FUNDAMENTALS OF
DATABASE
SYSTEMS

7TH Edition

ELMASRI • NAVATHE

CHAPTER 3

Data Modeling Using the Entity-Relationship (ER) Model

Chapter 3 Outline

- **Overview of Database Design Process**
- **Example Database Application (COMPANY)**
- **ER Model Concepts**
  - 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 CONSTRAINTS**
- **Relationships of Higher Degree**
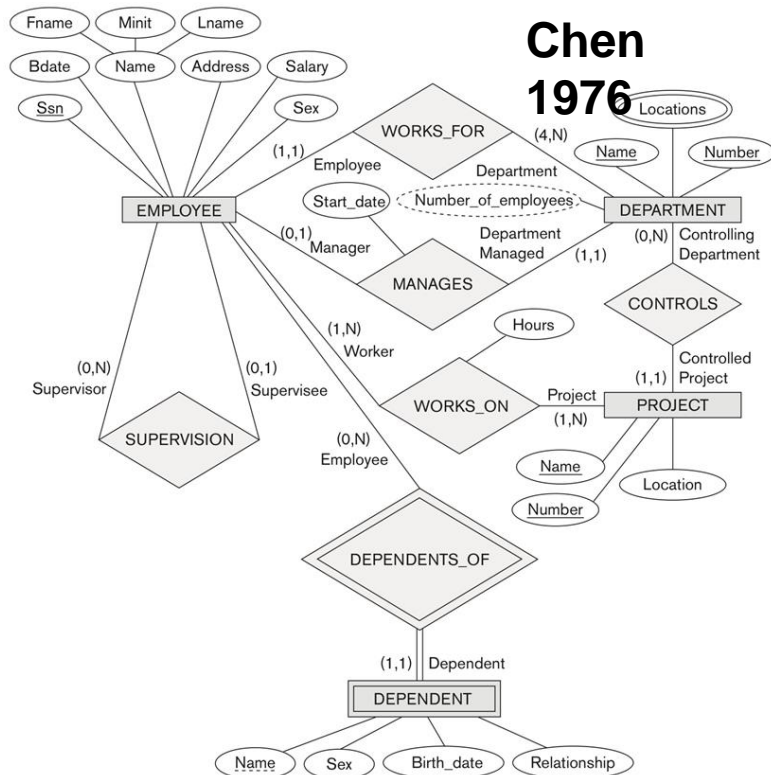
Peter Chen 1976

The Entity-Relationship Model—Toward a Unified View of Data

PETER PIN-SHAN CHEN

Massachusetts Institute of Technology

A data model, called the entity-relationship model, is proposed. This model incorporates some of the important semantic information about the real world. A special diagrammatic technique is introduced as a tool for database design. An example of database design and description using the model and the diagrammatic technique is given. Some implications for data integrity, information retrieval, and data manipulation are discussed.

The entity-relationship model can be used as a basis for unification of different views of data: the network model, the relational model, and the entity set model. Semantic ambiguities in these models are analyzed. Possible ways to derive their views of data from the entity-relationship model are presented.

Key Words and Phrases: database design, logical view of data, semantics of data, data models, entity-relationship model, relational model, Data Base Task Group, network model, entity set model, data definition and manipulation, data integrity and consistency
CR Categories: 3.50, 3.70, 4.33, 4.34

1. INTRODUCTION

The logical view of data has been an important issue in recent years. Three major data models have been proposed: the network model [2, 3, 7], the relational model [8], and the entity set model [25]. These models have their own strengths and weaknesses. The network model provides a more natural view of data by separating entities and relationships (to a certain extent), but its capability to achieve data independence has been challenged [8]. The relational model is based on relational theory and can achieve a high degree of data independence, but it may lose some important semantic information about the real world [12, 15, 23]. The entity set model, which is based on set theory, also achieves a high degree of data independence, but its viewing of values such as "3" or "red" may not be natural to some people [25].

This paper presents the entity-relationship model, which has most of the advantages of the above three models. The entity-relationship model adopts the

Overview of Database Design Process

- Two main activities:
  - Database design
  - Applications design
- Focus in this chapter is on <u>conceptual database design</u>
  - To design the conceptual schema for a database application
- Applications design focuses on the programs and interfaces that access the database
  - Generally considered part of software engineering
- What you learnt already:
  - The relational model used to store a database
  - The SQL language and constraints that are needed to define the database

Overview of Database Design Process
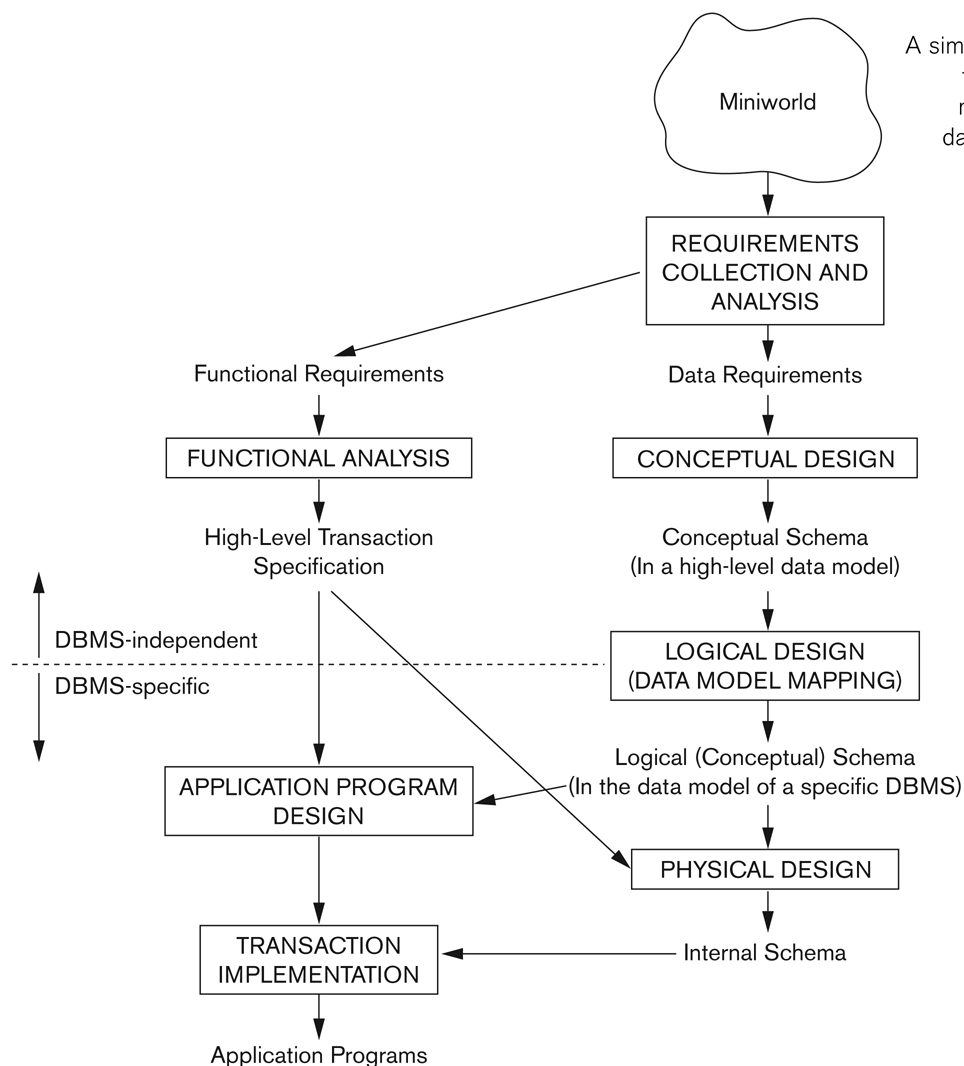
Figure 3.1
A simplified diagram to illustrate the main phases of database design.

Methodologies for Conceptual Design
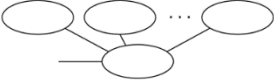
- Entity Relationship (ER) Diagrams (This Chapter)
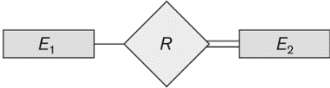- Enhanced Entity Relationship (EER) Diagrams (Chapter 4) – (deferred to Lecture 6).

