGURE 3.3

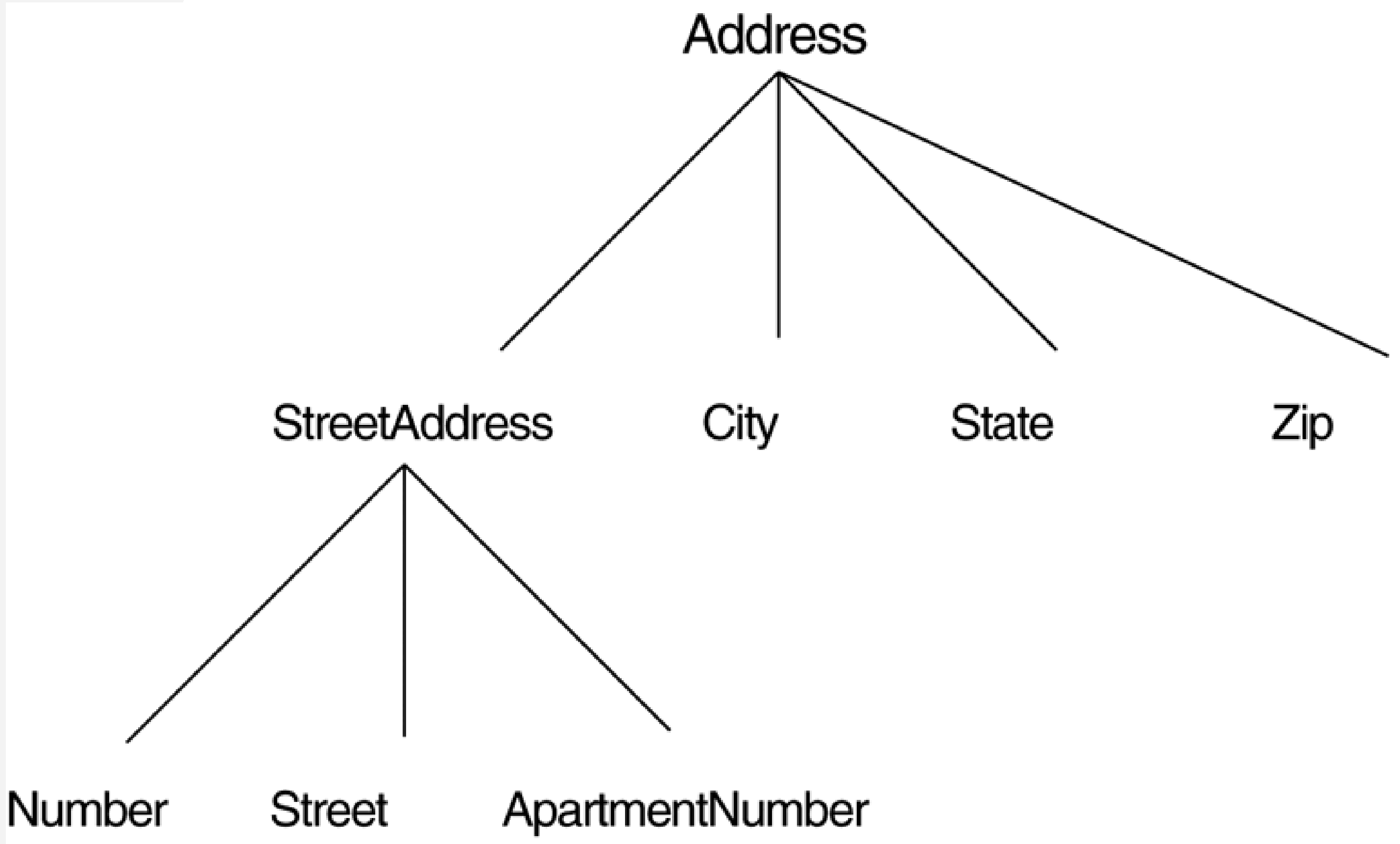
Two entities, employee e_1 and company c_1, and their attributes.



Types of Attributes (1)

- **Simple**
  - Each entity has a single atomic value for the attribute.
    - ➢ For example, SSN or Sex.
- **Composite**
  - The attribute may be composed of several components.
    - ➢ For example, Address (Apt#, House#, Street, City, State, ZipCode, Country) or Name (FirstName, MiddleName, LastName). **Single-valued** – age, height
- **Multi-valued**
  - An entity may have multiple values for that attribute.
    - ➢ For example, Color of a CAR or PreviousDegrees of a STUDENT. Denoted as {Color} or {PreviousDegrees}.

FIGURE 3.4 A hierarchy of composite attributes.

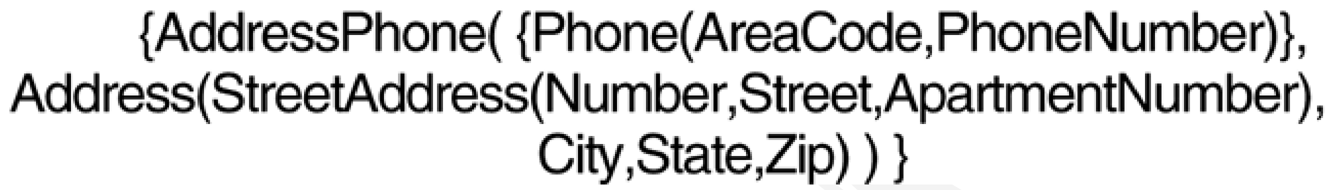


Composition may form a hierarchy where some components are themselves composite.

Types of Attributes (2)

- In general, composite and multi-valued attributes may be nested arbitrarily to any number of levels although this is rare.
    - For example, PreviousDegrees of a STUDENT is a composite multi-valued attribute denoted by **{PreviousDegrees (College, Year, Degree, Field)}.**

FIGURE 3.5
A complex attribute: AddressPhone.


{AddressPhone({Phone(AreaCode,PhoneNumber)},
Address(StreetAddress(Number,Street,ApartmentNumber),
City,State,Zip)) }

Types of Attributes (3)

Stored attributes : birth date

Derived attributes : age

Null values : a particular attribute may not have applicable value

- Example : an apartment number attribute for a single-family home, College degree
- Null can be used if we do not know the value of an attribute for a particular entity
  - the attribute value exists but missing: height
  - the attribute value is not known whether the attribute value exist: home phone

Entity Types and Key Attributes (1)

Entities with the same basic attributes are grouped or typed into **an entity type.**

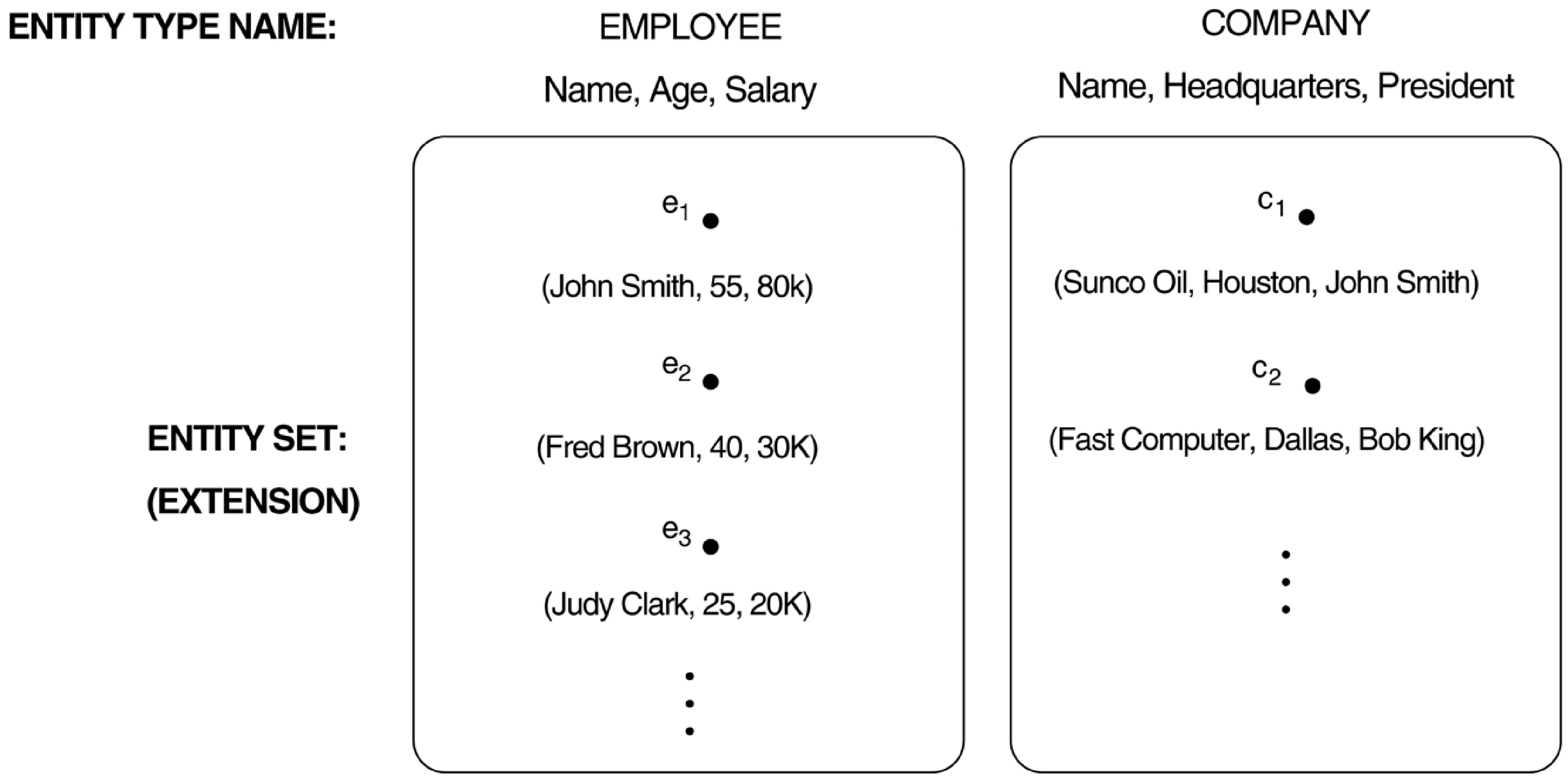
- **For example, the EMPLOYEE entity type or the PROJECT entity type.**

The collection of all entities of a particular entity type in a database at any point in time is called **an entity set**

An attribute of an entity type for which each entity must have a unique value is called **a key attribute** (uniqueness constraint) of the entity type.

- **For example, SSN of EMPLOYEE.**

FIGURE 3.6
Two entity types, EMPLOYEE and COMPANY, and some member entities of each.



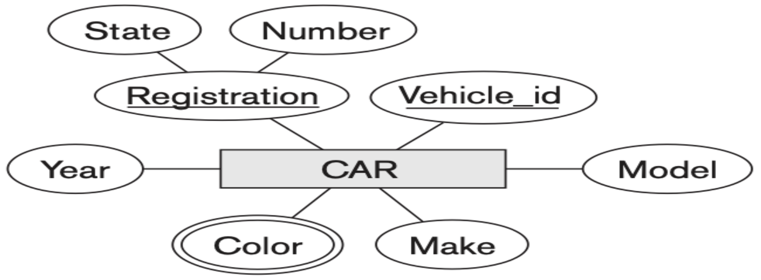
Entity Types and Key Attributes (2)

- An entity type may have more than one key. For example, the CAR entity type may have two keys:
  - **Vehicle_id** (popularly called VIN) and
  - **Registration** (Number, State), also known as license_plate number.
- A key attribute may be composite. For example, ***Registration*** is a key of the CAR entity type with components *(Number, State)*.

FIGURE 3.7

The CAR entity type with two key attributes, Registration and VehicleID.

(a)



(b)

CAR
Registration (Number, State), Vehicle_id, Make, Model, Year, {Color}

CAR$_1$
((ABC 123, TEXAS), TK629, Ford Mustang, convertible, 2004 {red, black})

CAR$_2$
((ABC 123, NEW YORK), WP9872, Nissan Maxima, 4-door, 2005, {blue})

CAR$_3$
((VSY 720, TEXAS), TD729, Chrysler LeBaron, 4-door, 2002, {white, blue})

⋮

Value sets (domains) of attribute

- Each simple attribute of an entity type is associated with a value set (or domain of values) = basic data types available in most programming languages such as
    - Integer, String, Boolean, Float, Enumerated type, Subrage etc.

FIGURE 3.8
Preliminary design of entity types for the COMPANY database

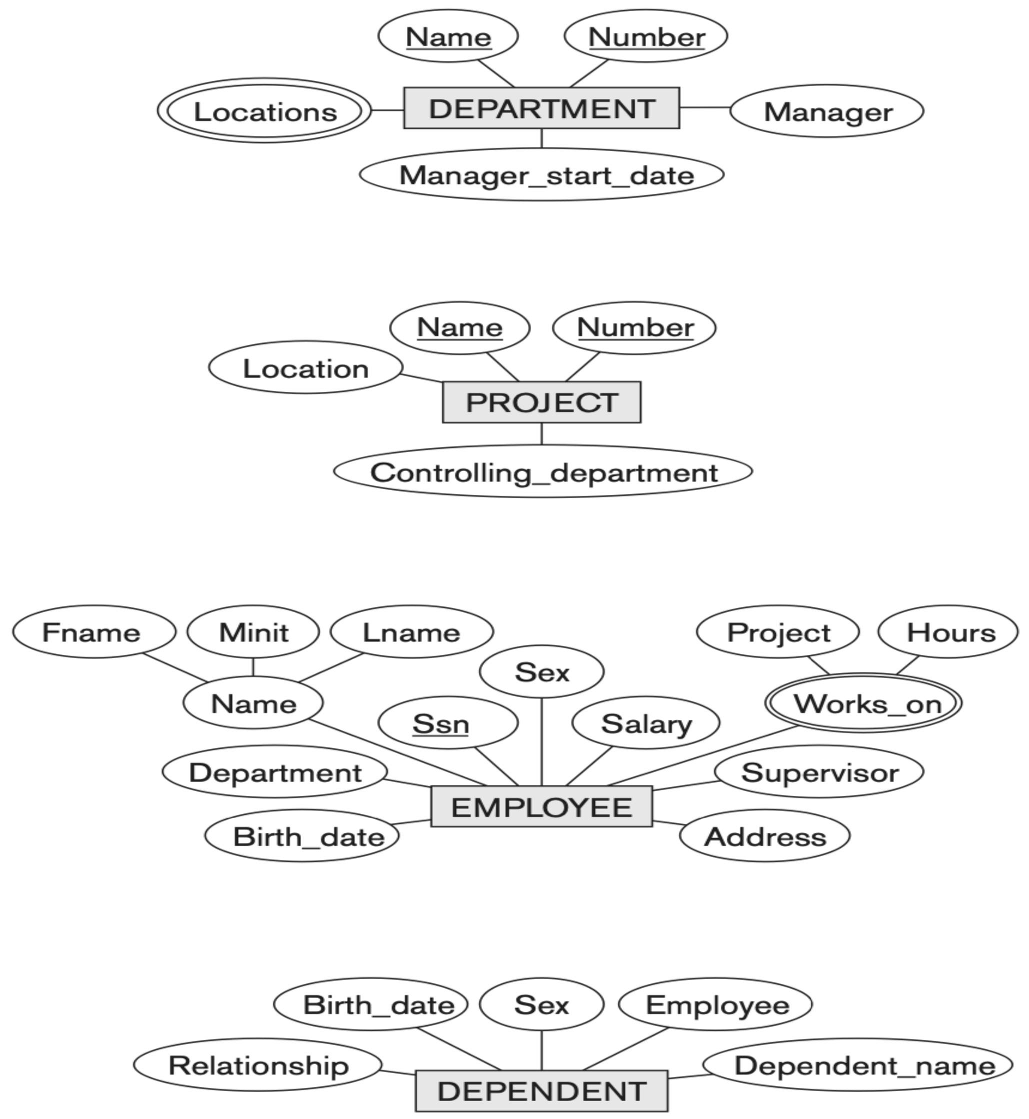
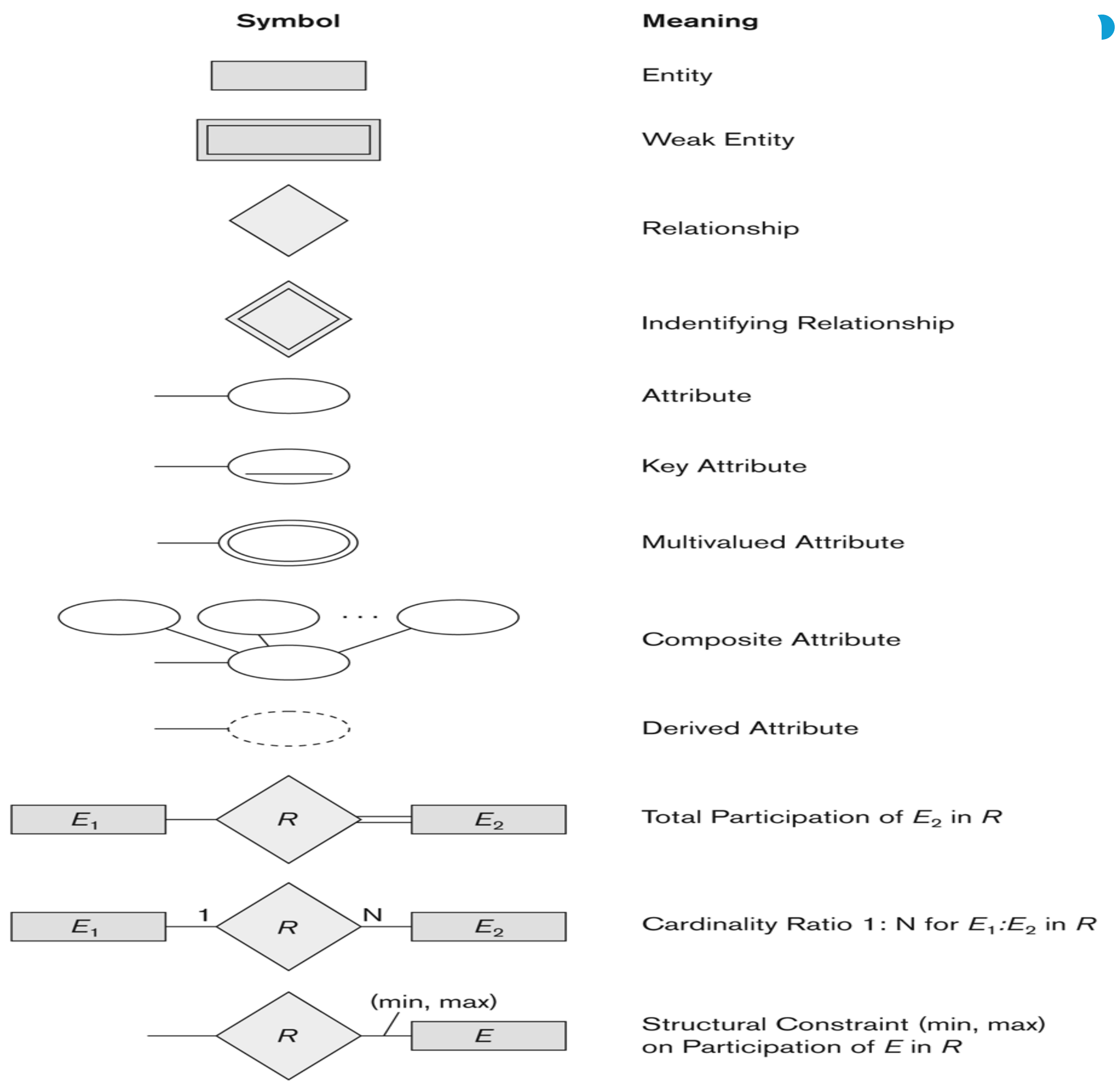
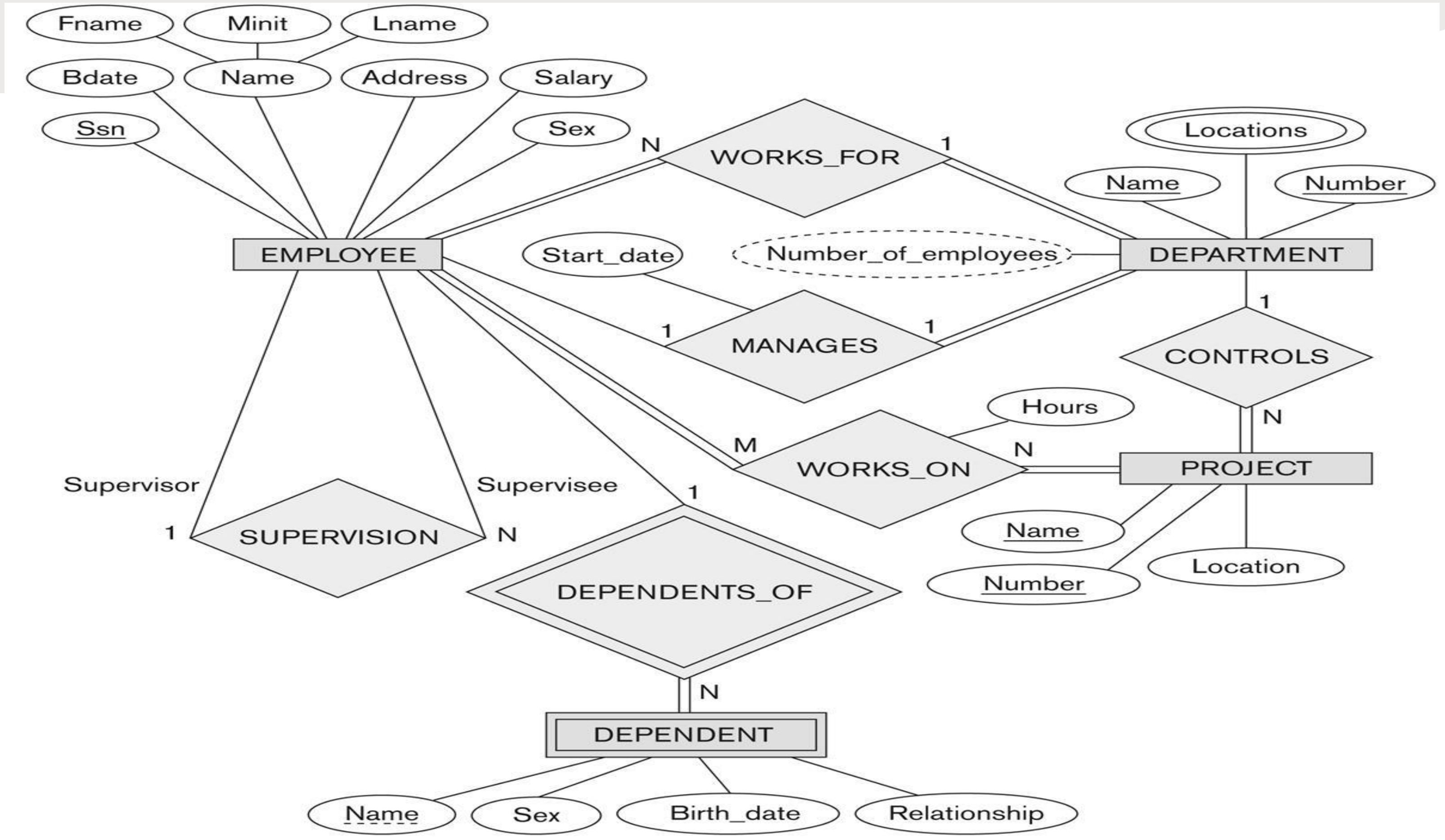


Figure 3.14
Summary of the notation for ER diagrams.


Symbol
Meaning
Entity
Weak Entity
Relationship
Indentifying Relationship
Attribute
Key Attribute
Multivalued Attribute
Composite Attribute
Derived Attribute
E_1
R
E_2
Total Participation of E_2 in R
E_1
1
R
N
E_2
Cardinality Ratio 1 : N for E_1:E_2 in R
(min, max)
R
E
Structural Constraint (min, max) on Participation of E in R

- Chen, Peter *(March 1976). "The Entity-Relationship Model - Toward a Unified View of Data". ACM Transactions on Database Systems.* ***1*** *(1): 9–36.*

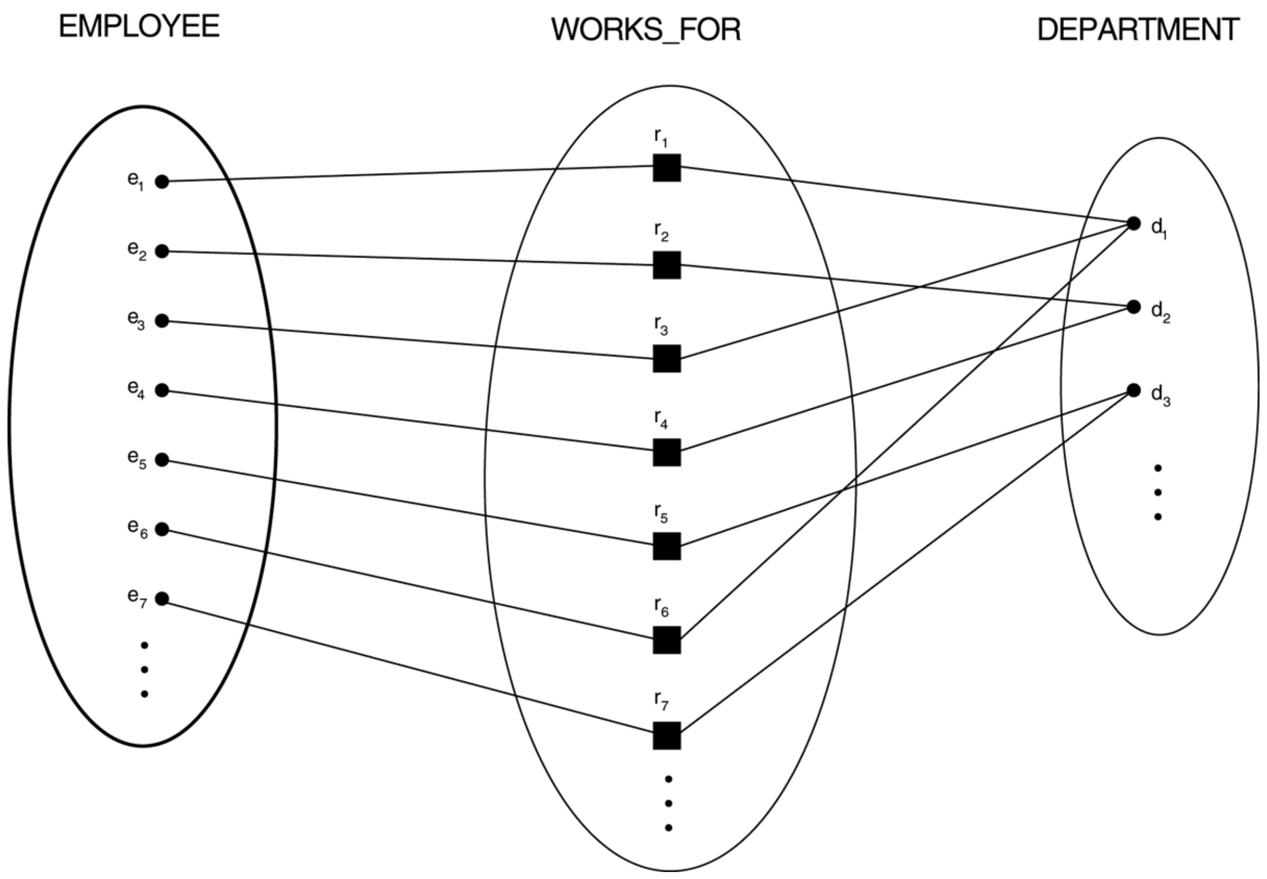
ER DIAGRAM – Entity Types are: EMPLOYEE, DEPARTMENT, PROJECT, DEPENDENT



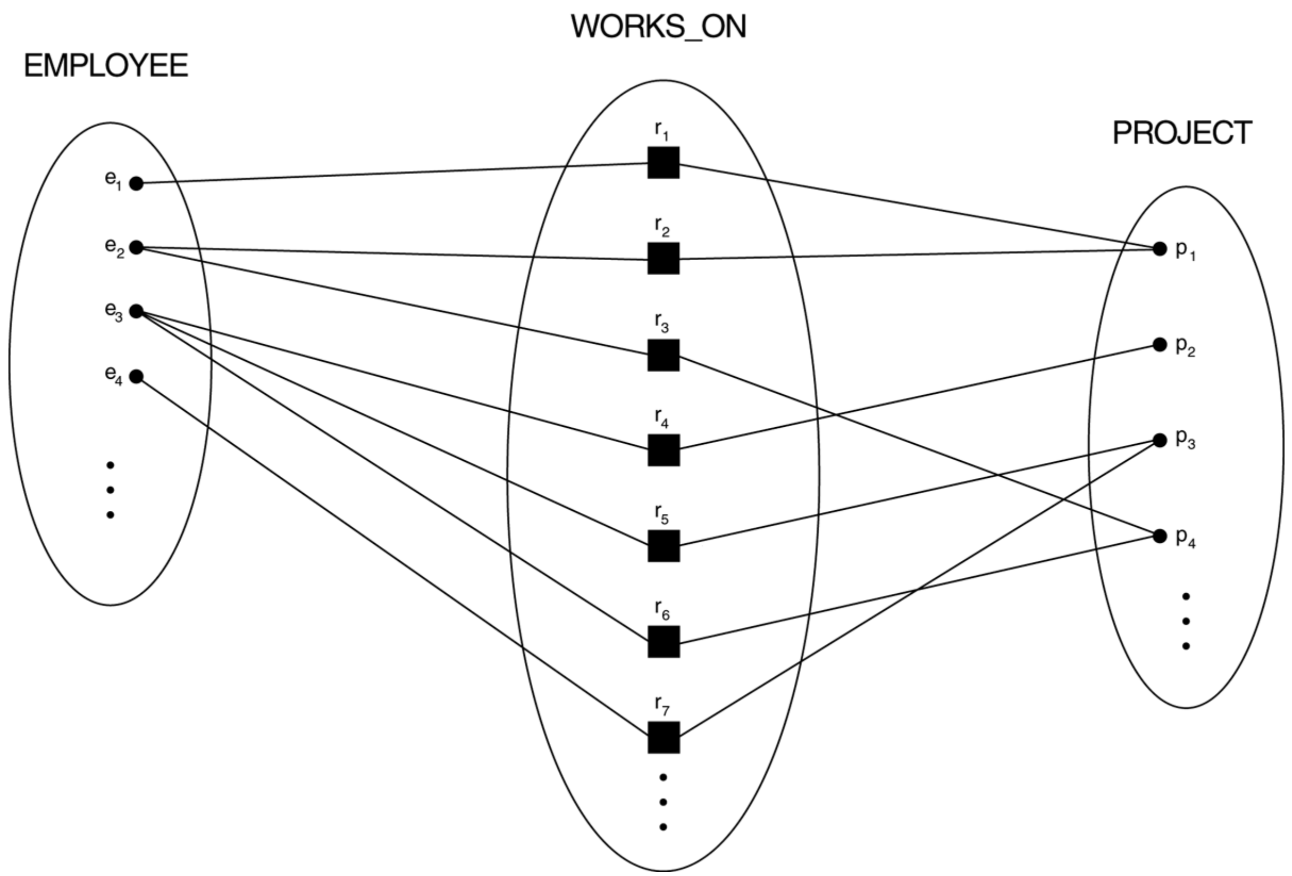
Relationships and Relationship Types (1)

- A relationship relates two or more distinct entities with a specific meaning.
  - For example, EMPLOYEE John Smith works on the ProductX PROJECT or EMPLOYEE Franklin Wong manages the Research DEPARTMENT.
- Relationships of the same type are grouped or typed into a relationship type.
  - For example, the WORKS_ON relationship type in which EMPLOYEEs and PROJECTs participate,
  - or the MANAGES relationship type in which EMPLOYEEs and DEPARTMENTs participate.
- The **degree of a relationship type** is the number of participating entity types.
  - Both MANAGES and WORKS_ON are binary relationships.