What is happening in practice in Industry and Business?

- Use of Design Tools is common for designing and documenting large scale designs. Each DBMS vendor has their own suite of tools.
- The UML (Unified Modeling Language) Class Diagrams are popular in industry to document conceptual database designs.
- Tools such as Rational Rose are heavily in use.

NOTATION for ER diagrams

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
R — (min, max) — E	Structural Constraint (min, max) on Participation of E in R

Example : COMPANY Database (abbreviated)
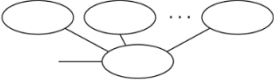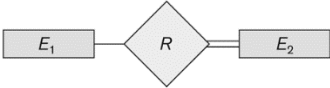
- We need to create a database schema design based on the following (simplified) **requirements** of the COMPANY Database:

    - The company is organized into DEPARTMENTs. Each department has a name, number and an 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 (Continued)

- The database will store each EMPLOYEE's social security number, address, salary, sex, and birthdate.
  - Each employee *works for* one department but may *work on* several projects.
  - The DB will keep track of the number of hours per week that an employee currently works on each project.
  - It is required to keep track of the *direct supervisor* of each employee.
- Each employee may *have* a number of DEPENDENTs.
  - For each dependent, the DB keeps a record of name, sex, birthdate, and relationship to the employee.

NOTATION for ER diagrams

Figure 3.14
Summary of the
notation for ER
diagrams.

Symbol	Meaning
▭	Entity
▭ (double)	Weak Entity
◇	Relationship
◇ (double)	Indentifying Relationship
⬭	Attribute
⬭ (underlined)	Key Attribute
⬭ (double)	Multivalued Attribute
composite	Composite Attribute
⬭ (dashed)	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

R (min, max) E Structural Constraint (min, max) on Participation of E in R

Initial Conceptual Design of Entity Types for the COMPANY Database Schema

- Based on the requirements, we can identify four initial entity types in the COMPANY database:
  - DEPARTMENT
  - PROJECT
  - EMPLOYEE
  - DEPENDENT
- Their initial conceptual design is shown on the following slide
- The initial attributes shown are derived from the requirements description

Initial Design of Entity Types:
EMPLOYEE, DEPARTMENT, PROJECT, DEPENDENT
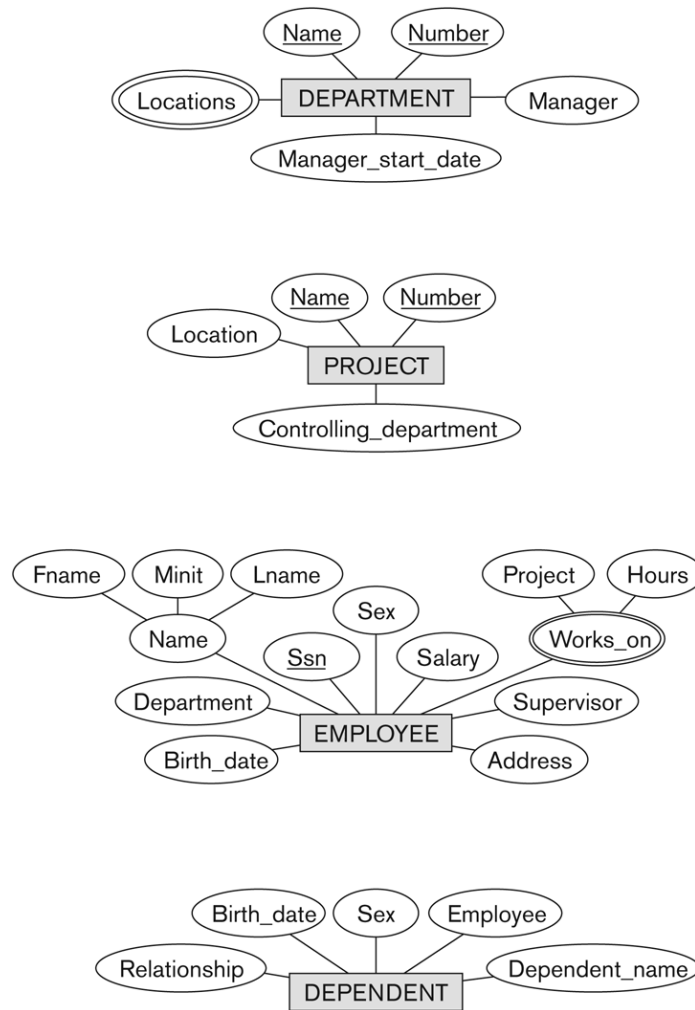
Figure 3.8
Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

ER Model Concepts

- **Entities and Attributes**
  - Entity is a basic concept for the ER model. Entities are specific things or objects in the mini-world that are represented in the database.
    - For example the EMPLOYEE John Smith, the Research DEPARTMENT, the ProductX PROJECT
  - Attributes are properties used to describe an entity.
    - For example an EMPLOYEE entity may have the attributes 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, date, enumerated type, …

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).
    - Composition may form a hierarchy where some components are themselves composite.
- ## 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}.

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)}
    - Multiple PreviousDegrees values can exist
    - Each has four subcomponent attributes:
        - College, Year, Degree, Field

Example of a composite attribute
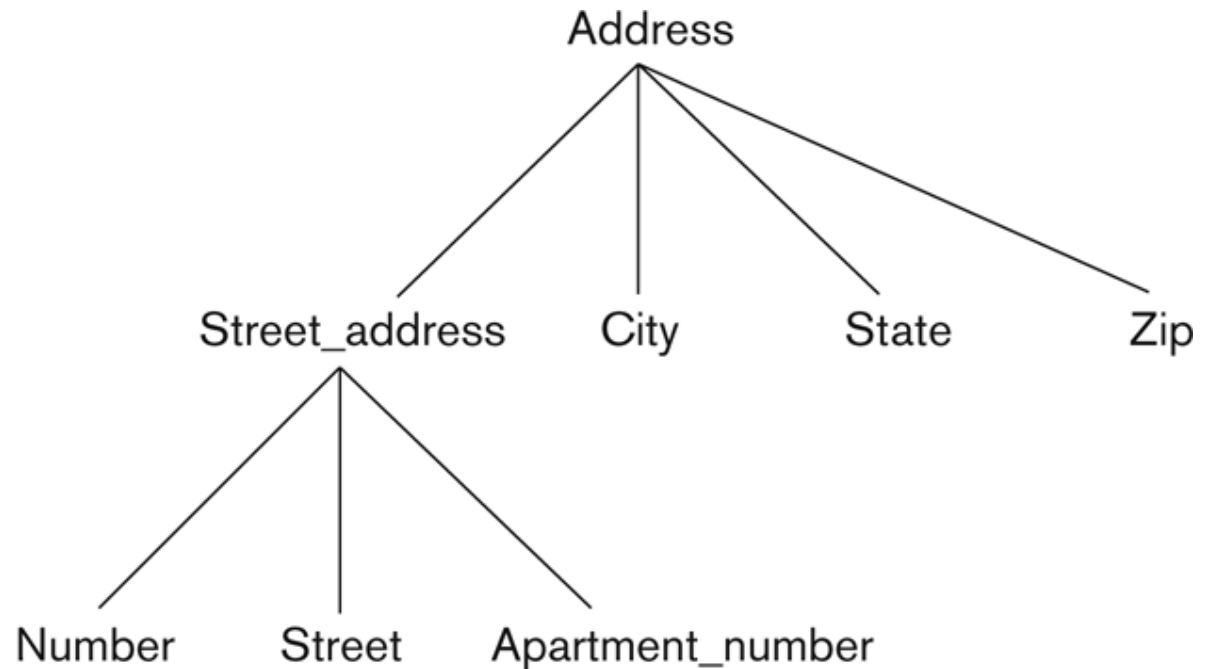

Figure 3.4
A hierarchy of composite attributes.