FIGURE 3.9
Some instances in the WORKS_FOR relationship set



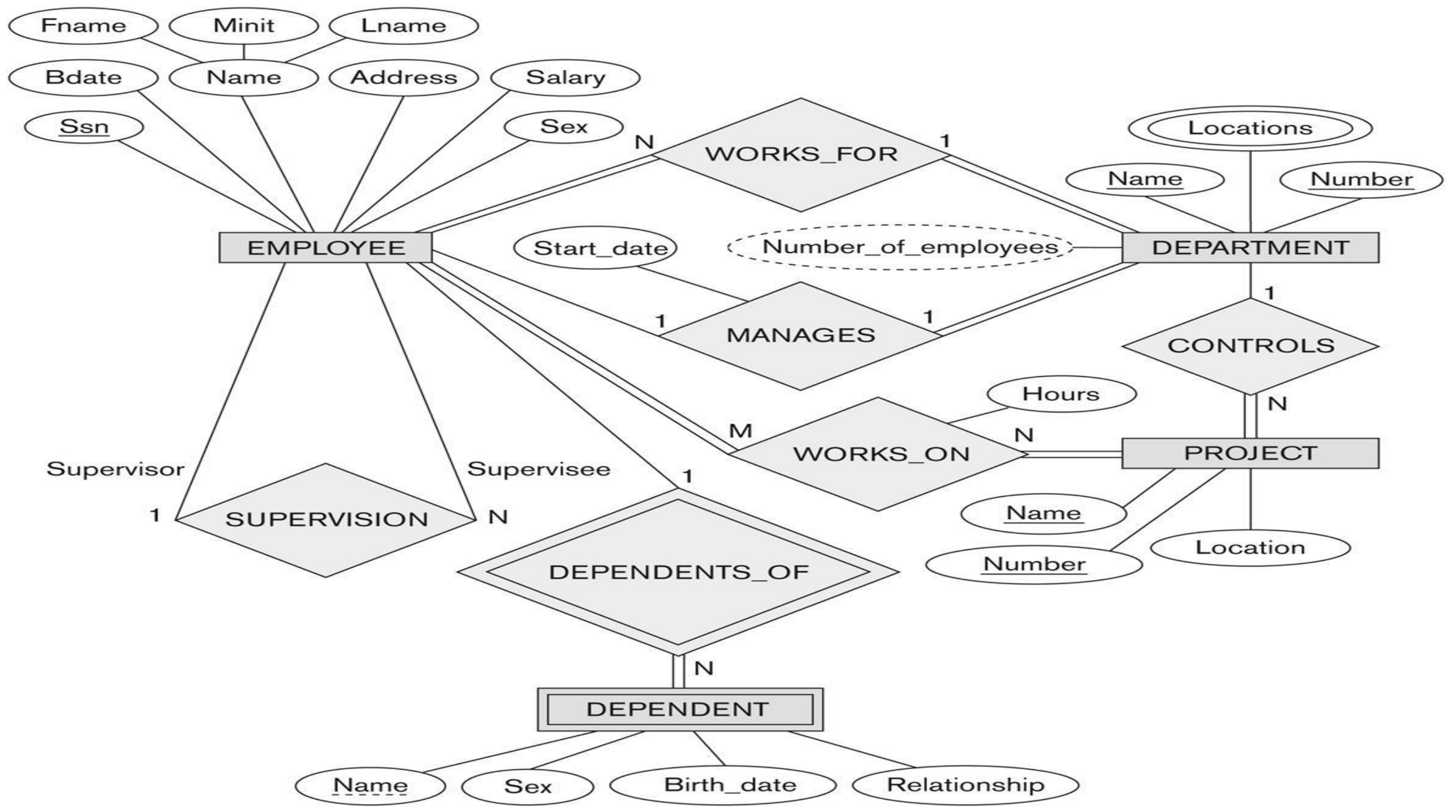
Example relationship instances of the WORKS_ON relationship between EMPLOYEE and PROJECT



Relationships and Relationship Types (2)

- **More than one relationship type can exist with the same participating entity types.**
  - For example, **MANAGES** and **WORKS_FOR** are distinct relationships between **EMPLOYEE** and **DEPARTMENT**, but with different meanings and different relationship instances.

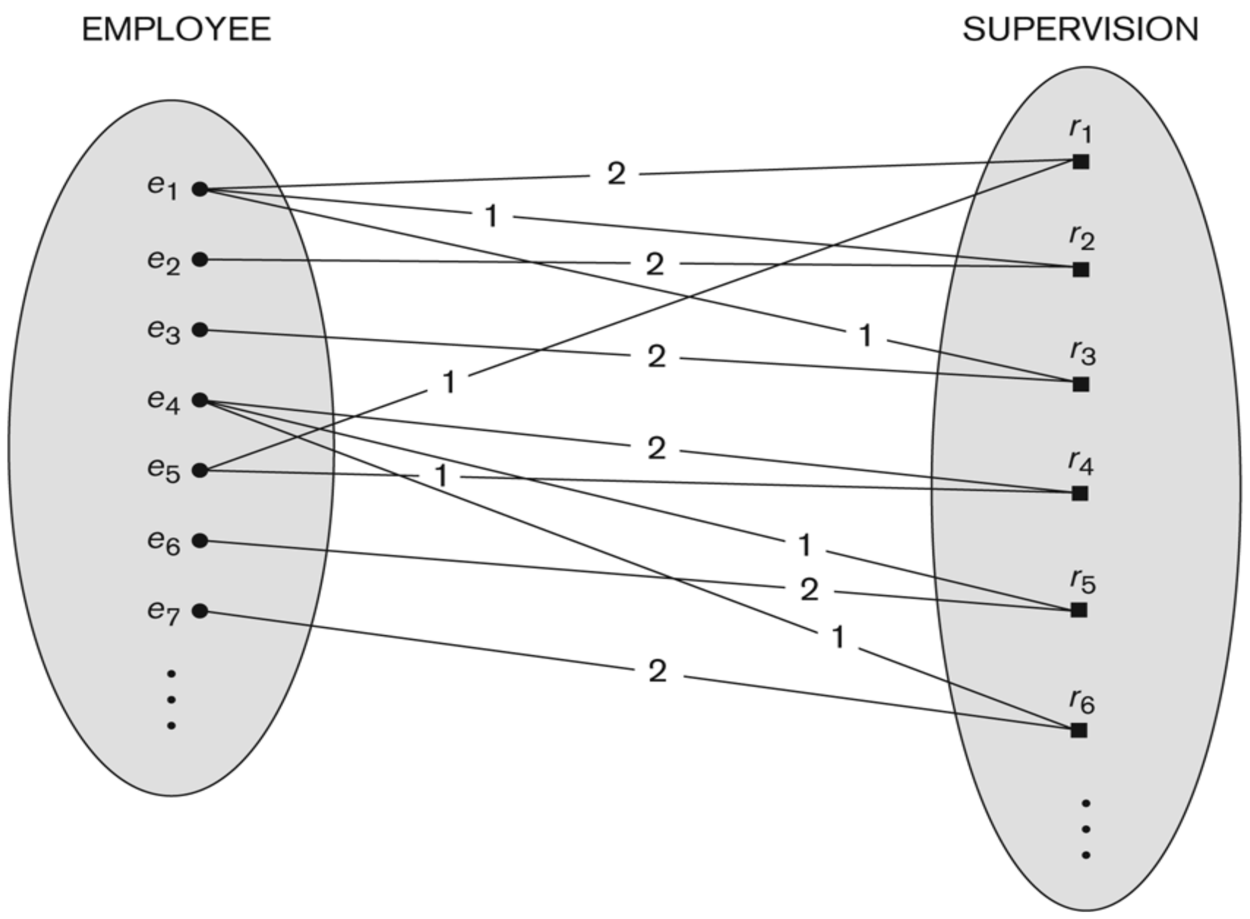
ER DIAGRAM – Relationship Types are: WORKS_FOR, MANAGES, WORKS_ON, CONTROLS, SUPERVISION, DEPENDENTS_OF



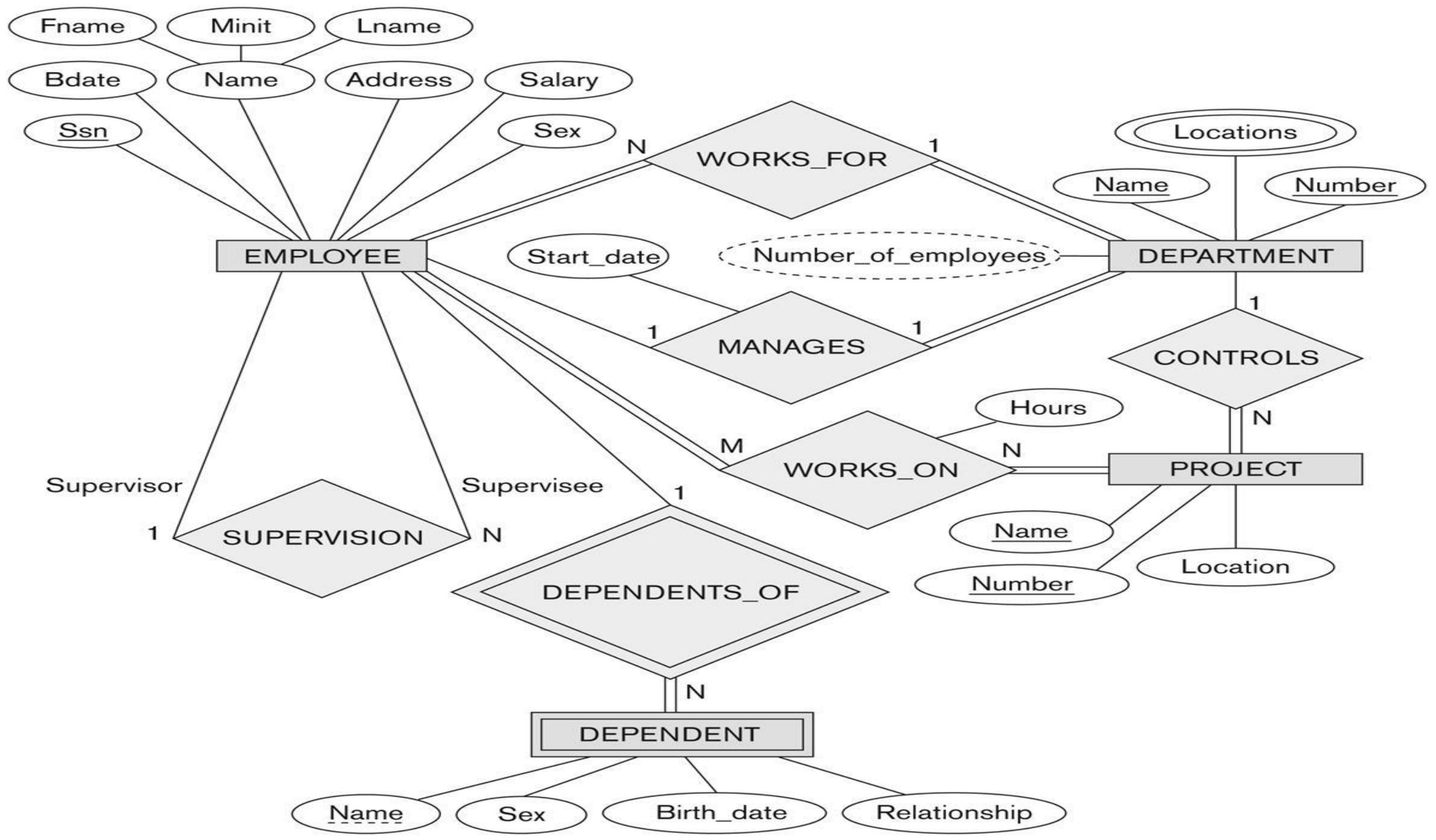
Relationships and Relationship Types (3)

- We can also have a **recursive** relationship type.
- Both participations are same entity type in different roles.
- For example, **SUPERVISION** relationships between EMPLOYEE (in role of **supervisor** or boss) and (another) EMPLOYEE (in role of **subordinate** or worker).
- In following figure, first role participation labeled with 1 and second role participation labeled with 2.
- In ER diagram, need to display role names to distinguish participations.

A Recursive Relationship Type SUPERVISION



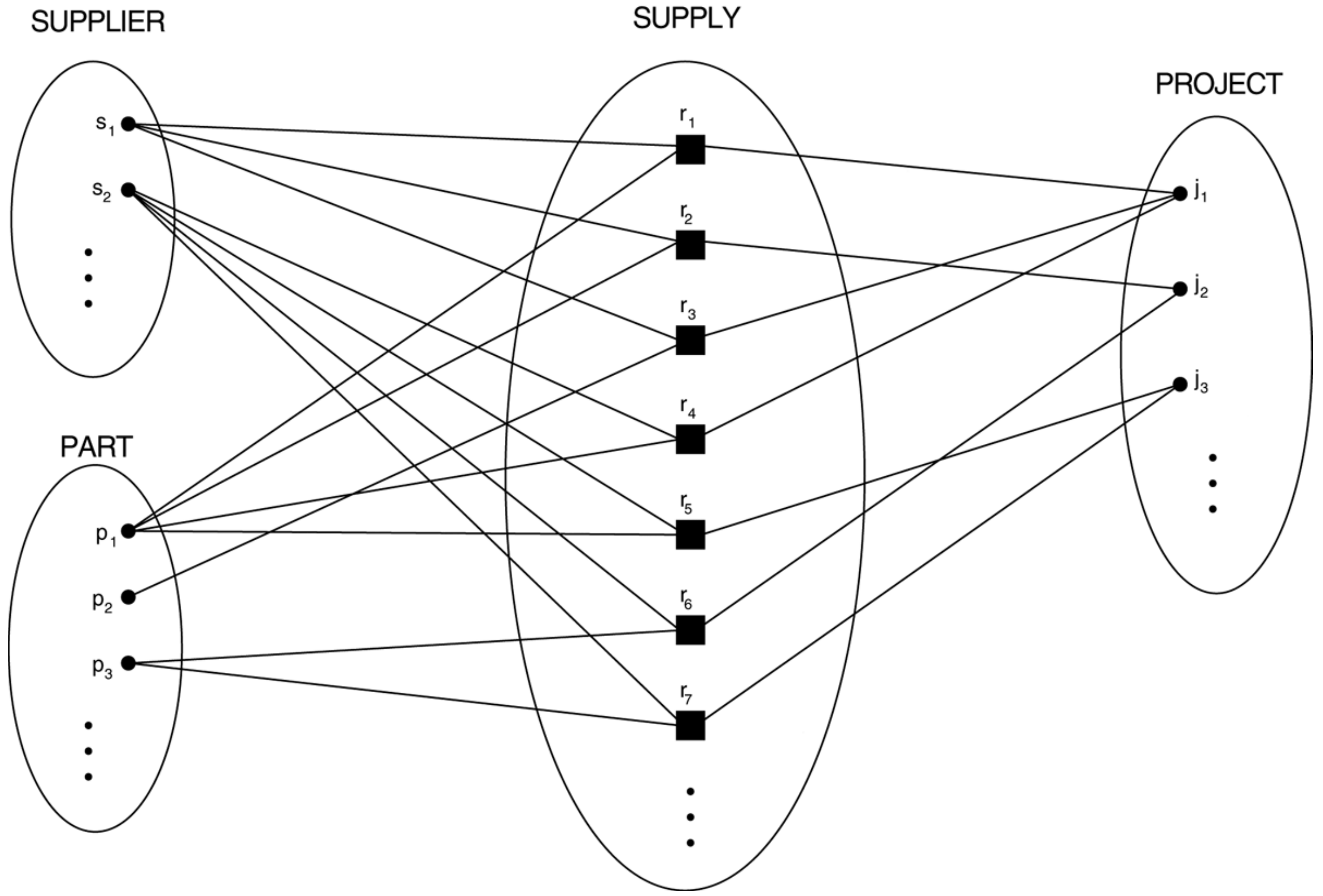
Recursive Relationship Type: SUPERVISION (participation role names are shown)



Relationships of Higher Degree

- Relationship types of degree **2** are called **binary**
- Relationship types of degree **3** are called **ternary** and of degree **n** are called **n-ary**
- In general, an n-ary relationship *is not* equivalent to n binary relationships

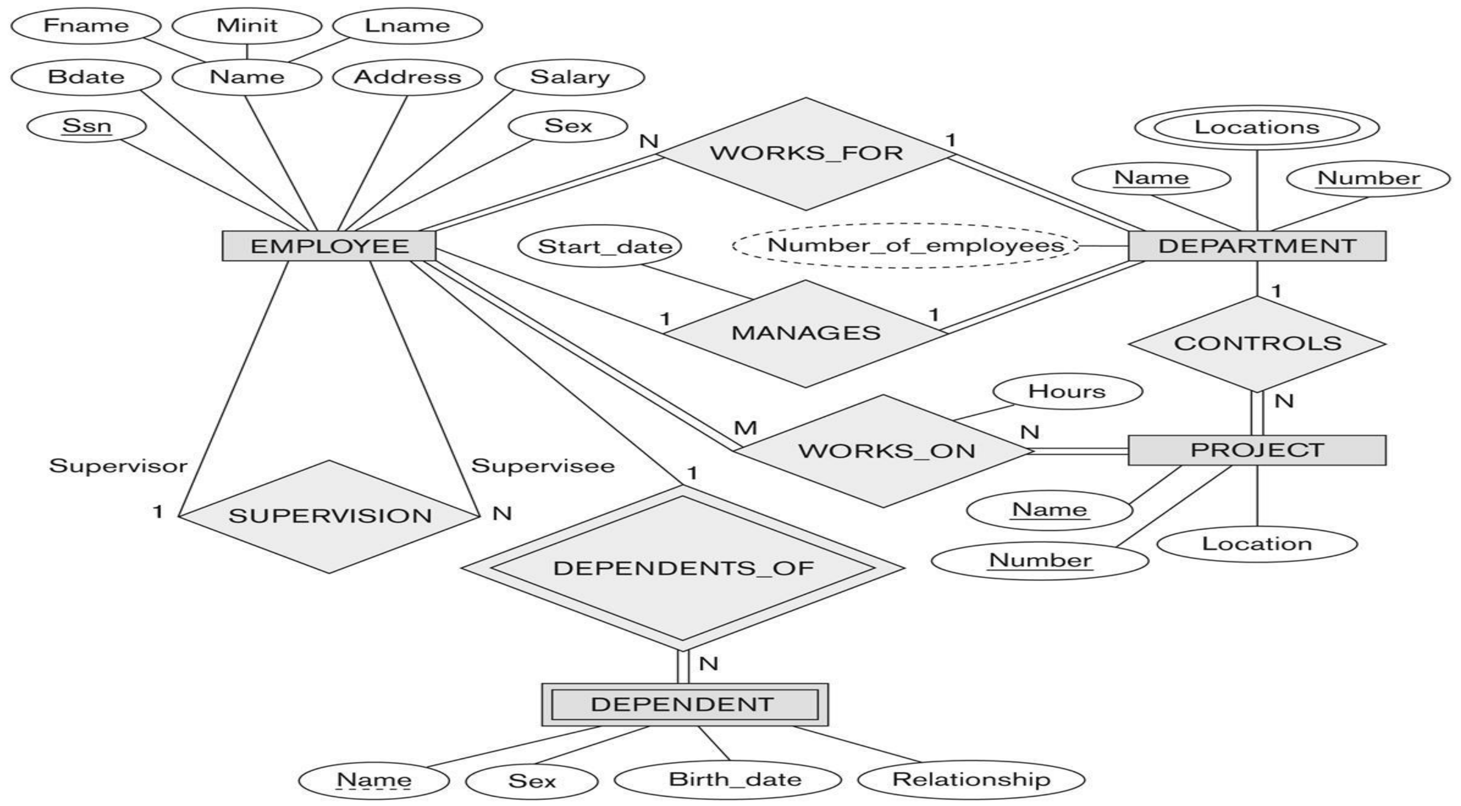
FIGURE 3.10
Some relationship instances in the SUPPLY ternary relationship set.



Attributes of Relationship types

- A relationship type can have attributes;
  - For example, HoursPerWeek of **WORKS_ON**; its value for each relationship instance describes the number of hours per week that an **EMPLOYEE** works on a **PROJECT**.

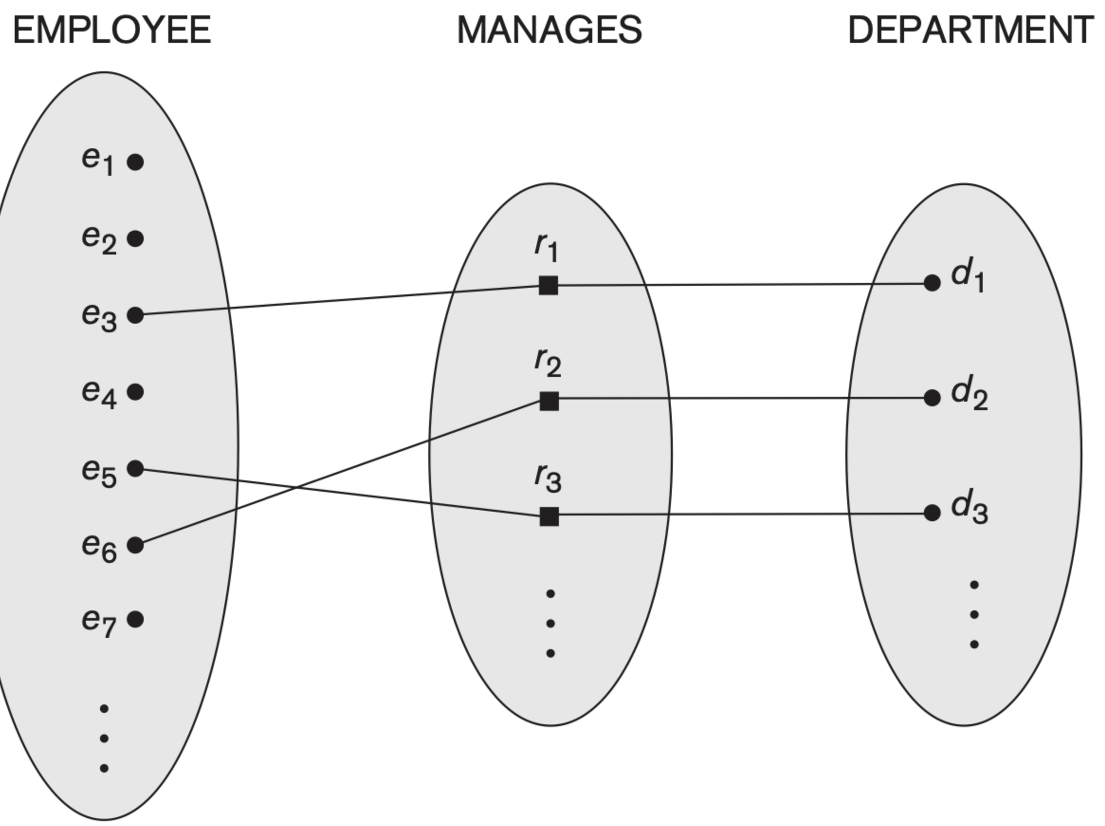
Attribute of a Relationship Type is: Hours of WORKS_ON



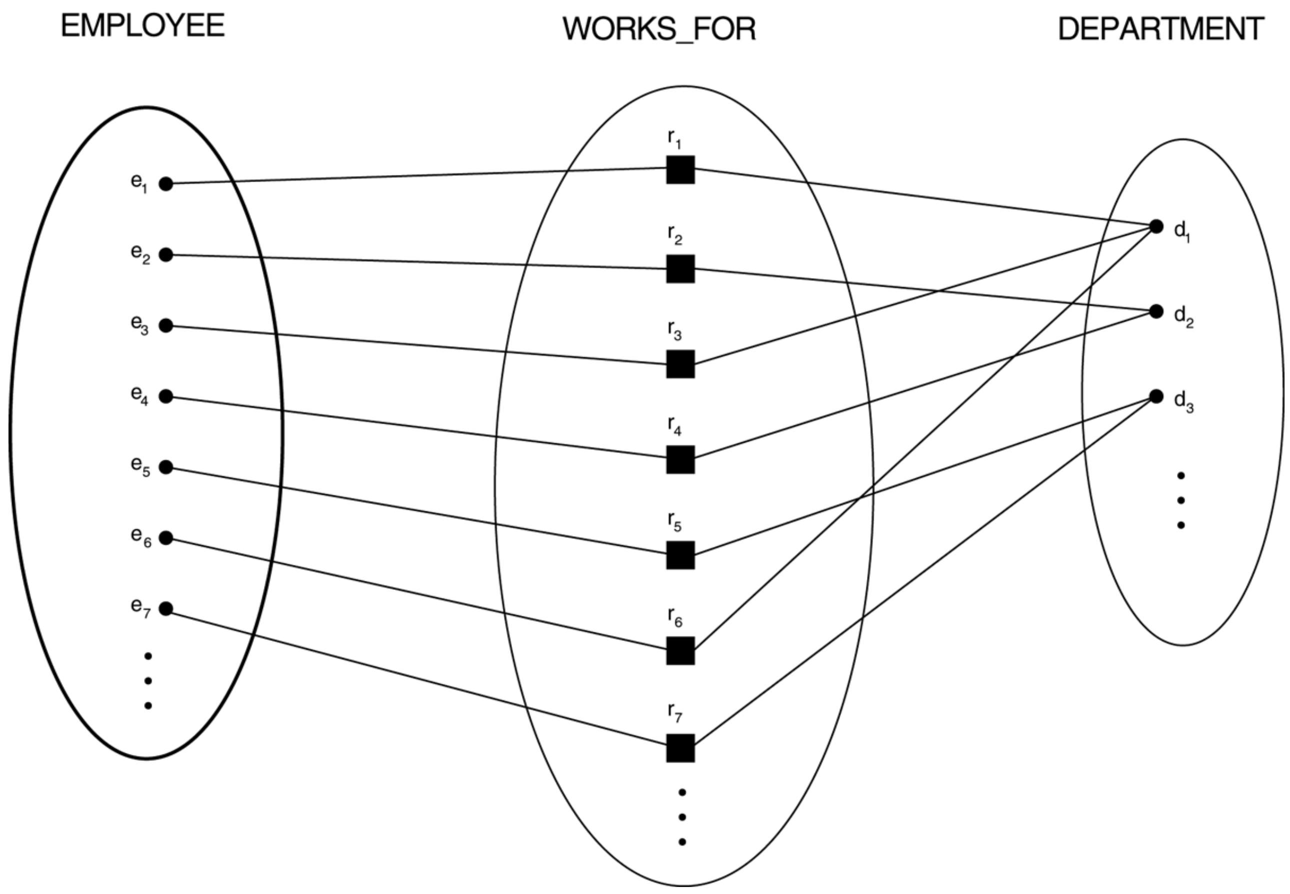
Constraints on Relationships

- **Cardinality Ratios (**SHOWN BY PLACING APPROPRIATE NUMBER ON THE LINK)
  - One-to-one (1:1)
  - One-to-many (1:N) or Many-to-one (N:1)
  - Many-to-many (M:N)
- **Participation constraint**
  - Total participation (one or more - mandatory): **shown by double lining the link**
  - Partial participation (zero – optional): **shown by single lining the link**
- We will refer to the cardinality ratio and participation constraints, taken together, as the **structural constraints** of a relationship type.