Entity Types and Key Attributes (1)

- Entities with the same basic attributes are grouped or typed into an entity type.
  - For example, the entity type EMPLOYEE and PROJECT.
- An attribute of an entity type for which each entity must have a unique value is called a key attribute of the entity type.
  - For example, SSN of EMPLOYEE.

Entity Types and Key Attributes (2)

- A key attribute may be composite.
  - VehicleTagNumber is a key of the CAR entity type with components (Number, State).
- An entity type may have more than one key.
  - The CAR entity type may have two keys:
    - VehicleIdentificationNumber (popularly called VIN)
    - VehicleTagNumber (Number, State), aka license plate number.
- Each key is underlined in ER schema (Note: this is different from the relational schema where only one "primary key" is underlined).

Entity Type CAR with two keys and a corresponding Entity Set

(a)

State   Number
Registration   Vehicle_id
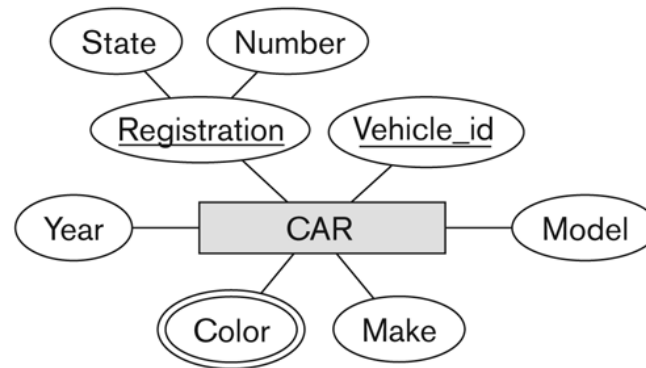Year   CAR   Model
Color   Make

Figure 3.7
The CAR entity type with two key attributes, Registration and Vehicle_id. (a) ER diagram notation. (b) Entity set with three entities.

(b)

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

CAR₁
((ABC 123, TEXAS), TK629, Ford Mustang, convertible, 2004 {red, black})

CAR₂
((ABC 123, NEW YORK), WP9872, Nissan Maxima, 4-door, 2005, {blue})

CAR₃
((VSY 720, TEXAS), TD729, Chrysler LeBaron, 4-door, 2002, {white, blue})

Entity Set

- Each entity type will have a collection of entities stored in the database
  - Called the **entity set** or sometimes **entity collection**
- Previous slide shows three CAR entity instances in the entity set for CAR
- Same name (CAR) used to refer to both the entity type and the entity set
- However, entity type and entity set may be given different names
- Entity set is the current *state* of the entities of that type that are stored in the database

Value Sets (Domains) of Attributes

- Each simple attribute is associated with a value set
  - E.g., Lastname has a value which is a character string of upto 15 characters, say
  - Date has a value consisting of MM-DD-YYYY where each letter is an integer
- A **value set** specifies the set of values associated with an attribute
- In SQL we capture it in terms of a data type.

Attributes and Value Sets

- Value sets are similar to data types in most programming languages – e.g., integer, character (n), real, bit

- Mathematically, an attribute A for an entity type E whose value set is V is defined as a function

$$A : E \rightarrow P(V)$$

Where P(V) indicates a power set (which means all possible subsets) of V. The above definition covers simple and multivalued attributes.[see Chen, 1976 for further details].
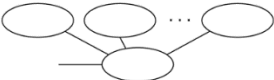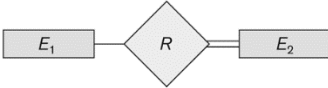
- We refer to the value of attribute A for entity e as A(e).

Displaying an Entity type

- In ER diagrams, an entity type is displayed in a rectangular box
- Attributes are displayed in ovals
    - Each attribute is connected to its entity type
    - Components of a composite attribute are connected to the oval representing the composite attribute
    - Each key attribute is underlined
    - Multivalued attributes displayed in double ovals
- See the full ER notation in advance on the next slide

NOTATION for ER diagrams

Figure 3.14 Summary of the notation for ER diagrams.

Symbol	Meaning
▭	Entity
▭ (double)	Weak Entity
◇	Relationship
◈	Indentifying Relationship
⬭	Attribute
⬭ (underlined)	Key Attribute
⬭ (double)	Multivalued Attribute
⬭...⬭	Composite Attribute
⬭ (dashed)	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
R (min, max) E	Structural Constraint (min, max) on Participation of E in R

Refining the initial design by introducing **relationships**

- The initial design is typically not complete
- Some aspects in the requirements will be represented as **relationships**
- ER model has three main concepts:
  - Entities (and their entity types and entity sets)
  - Attributes (simple, composite, multivalued)
  - Relationships (and their relationship types and relationship sets)
- We introduce relationship concepts next

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.

Relationship instances of the WORKS_FOR **N:1 (many-to-one)** relationship between EMPLOYEE and DEPARTMENT
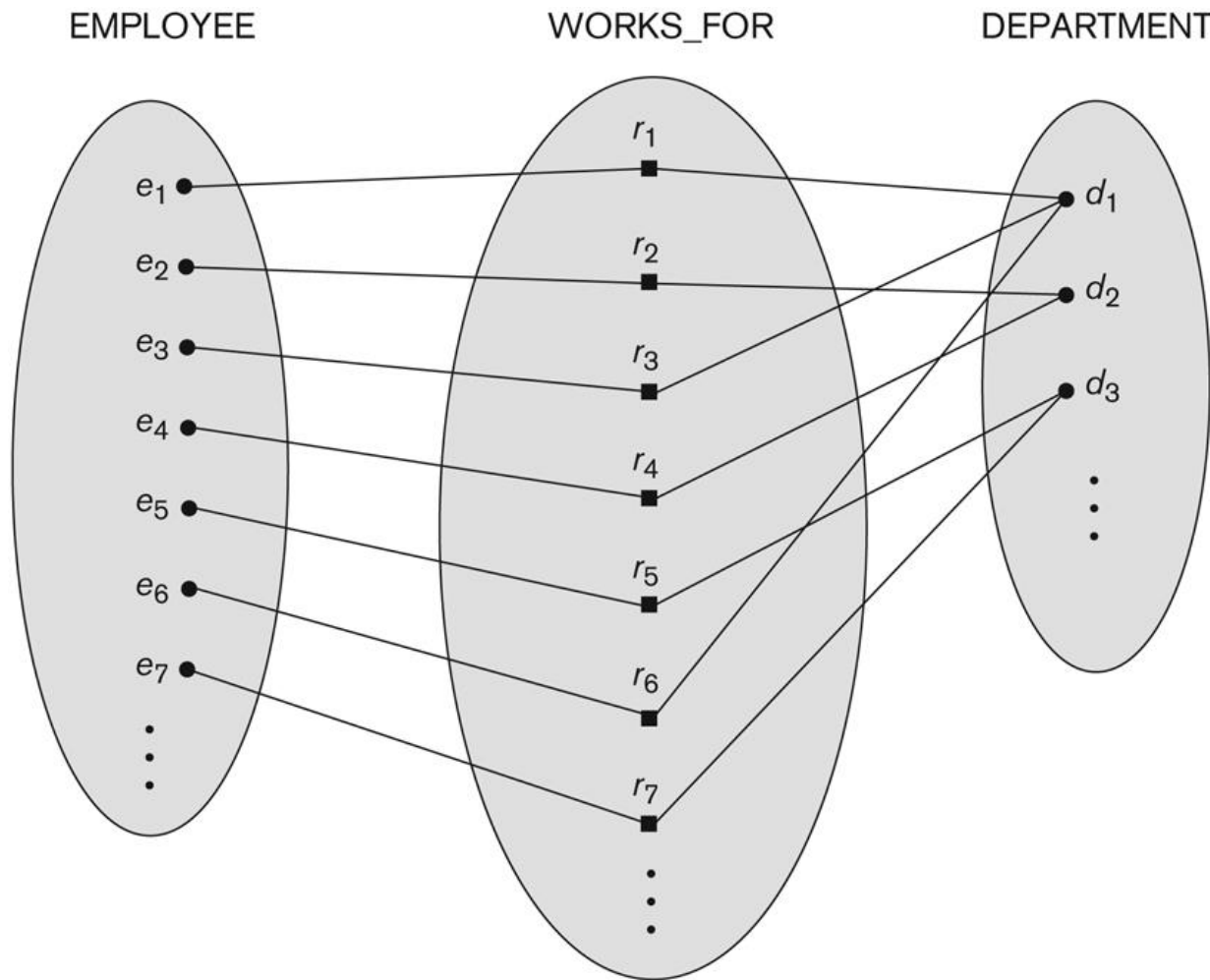

Figure 3.9
Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT.