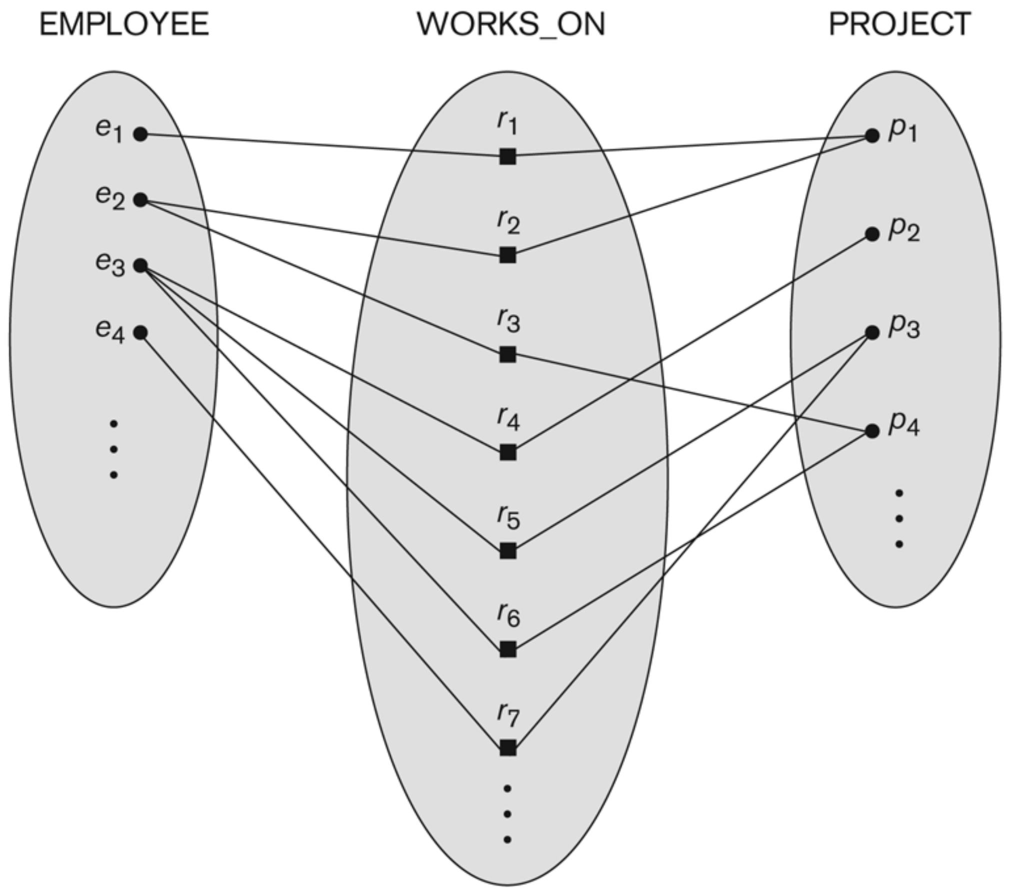
One-to-one (1:1) RELATIONSHIP



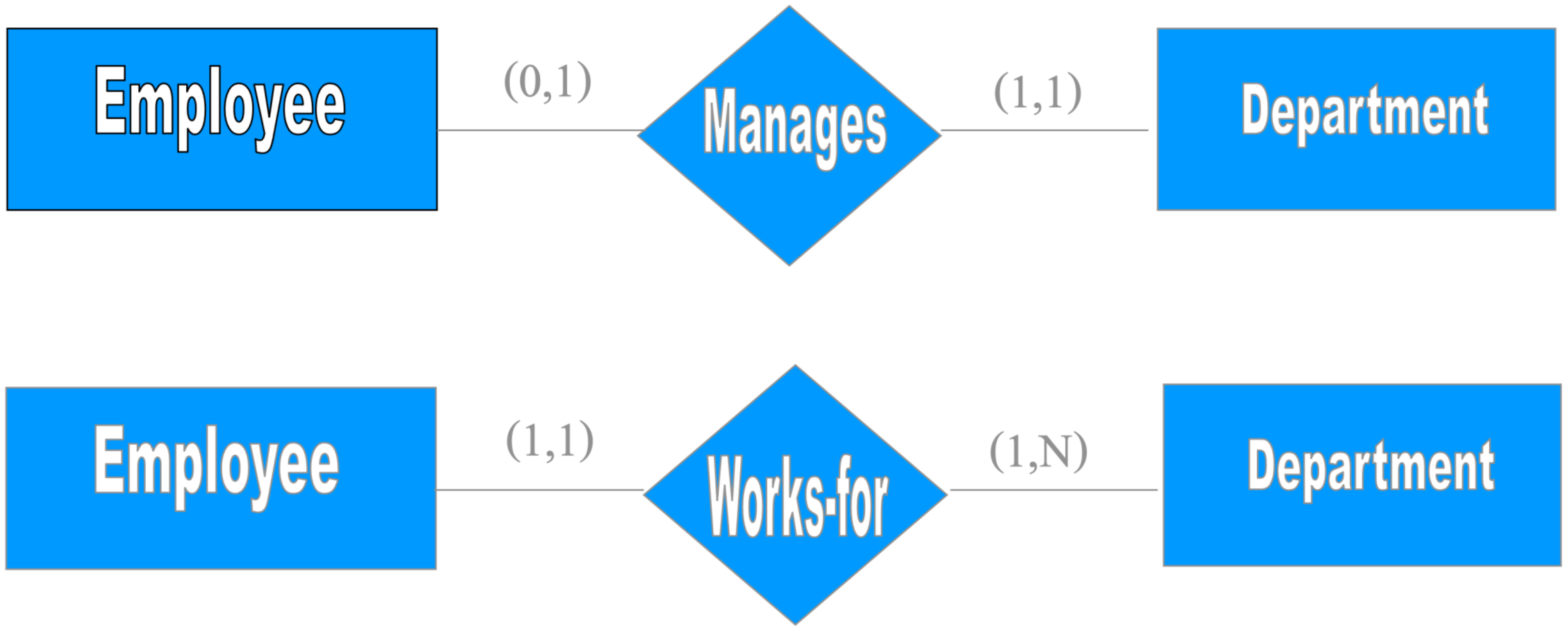
Many-to-one (N:1) RELATIONSHIP



Many-to-many (M:N) RELATIONSHIP

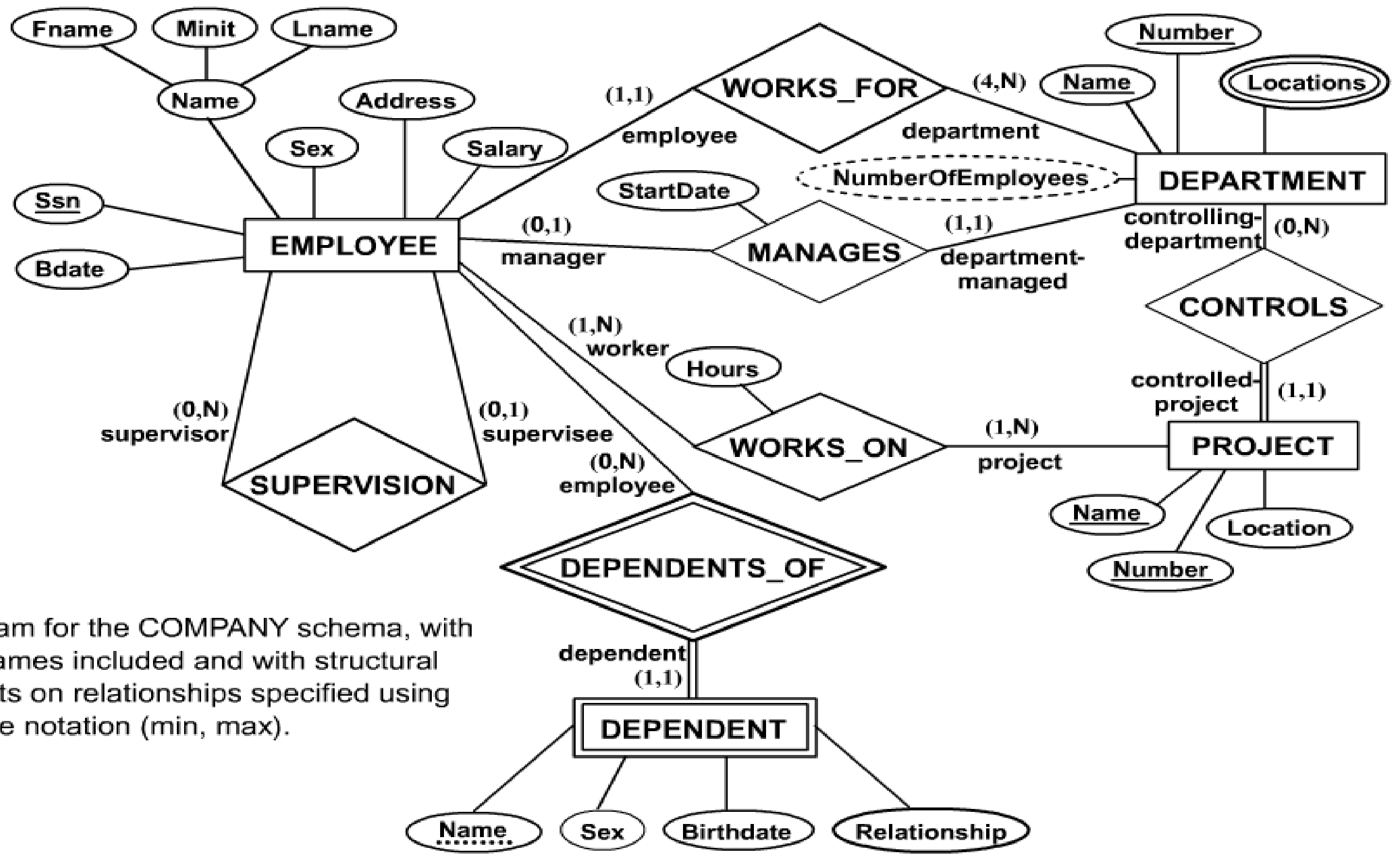


Alternative (min, max) notation for relationship structural constraints:



COMPANY ER Schema Diagram using (min, max) notation

Alternative ER Notations



ER diagram for the COMPANY schema, with all role names included and with structural constraints on relationships specified using alternative notation (min, max).

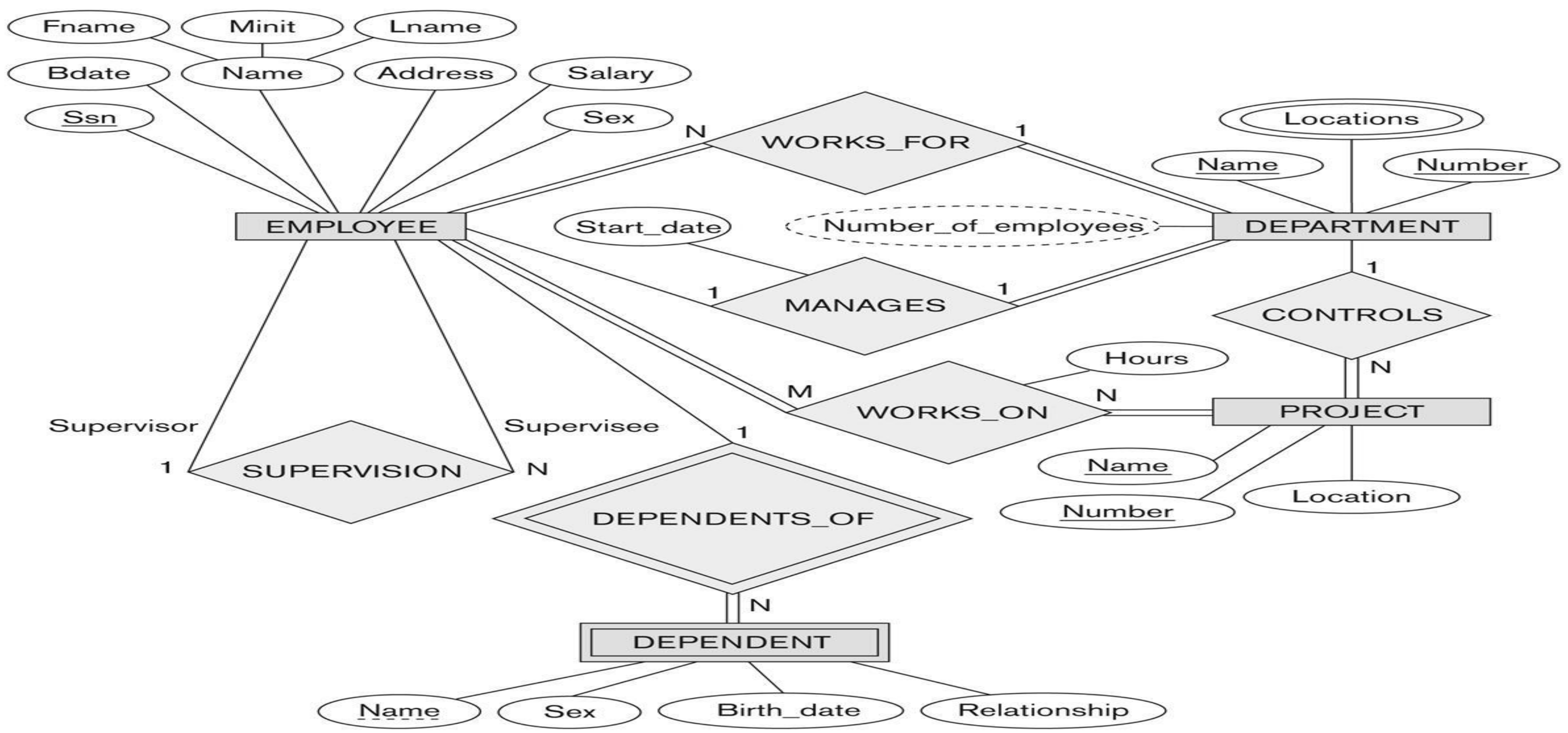
Weak Entity Types

- An entity that does not have a key attribute
- A weak entity must participate in an identifying relationship type with an owner or identifying entity type
- Entities are identified by the combination of:
  - A partial key of the weak entity type
  - A key of the particular entity they are related to in the identifying entity type

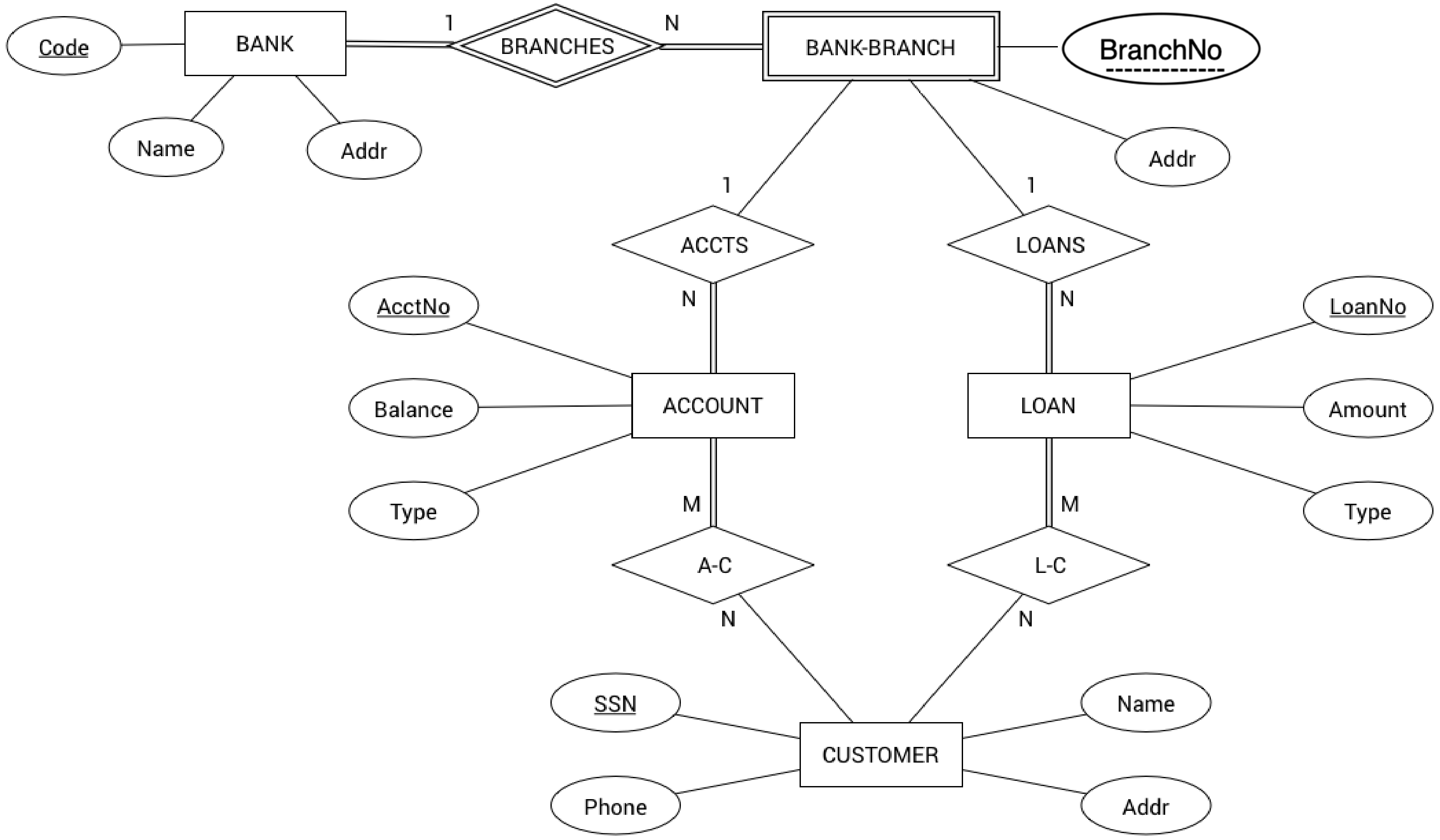
Example:

- Suppose that a DEPENDENT entity is identified by the dependent's first name and birthdate, *and* the specific EMPLOYEE that the dependent is related to.
- DEPENDENT is a weak entity type with EMPLOYEE as its identifying entity type via the identifying relationship type DEPENDENT_OF

Weak Entity Type: DEPENDENT
Identifying Relationship: DEPENDENTS_OF

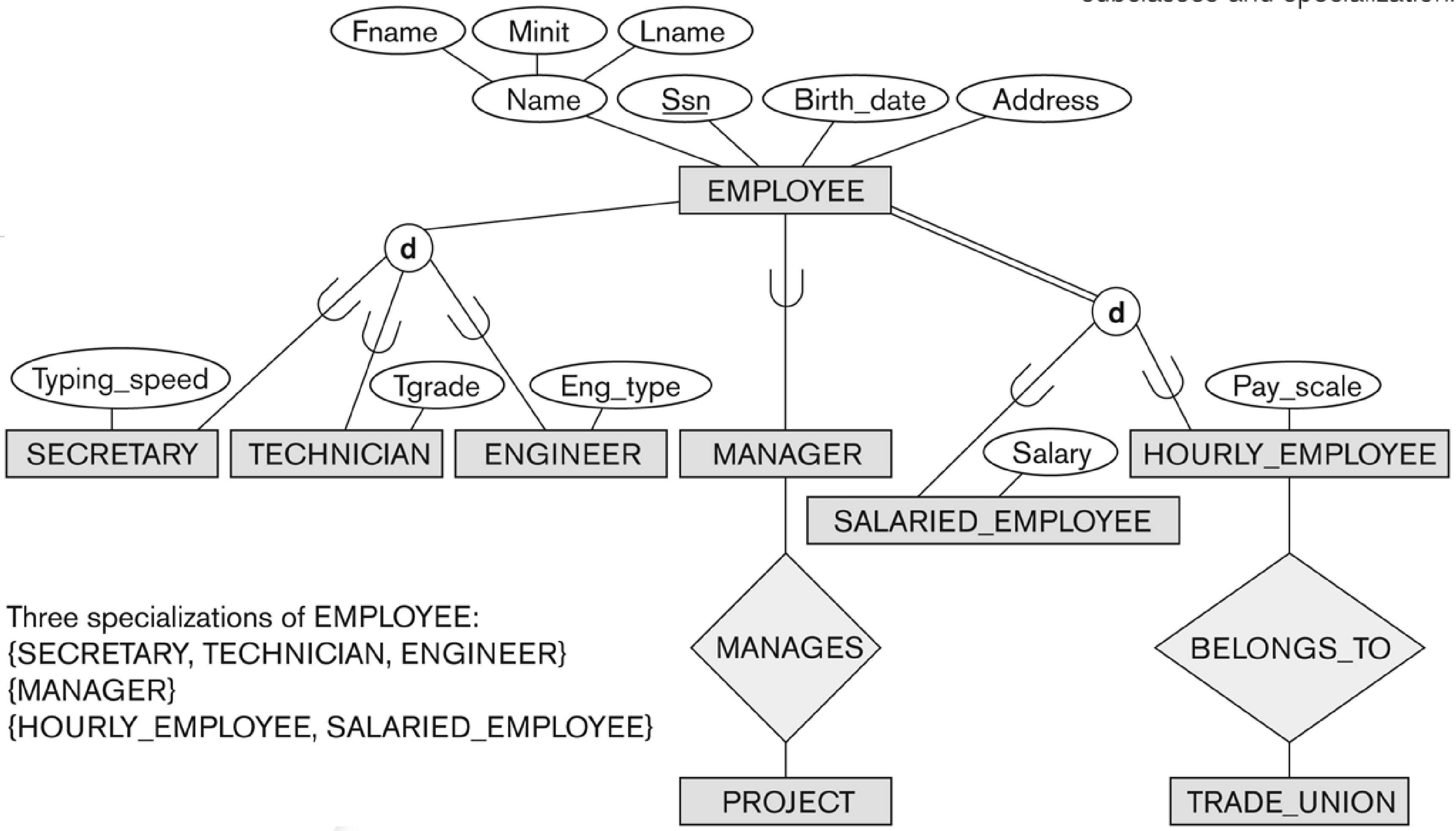


ER DIAGRAM FOR A BANK DATABASE



Subclasses and Superclasses (1)

- An entity type may have additional meaningful subgroupings of its entities
- Example: EMPLOYEE may be further grouped into SECRETARY, ENGINEER, MANAGER, TECHNICIAN, SALARIED_EMPLOYEE, HOURLY_EMPLOYEE,…
  - Each of these groupings is a subset of EMPLOYEE entities
  - Each is called a subclass of EMPLOYEE
  - EMPLOYEE is the superclass for each of these subclasses
- These are called superclass/subclass relationships.
- Example: EMPLOYEE/SECRETARY, EMPLOYEE/TECHNICIAN



Three specializations of EMPLOYEE:
{SECRETARY, TECHNICIAN, ENGINEER}
{MANAGER}
{HOURLY_EMPLOYEE, SALARIED_EMPLOYEE}

Figure 4.1
EER diagram notation to represent subclasses and specialization.

Subclasses and Superclasses (2)

- These are also called IS-A relationships (SECRETARY IS-A EMPLOYEE, TECHNICIAN IS-A EMPLOYEE, ...).
- Note: An entity that is member of a subclass represents the same real-world entity as some member of the superclass
  - The Subclass member is the same entity in a distinct specific role
  - An entity cannot exist in the database merely by being a member of a subclass; it must also be a member of the superclass
  - A member of the superclass can be optionally included as a member of any number of its subclasses
- Example: A salaried employee who is also an engineer belongs to the two subclasses ENGINEER and SALARIED_EMPLOYEE
  - It is not necessary that every entity in a superclass be a member of some subclass

Attribute Inheritance in Superclass / Subclass Relationships

- An entity that is member of a subclass *inherits* all attributes of the entity as a member of the superclass
- It also inherits all relationships

Specialization

- Is the process of defining a set of subclasses of a superclass
- The set of subclasses is based upon some distinguishing characteristics of the entities in the superclass
- Example: {SECRETARY, ENGINEER, TECHNICIAN} is a specialization of EMPLOYEE based upon job type.
  - May have several specializations of the same superclass

Specialization (2)

- Example: Another specialization of EMPLOYEE based in *method of pay* is {SALARIED_EMPLOYEE, HOURLY_EMPLOYEE}.
  - Superclass/subclass relationships and specialization can be diagrammatically represented in ER diagrams
  - Attributes of a subclass are called specific attributes. For example, TypingSpeed of SECRETARY
  - The subclass can participate in specific relationship types. For example, BELONGS_TO of HOURLY_EMPLOYEE

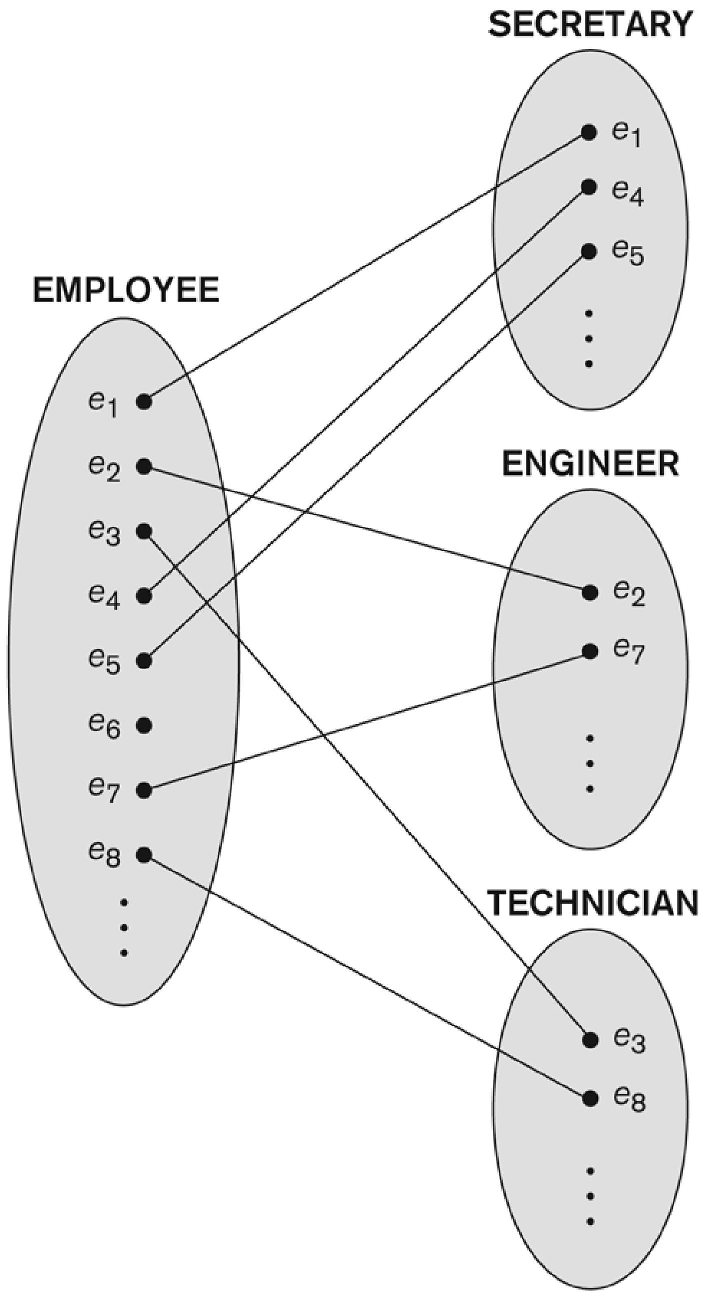
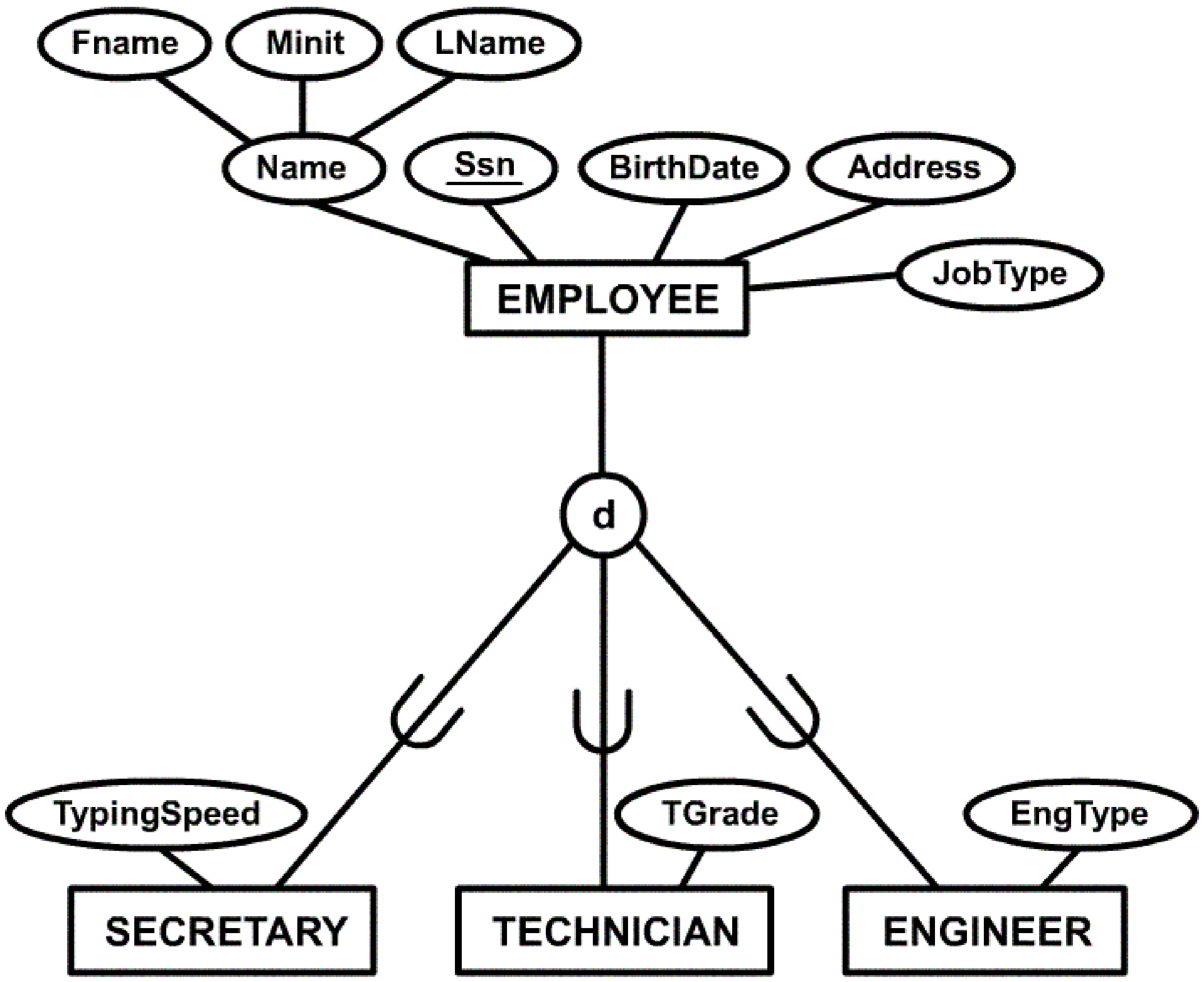


Figure 4.2
Instances of a specialization.

Example of a Specialization



Generalization

- The reverse of the specialization process
- Several classes with common features are generalized into a superclass; original classes become its subclasses
- Example: CAR, TRUCK generalized into VEHICLE; both CAR, TRUCK become subclasses of the superclass VEHICLE.
  - We can view {CAR, TRUCK} as a specialization of VEHICLE
  - Alternatively, we can view VEHICLE as a generalization of CAR and TRUCK

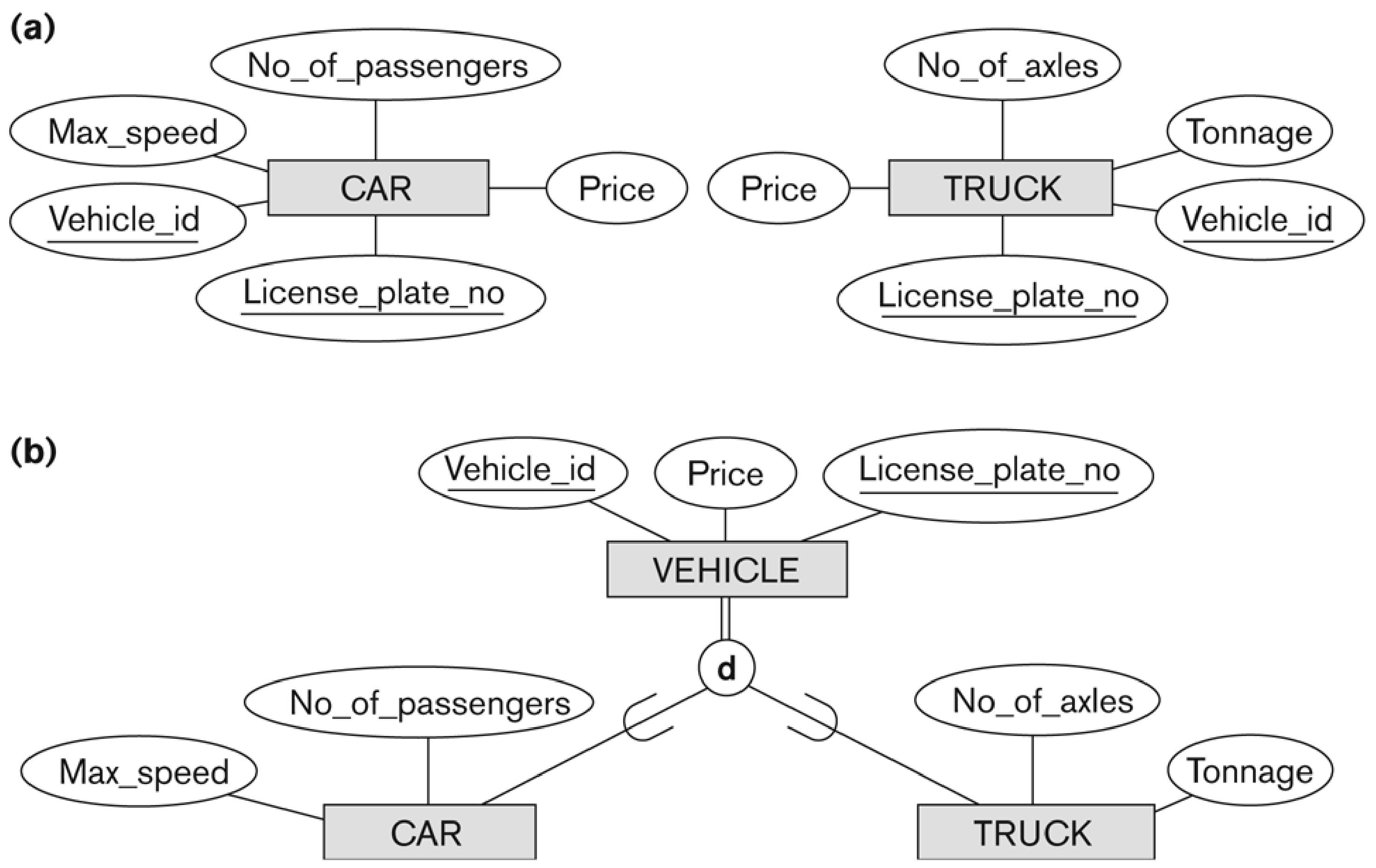


Figure 4.3
Generalization. (a) Two entity types, CAR and TRUCK.
(b) Generalizing CAR and TRUCK into the superclass VEHICLE.

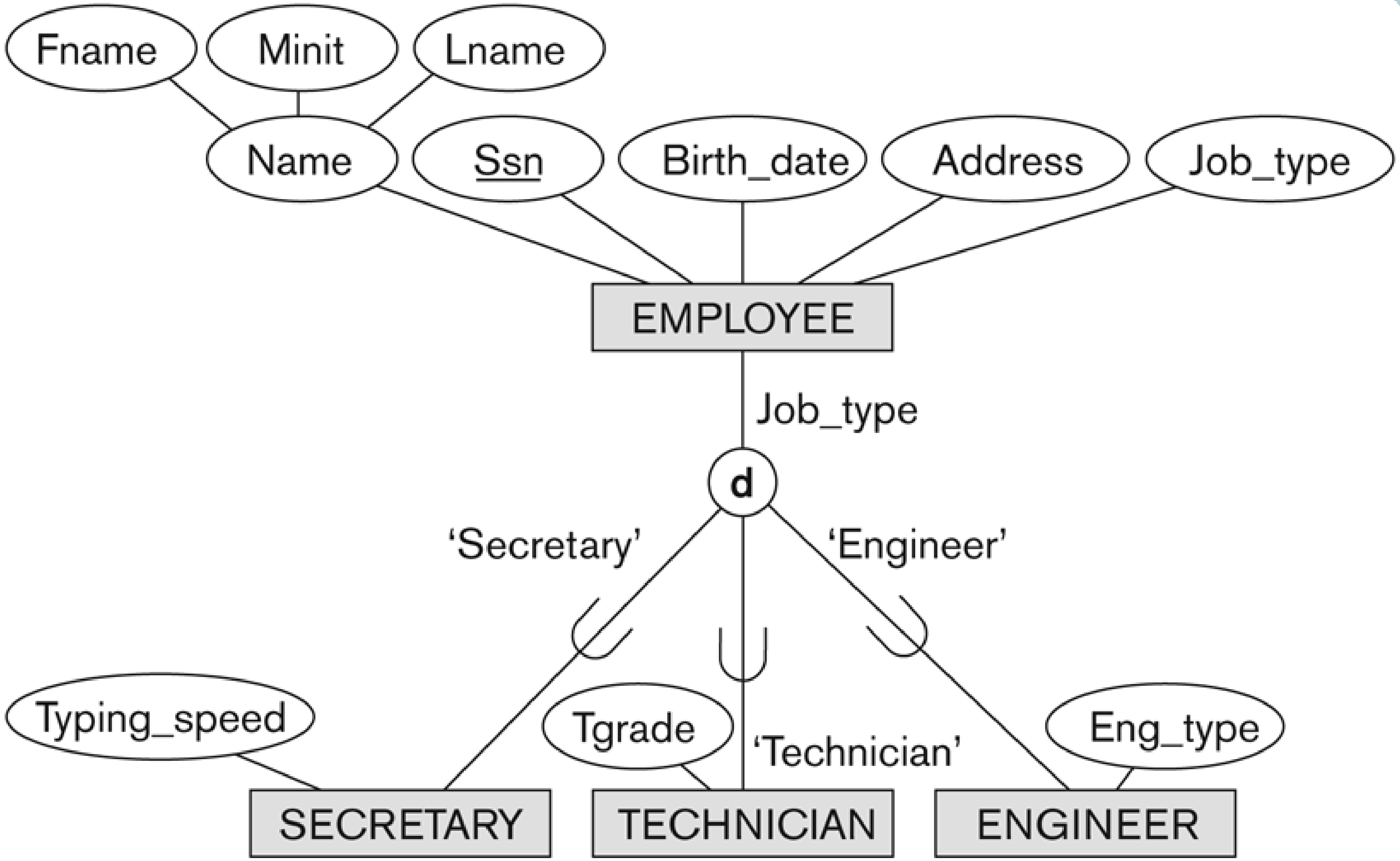
Constraints on Specialization and Generalization (1)

- If we can determine exactly those entities that will become members of each subclass by a condition, the subclasses are called ***predicate-defined*** (or condition-defined) subclasses
  - Condition is a constraint that determines subclass members
  - Display a predicate-defined subclass by writing the predicate condition next to the line attaching the subclass to its superclass
- If all subclasses in a specialization have membership condition *on same attribute of the superclas*s, specialization is called an ***attribute defined-*** **specialization**
  - Attribute is called the defining attribute of the specialization
  - Example: JobType is the defining attribute of the specialization {SECRETARY, TECHNICIAN, ENGINEER} of EMPLOYEE

Constraints on Specialization and Generalization (2)

- If no condition determines membership, the subclass is called *user-defined*
  - Membership in a subclass is determined by the database users by applying an operation to add an entity to the subclass
  - Membership in the subclass is specified individually for each entity in the superclass by the user

Figure 4.4
EER diagram notation for an attribute-defined specialization on Job_type.