Relationship instances of the **M:N (many-to-many)** WORKS_ON relationship between EMPLOYEE and PROJECT

Figure 3.13
An M:N relationship, WORKS_ON.
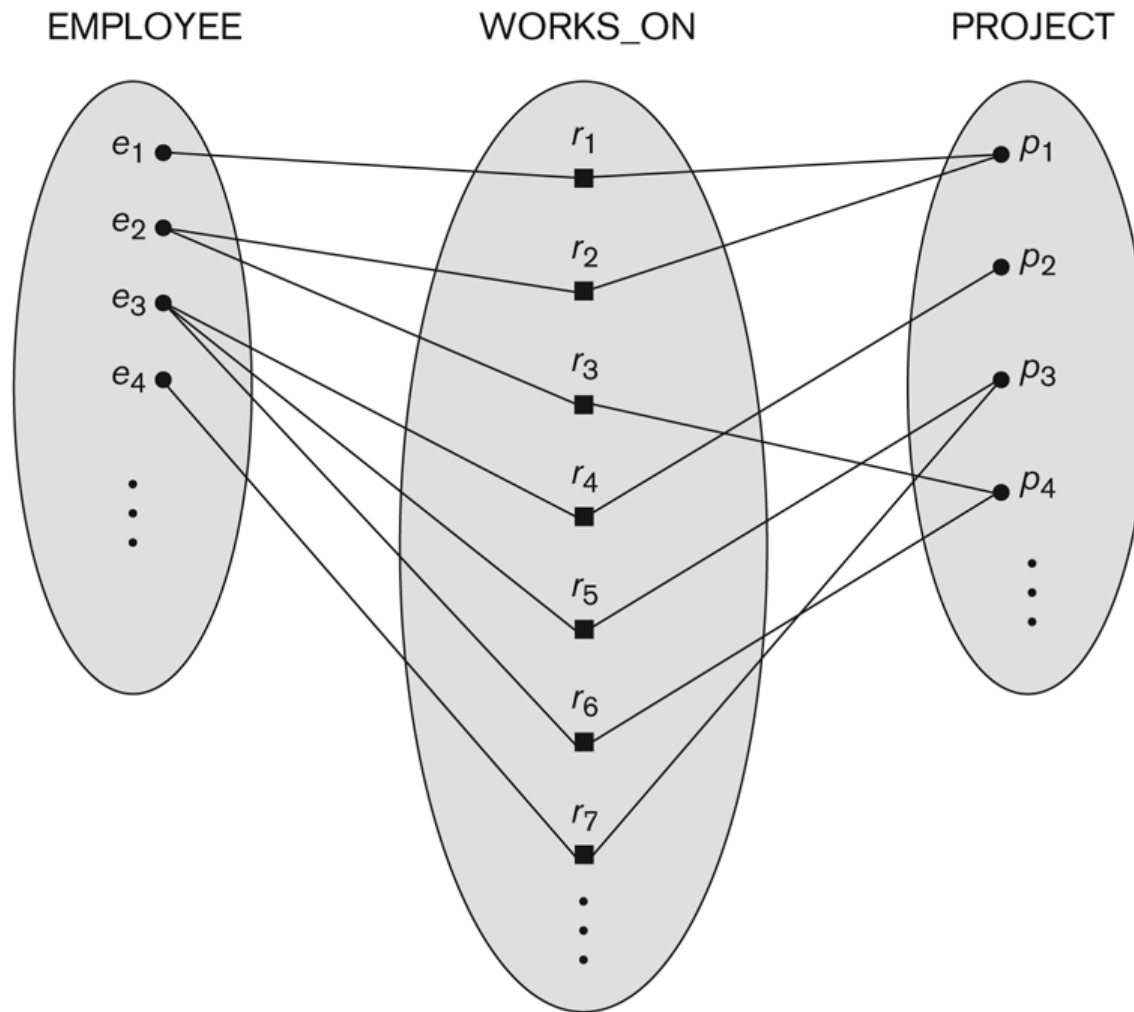
Relationship type vs. relationship set (1)

- Relationship Type:
    - Is the schema description of a relationship
    - Identifies the relationship name and the participating entity types
    - Also identifies certain relationship constraints
- Relationship Set:
    - The current set of relationship instances represented in the database
    - The current *state* of a relationship type

Relationship type vs. relationship set (2)

- Previous figures displayed the relationship sets
- Each instance in the set relates individual participating entities – one from each participating entity type
- In ER diagrams, we represent the *relationship type* as follows:
  - Diamond-shaped box is used to display a relationship type
  - Connected to the participating entity types via straight lines
  - Note that in ER modeling the relationship type is not shown with an arrow. **The names should be typically readable from left to right and top to bottom.**

Refining the COMPANY database schema by introducing relationships

- By examining the requirements, six relationship types are identified
- All are *binary* relationships(degree 2)
- Listed below with their participating entity types:
  - WORKS_FOR (between EMPLOYEE, DEPARTMENT)
  - MANAGES (also between EMPLOYEE, DEPARTMENT)
  - CONTROLS (between DEPARTMENT, PROJECT)
  - WORKS_ON (between EMPLOYEE, PROJECT)
  - SUPERVISION (between EMPLOYEE (as subordinate), EMPLOYEE (as supervisor))
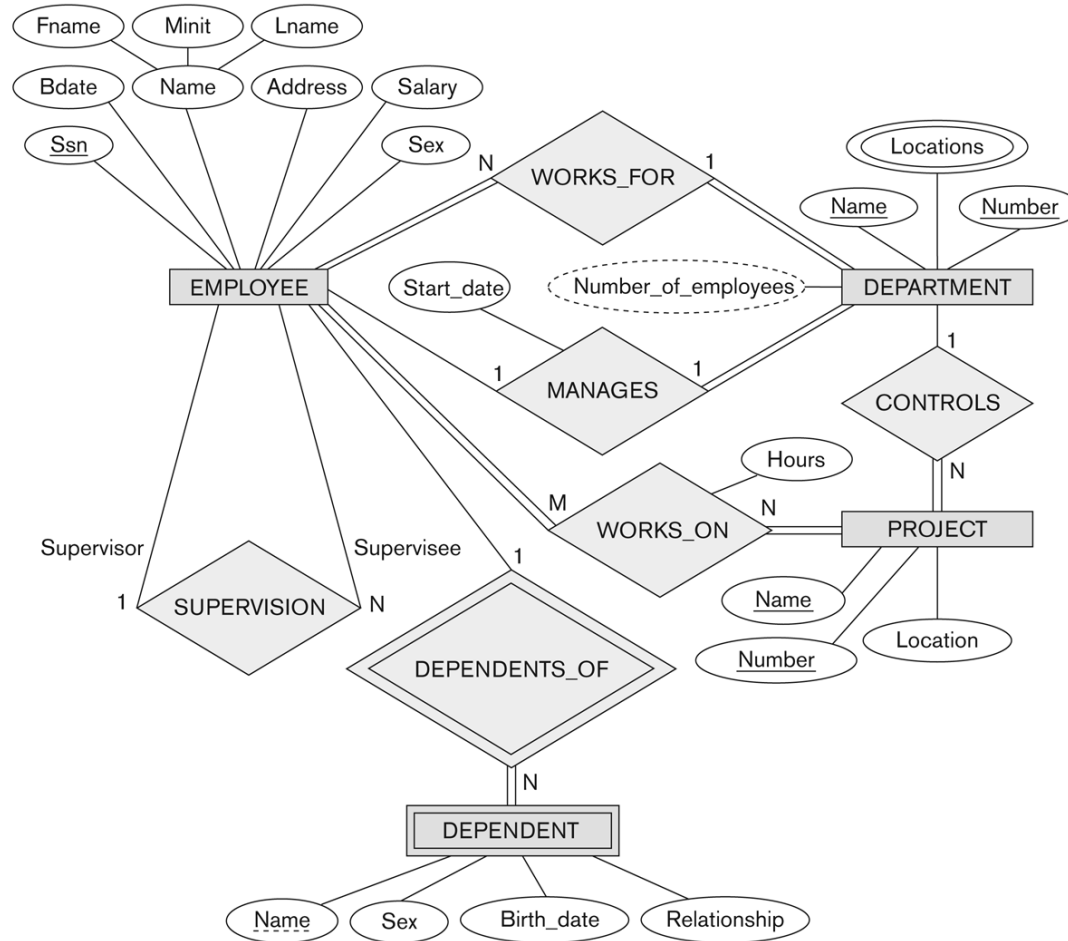 - DEPENDENTS_OF (between EMPLOYEE, DEPENDENT)

Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation
is introduced gradually throughout this chapter.

Constraints on Relationships

- Constraints on Relationship Types
  - (Also known as ratio constraints)
  - Cardinality Ratio Constraint (specifies *maximum* participation)
    - One-to-one (1:1)
    - One-to-many (1:N) or Many-to-one (N:1)
    - Many-to-many (M:N)
  - **Existence Dependency** Constraint (specifies *minimum* participation) (also called participation constraint)
    - zero (optional participation, not existence-dependent)
    - one or more (mandatory participation, existence-dependent)

Recursive Relationship Type

- A relationship type between the same participating entity type in **distinct roles**
- Also called a **self-referencing** relationship type.
- Example: the SUPERVISION relationship
- EMPLOYEE participates twice in two distinct roles:
  - supervisor (or boss) role
  - supervisee (or subordinate) role
- Each relationship instance relates two distinct EMPLOYEE entities:
  - One employee in *supervisor* role
  - One employee in *supervisee* role

Displaying a recursive relationship

- In a recursive relationship type.
  - Both participations are the 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.
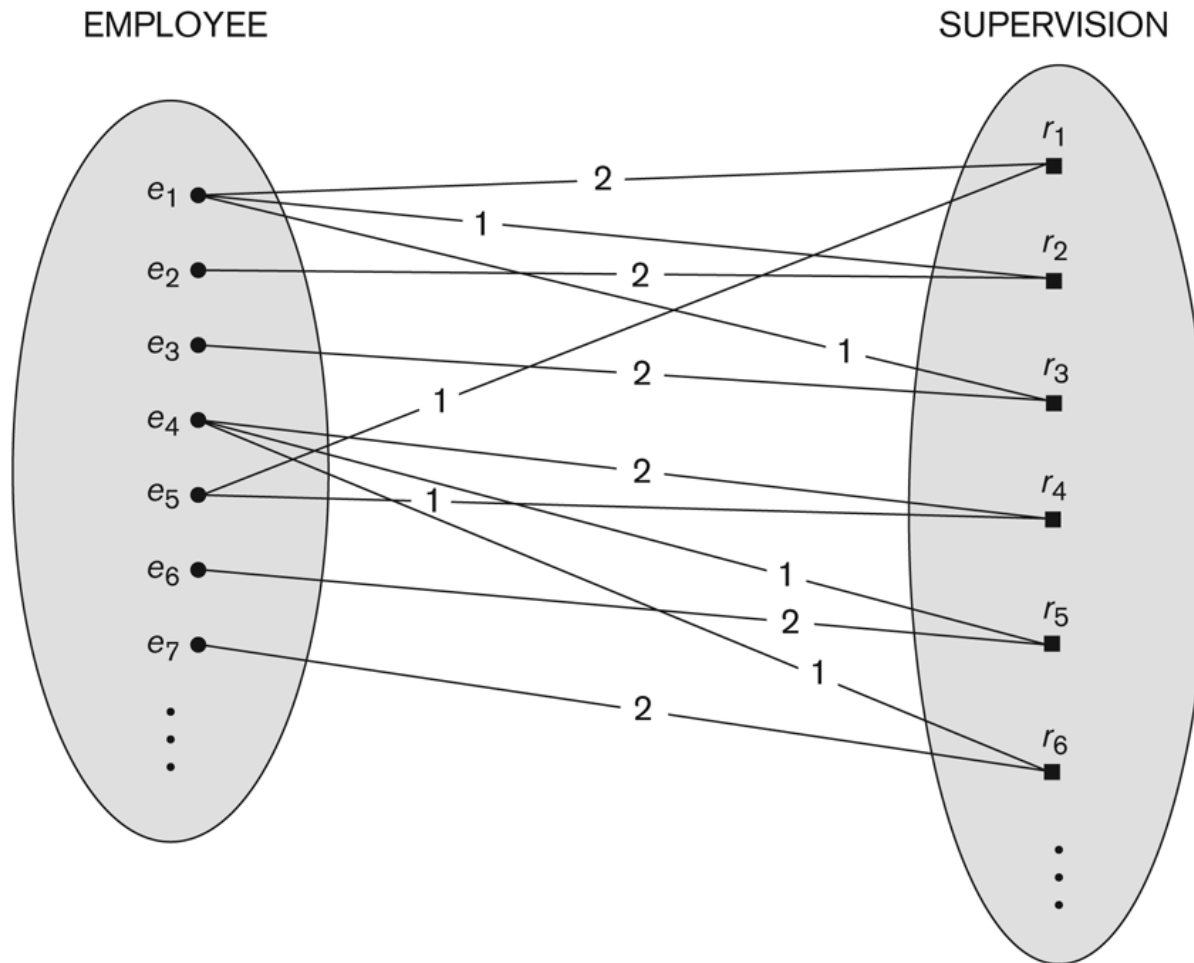
A Recursive Relationship Supervision`

Figure 3.11
A recursive relationship SUPERVISION between EMPLOYEE in the *supervisor* role (1) and EMPLOYEE in the *subordinate* role (2).
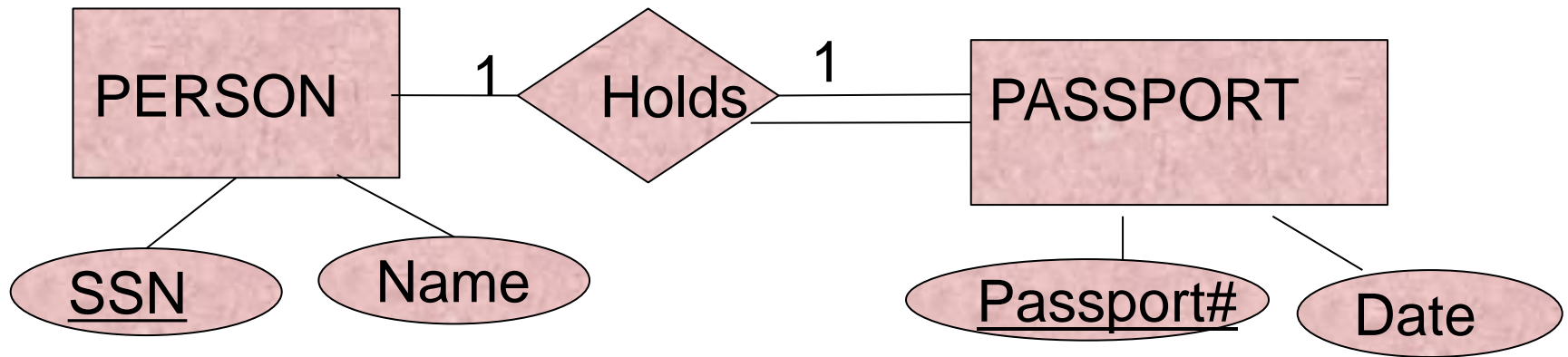
Weak Entity Types

- An entity that does not have a key attribute and that is identification-dependent on another entity type.
- 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
  - The particular entity they are related to in the identifying relationship type
- **Example:**
  - A DEPENDENT entity is identified by the dependent's first name, *and* the specific EMPLOYEE with whom the dependent is related
  - Name of DEPENDENT is the *partial key*
  - DEPENDENT is a *weak entity type*
  - EMPLOYEE is its identifying entity type via the identifying relationship type DEPENDENT_OF

Two kinds of dependencies among entity types

- Existence Dependency (or minimum participation constraint): E.G., Between two entity types PERSON and PASSPORT, Passport entity cannot exist in the database unless the Person entity that owns the passport is present.