Constraints on Specialization and Generalization (3)

- Two other conditions apply to a specialization/generalization:
- **Disjointness Constraint:**
    - Specifies that the subclasses of the specialization must be disjointed (an entity can be a member of at most one of the subclasses of the specialization)
    - Specified by **d** in EER diagram
    - If not disjointed, overlap; that is the same entity may be a member of more than one subclass of the specialization
    - Specified by **o** in EER diagram
- **Completeness Constraint:**
    - **Total** specifies that every entity in the superclass must be a member of some subclass in the specialization/ generalization
    - Shown in EER diagrams by a **double line**
    - **Partial** allows an entity not to belong to any of the subclasses
    - Shown in EER diagrams by a **single line**

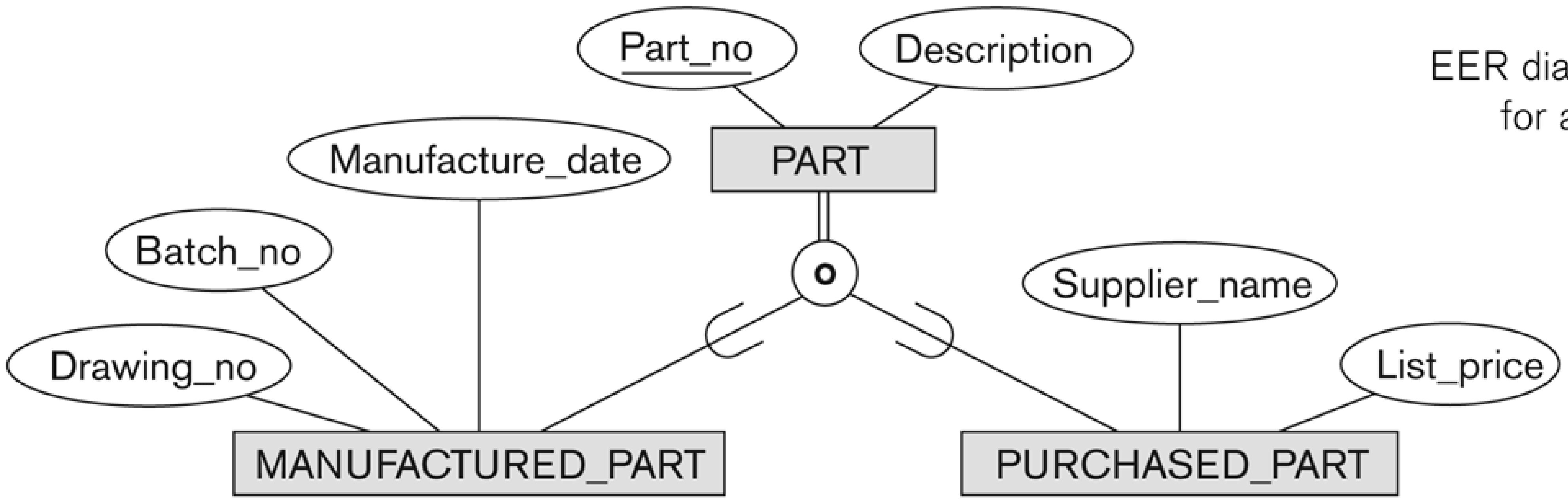
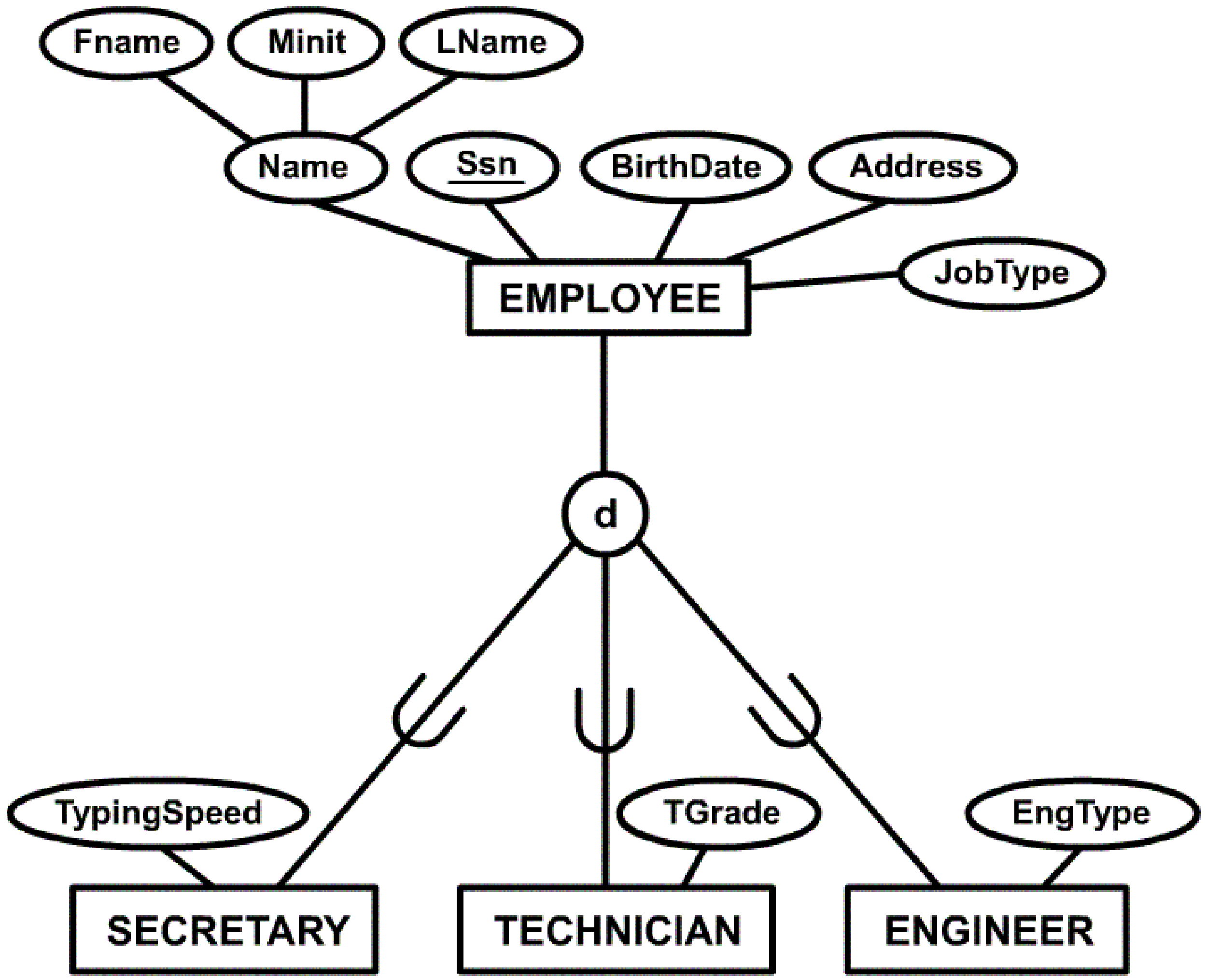


Figure 4.5
EER diagram notation for an overlapping (nondisjoint) specialization.

Constraints on Specialization and Generalization (4)

- Hence, we have four types of specialization/generalization:
  - Disjoint, total
  - Disjoint, partial
  - Overlapping, total
  - Overlapping, partial
- Note: Generalization usually is total because the superclass is derived from the subclasses.

Example of disjoint partial Specialization



Constraints on Specialization and Generalization (5)

- **Insertion and deletion rules**
  - **Deleting** an entity from a superclass implies that it is automatically deleted from all the subclasses to which it belongs
  - **Inserting** an entity in a superclass implies that the entity is mandatorily inserted in all predicate-defined (or attribute-defined) subclasses for which the entity satisfies the defining predicate
  - **Inserting** an entity in a superclass of a total specialization implies that the entity is mandatorily inserted in at least one of the subclasses of the specialization

Specialization / Generalization Hierarchies, Lattices and Shared Subclasses

A subclass may itself have further subclasses specified on it forms a hierarchy or a lattice

Hierarchy has a constraint that every subclass has only one superclass (called *single inheritance*)

In a lattice, a subclass can be subclass of more than one superclass (called *multiple inheritance*)

In a lattice or hierarchy, a subclass inherits attributes not only of its direct superclass, but also of all its predecessor superclasses

Specialization / Generalization Hierarchies, Lattices and Shared Subclasses (2)

- A subclass with more than one superclass is called a shared subclass
- Can have specialization hierarchies or lattices, or generalization hierarchies or lattices
- In specialization, start with an entity type and then define subclasses of the entity type by successive specialization (top-down conceptual refinement process)
- In generalization, start with many entity types and generalize those that have common properties (bottom-up conceptual synthesis process)
- In practice, the combination of two processes is employed

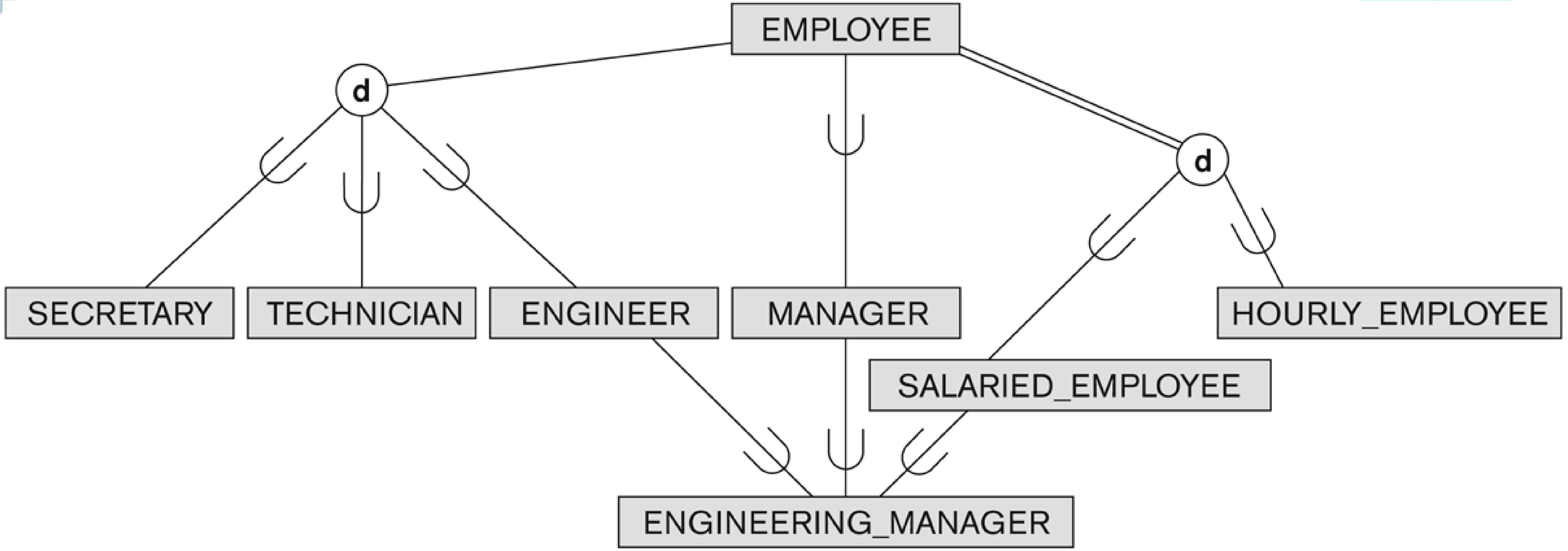


Figure 4.6
A specialization lattice with shared subclass ENGINEERING_MANAGER.

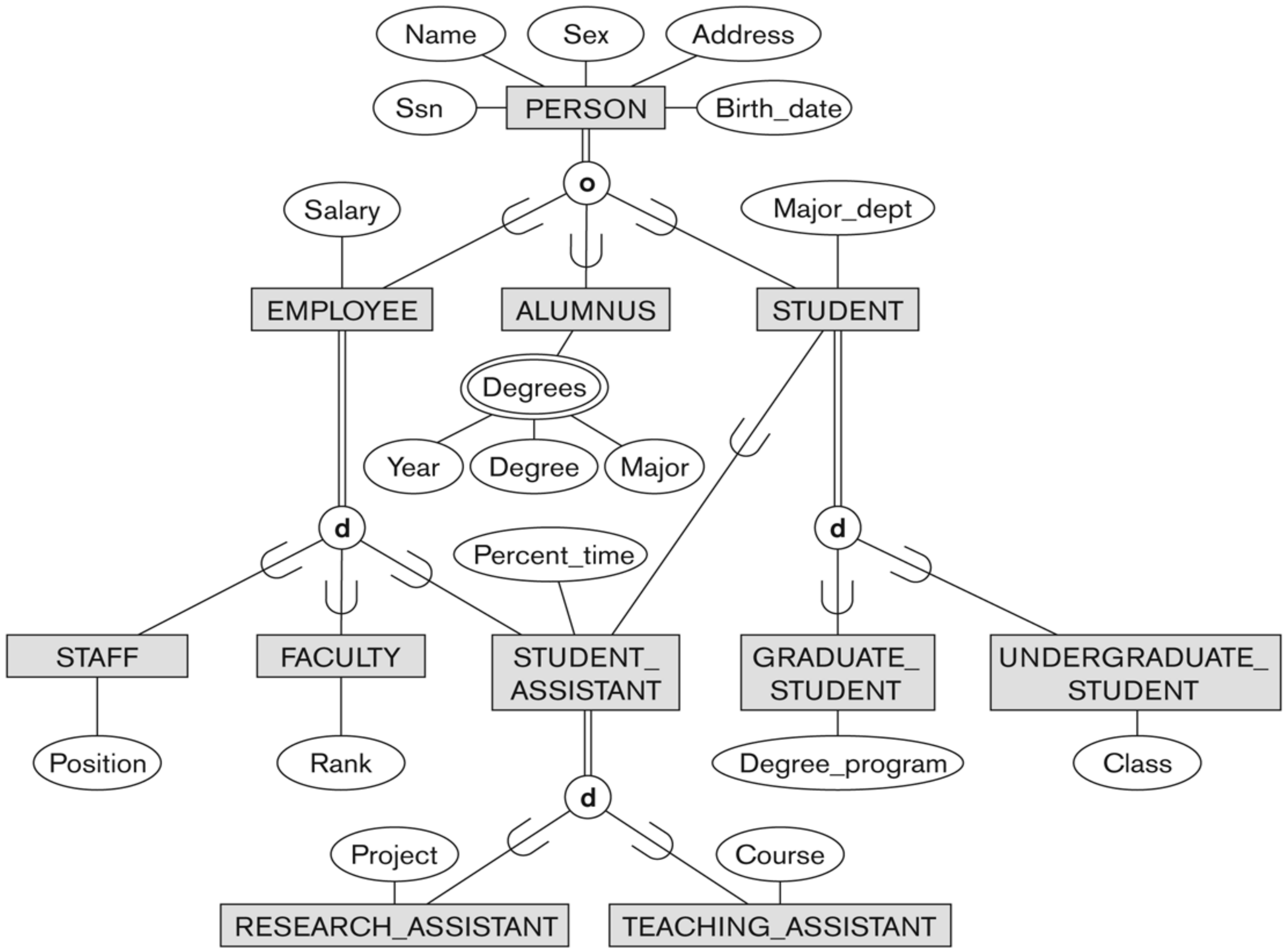


Figure 4.7
A specialization lattice with multiple inheritance for a UNIVERSITY database.

Categories (UNION TYPES)

- All the superclass/subclass relationships we have seen thus far have a single superclass
- A shared subclass is subclass in more than one distinct superclass/subclass relationships, where each relationships has a single superclass (multiple inheritance)
- In some cases, need to model a single superclass/subclass relationship with more than one superclass
- Superclasses represent different entity types
- Such a subclass is called a category or UNION TYPE

Categories (UNION TYPES) (2)

- Example: Database for vehicle registration, vehicle owner can be a person, a bank (holding a lien on a vehicle) or a company.
    - Category (subclass) OWNER is a subset of the union of the three superclasses COMPANY, BANK, and PERSON
    - A category member must exist in at least one of its superclasses
- Note: The difference from shared subclass, which is subset of the intersection of its superclasses (shared subclass member must exist in all of its superclasses).