- Identification Dependency between BUILDING and ROOM.

- Room cannot exist unless corresponding Building is present (existence dependency)

- Room can be fully identified only if we know what Building it is in (identification dependency)

Just Existence Dependency – no Weak Entity type

- NOTICE that PERSON and PASSPORT each have their own identifiers: SSN and Passport#

- PASSPORT is Existence-dependent on PERSON

- PASSPORT is NOT Identification-dependent on PERSON

Identification Dependency leads to Weak Entity type
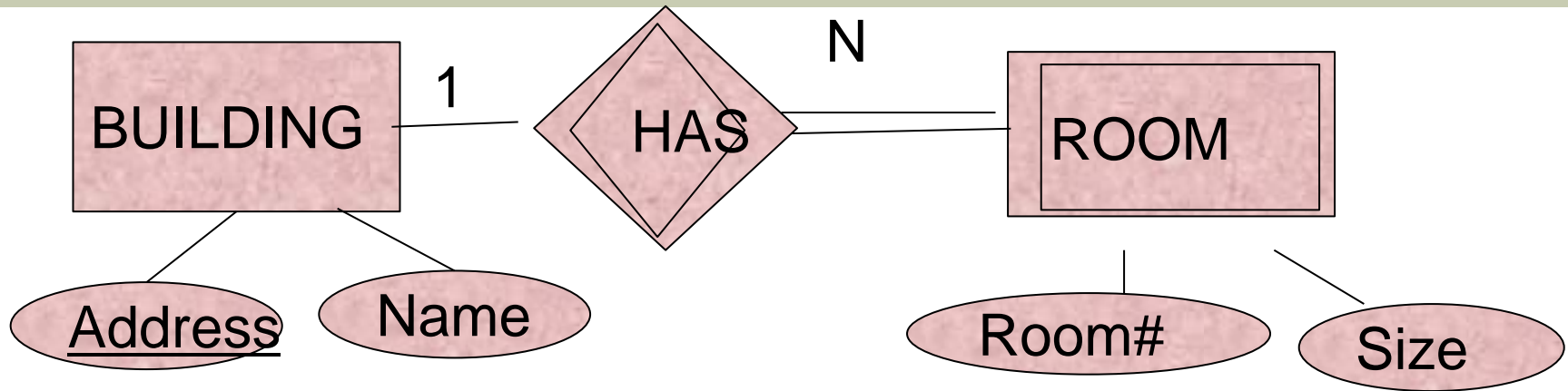
- Identification Dependency (implies weak entity type).
- Identification dependency implies existence dependency as a natural phenomenon.
- Note: the "local identifier" Room# should have a dotted underline to indicate its partial role in identification (not shown here due to diagramming problem).

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.
        - A value of HoursPerWeek depends on a particular (employee, project) combination
    - Most relationship attributes are used with M:N relationships
        - In 1:N relationships, they can be transferred to the entity type on the N-side of the relationship

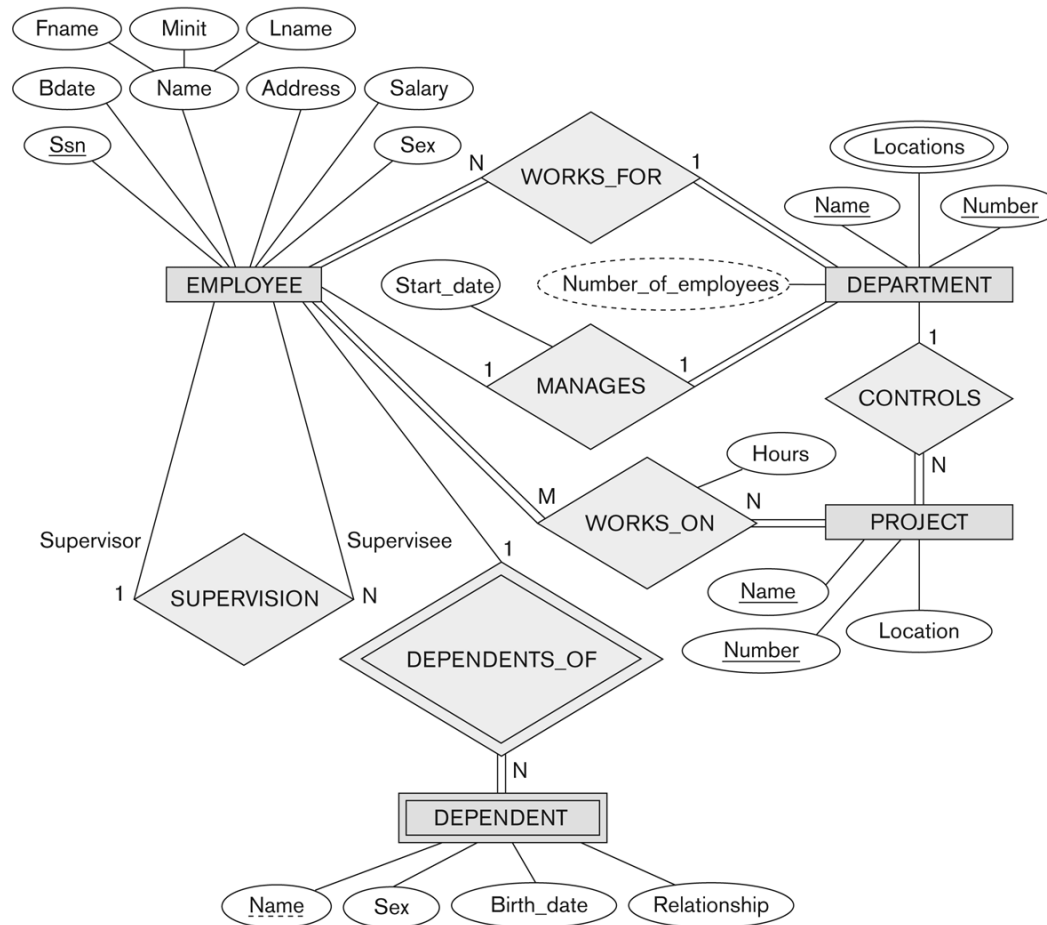# Example Attribute of a Relationship Type: Hours of WORKS_ON

Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

Notation for Constraints on Relationships

- Cardinality ratio (of a binary relationship): 1:1, 1:N, N:1, or M:N
  - Shown by placing appropriate numbers on the relationship edges.
- Participation constraint (on each participating entity type): total (called existence dependency) or partial.
  - Total shown by double line, partial by single line.
- NOTE: These are easy to specify for Binary Relationship Types.

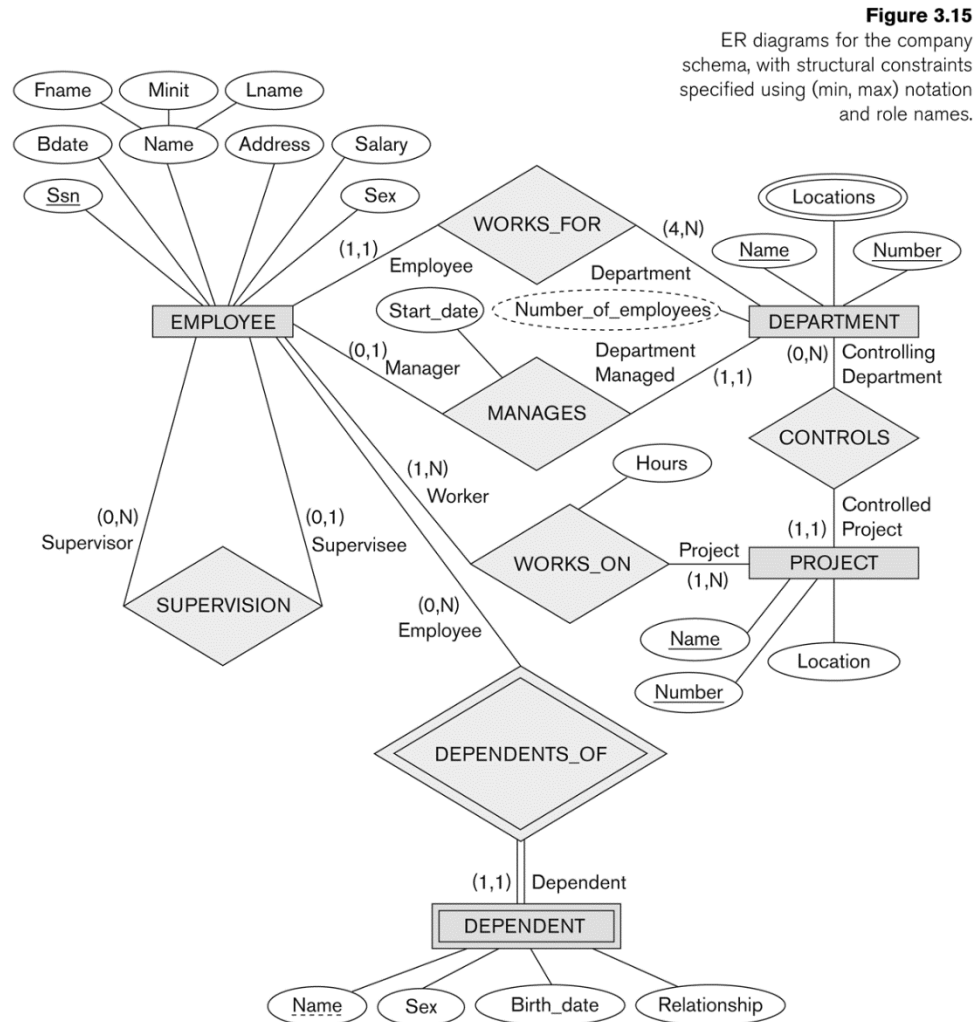# Alternative (min, max) notation for relationship structural constraints:

- Specified on each participation of an entity type E in a relationship type R
- Specifies that each entity e in E participates in at least *min* and at most *max* relationship instances in R
- Default: (no constraint): min=0, max=n (signifying no limit)
- Must have min≤max, min≥0, max ≥1
- Derived from the knowledge of mini-world constraints
- Examples:
  - A department has exactly one manager and an employee can manage at most one department.
    - Specify (0,1) for participation of EMPLOYEE in MANAGES
    - Specify (1,1) for participation of DEPARTMENT in MANAGES
  - An employee can work for exactly one department but a department can have any number of employees.
    - Specify (1,1) for participation of EMPLOYEE in WORKS_FOR
    - Specify (0,n) for participation of DEPARTMENT in WORKS_FOR

The (min,max) notation for relationship constraints
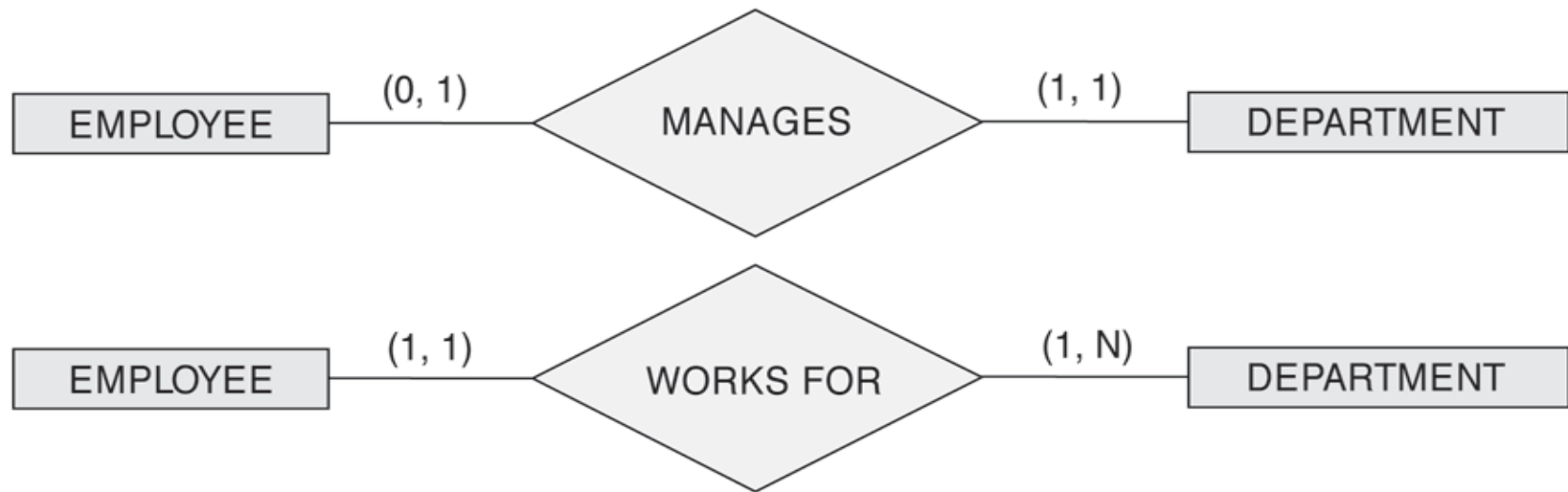
- Read the min,max numbers next to the entity type and looking **away from** the entity type

- E.g., in the upper schema, an Employee entity may participate in 0 (or no) instances of the manages relationship, but at a maximum can participate in 1 (one) instance of the Manages relationship. Hence it is shown as (0,1).

COMPANY ER Schema Diagram using (min, max) notation

Figure 3.15
ER diagrams for the company schema, with structural constraints specified using (min, max) notation and role names.

Relationships of Higher Degree

- Relationship types of degree 2 are called binary
- Relationship types of degree 3 are called ternary
- Relationships of degree n are called n-ary
- In general, an n-ary relationship is not equivalent to n binary relationships
- Constraints are harder to specify for higher-degree relationships (n > 2) than for binary relationships

Discussion of n-ary relationships (n > 2)

- In general, 3 binary relationships can represent different information than a single ternary relationship (see Figure 3.17a and b on next slide)

- If needed, the binary and n-ary relationships can all be included in the schema design (see Figure 3.17a and b, where all relationships convey different meanings)

- In some cases, a ternary relationship type can be represented as a weak entity type if the data model allows a weak entity type to have multiple identifying relationships (and hence multiple owner entity types) (see Figure 3.17c)

Example of a ternary relationship
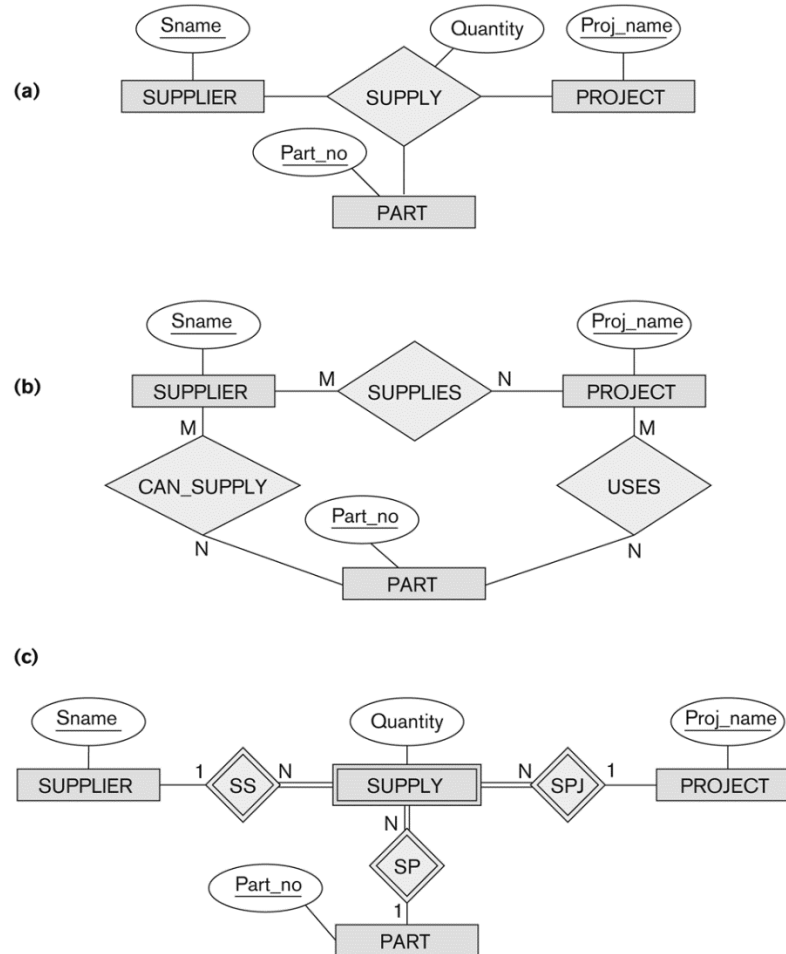
Figure 3.17
Ternary relationship types. (a) The SUPPLY relationship. (b) Three binary relationships not equivalent to SUPPLY. (c) SUPPLY represented as a weak entity type.