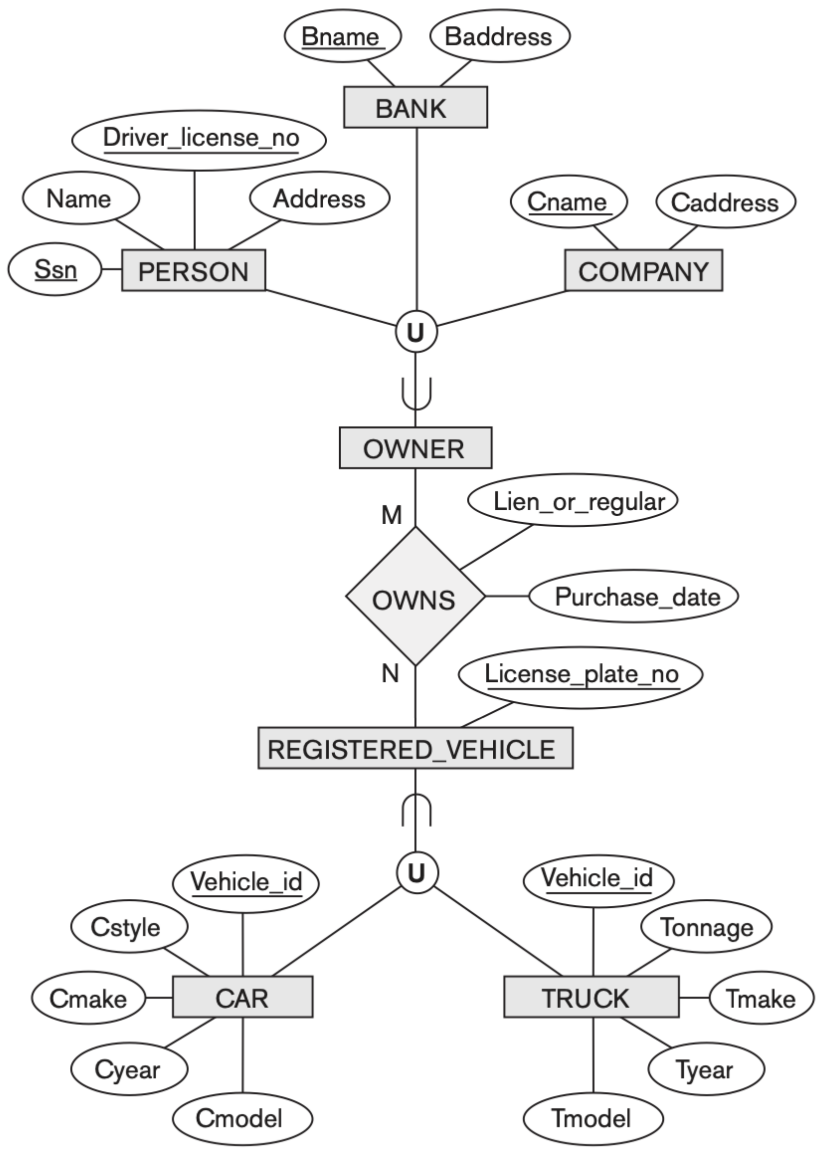
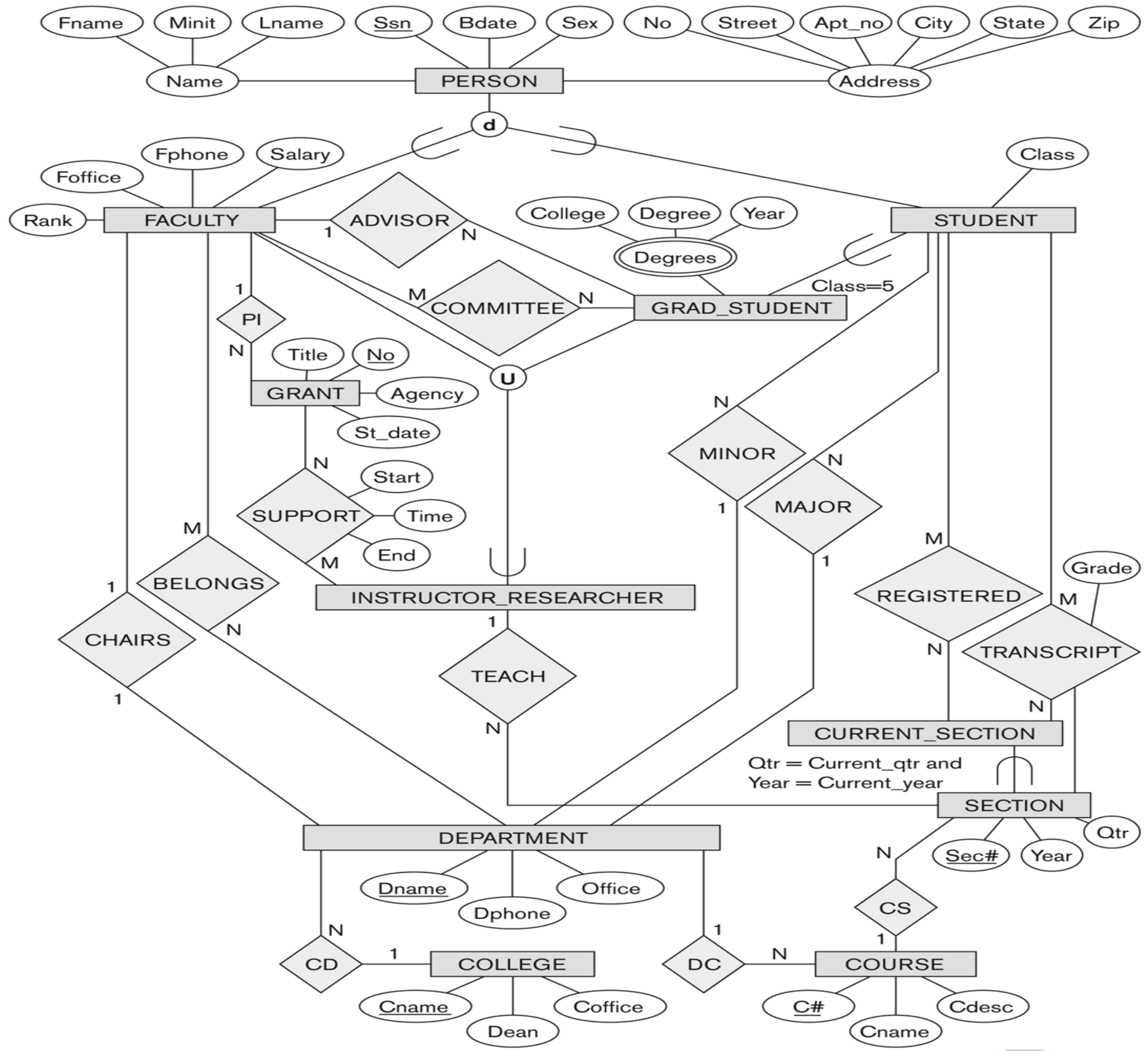
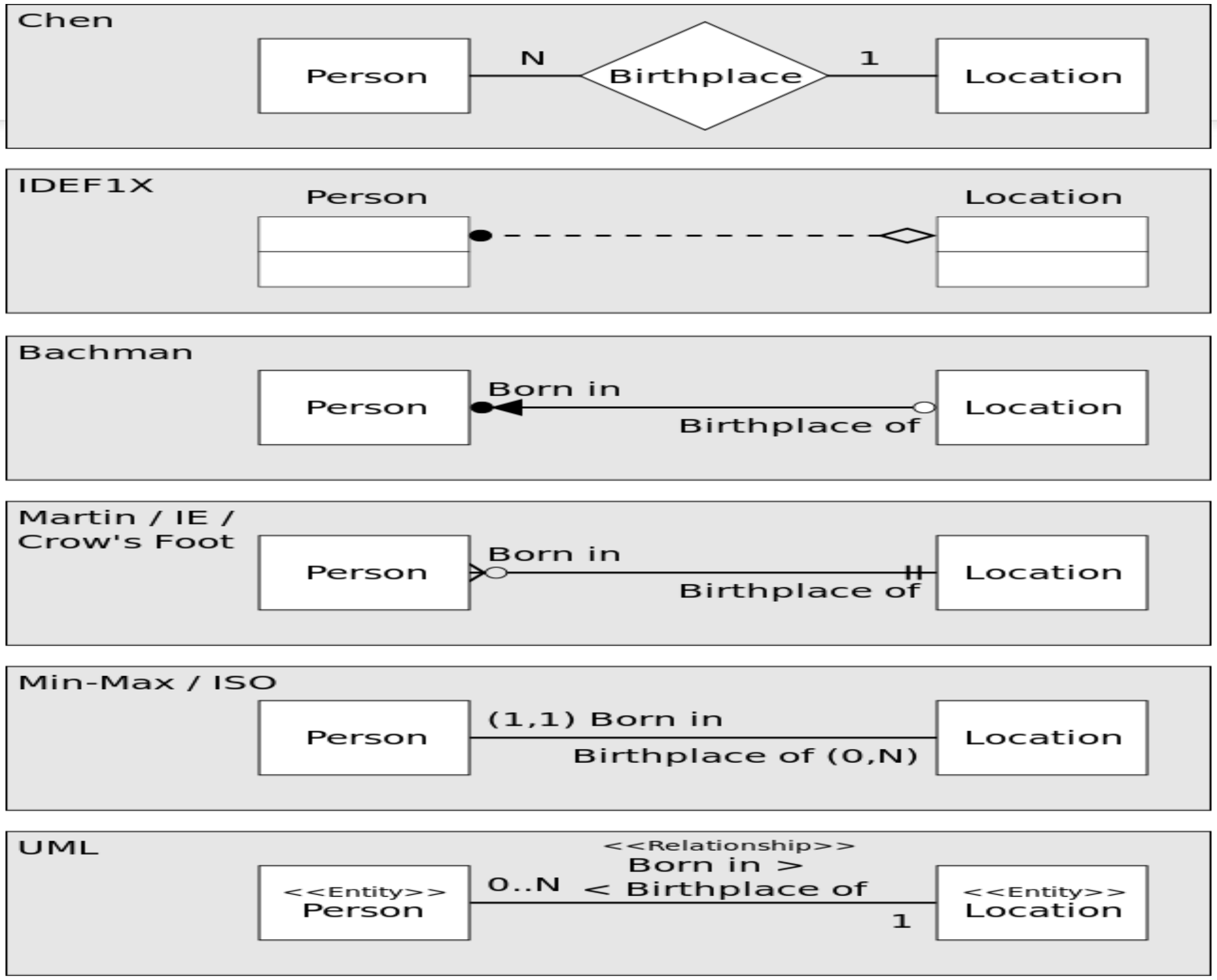


Figure 4.8
Two categories (union types): OWNER and REGISTERED_VEHICLE.

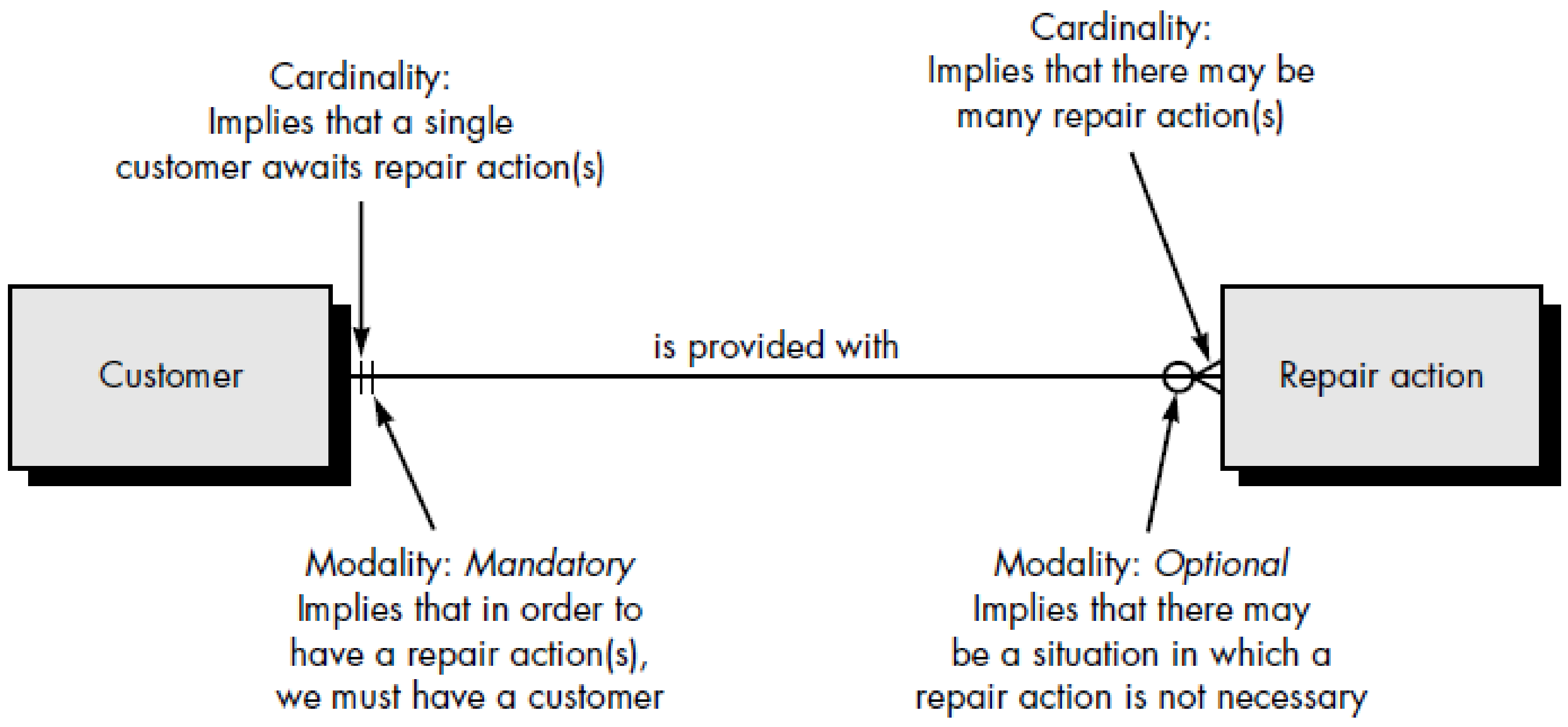
Figure 4.9
An EER conceptual schema for a UNIVERSITY database.



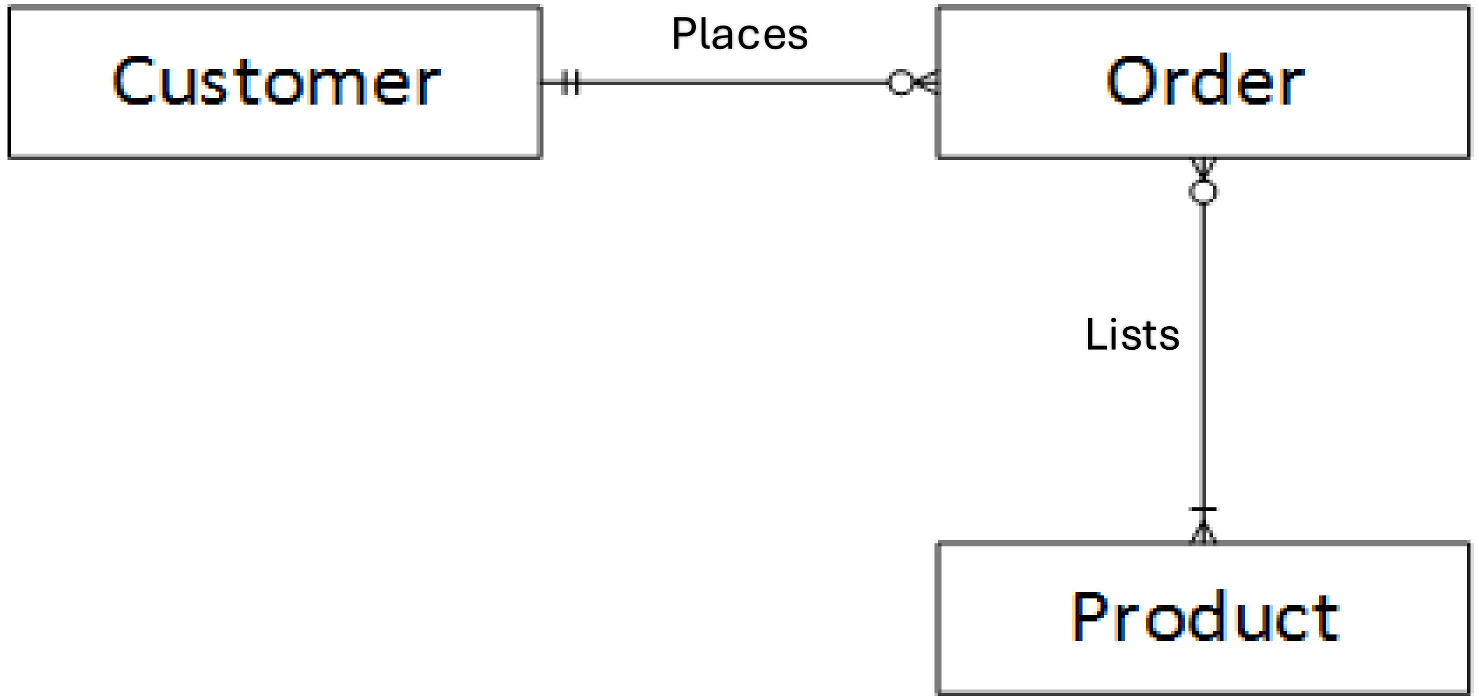
Various Notations



Crow's Foot Notation



Crow's Foot Notation: Example


Customer
Places
Order
Lists
Product