Meaning of a Ternary Relationship type

- **Implicit Constraint of the ER MODEL:**

In a ternary relationship type, every instance of the ternary relationship must have one instance of each of the three participating entity types.

- This rule extends to n-ary relationships where an n-ary relationship type is permissible only if every relationship instance represents a relationship involving one entity from each of the n participating entity types.

Ternary Relationship requires one of each participating entity types
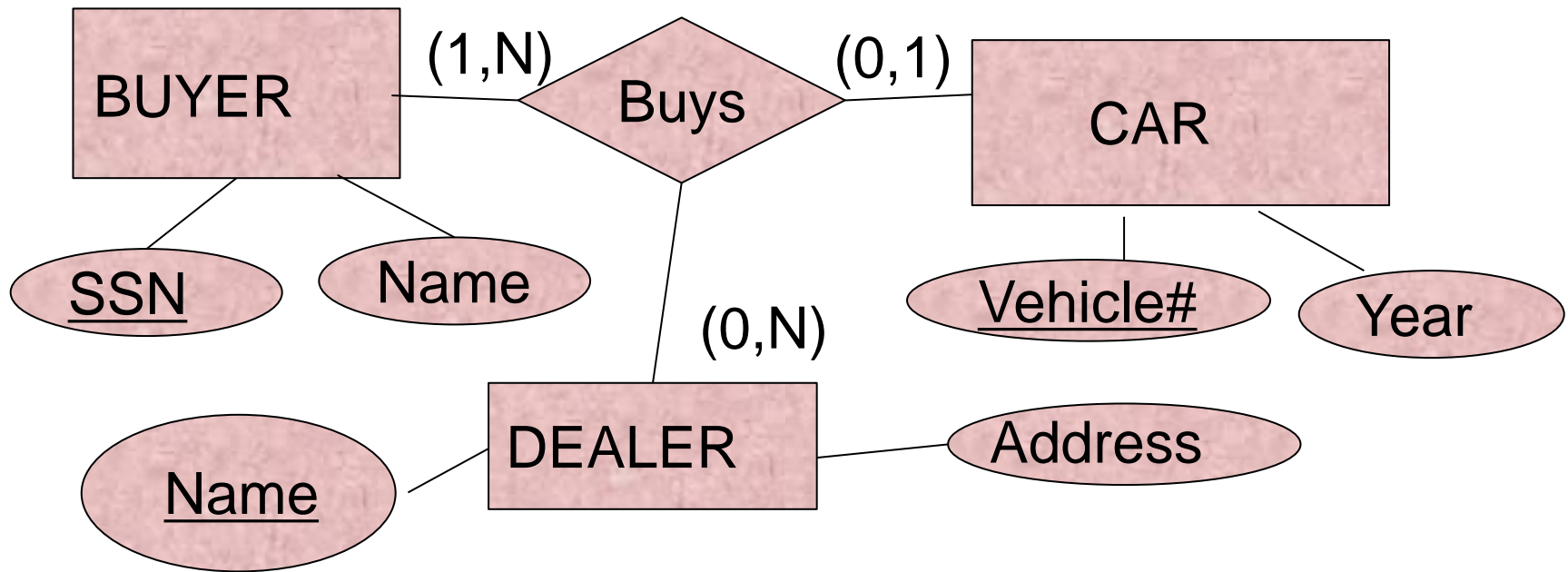
- Here, each instance of Buys relationship type relates one Buyer, one Car and one Dealer. Hence it is CORRECT.
- A Buyer buys at least one but may have bought any number of cars : (1,N)
- A Car in the database may not have been sold but can be sold only once: (0,1)

Ternary Relationship type must be used with utmost care
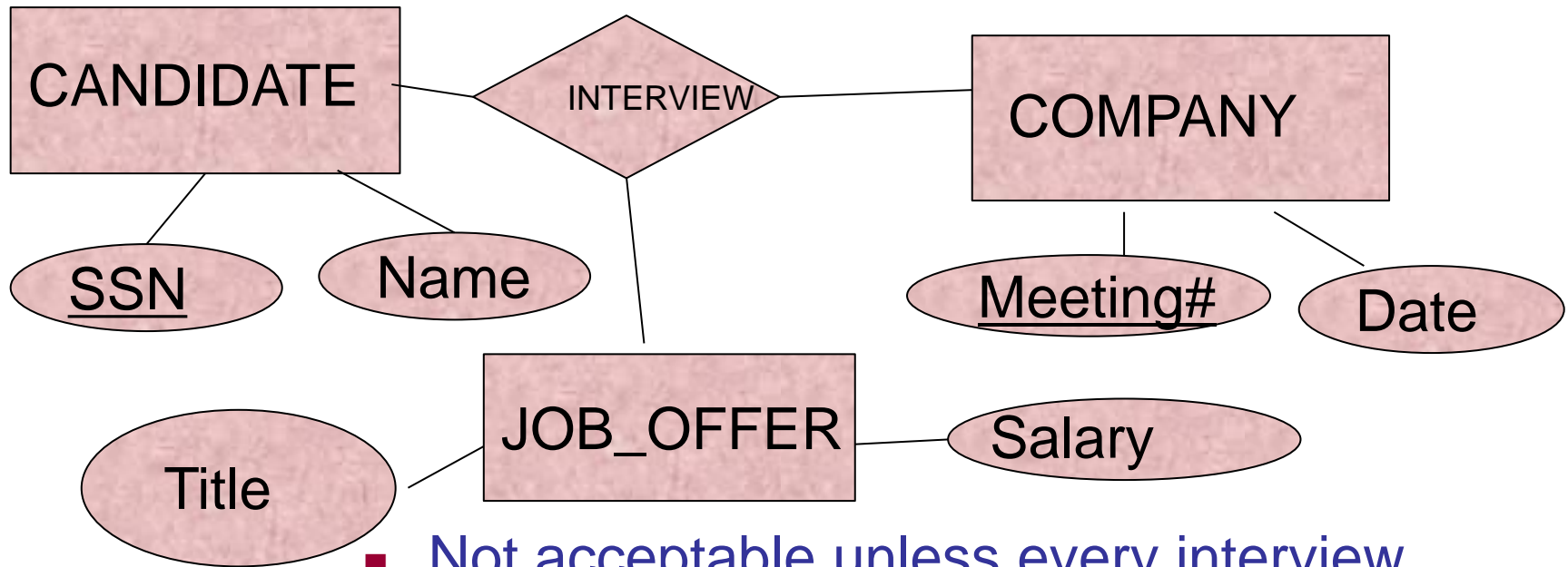
- Not acceptable unless every interview results in a Job Offer.

- For normal situation where only a few interviews result in job offers, above schema is INCORRECT.

Avoiding Ternaries for Easy Modeling
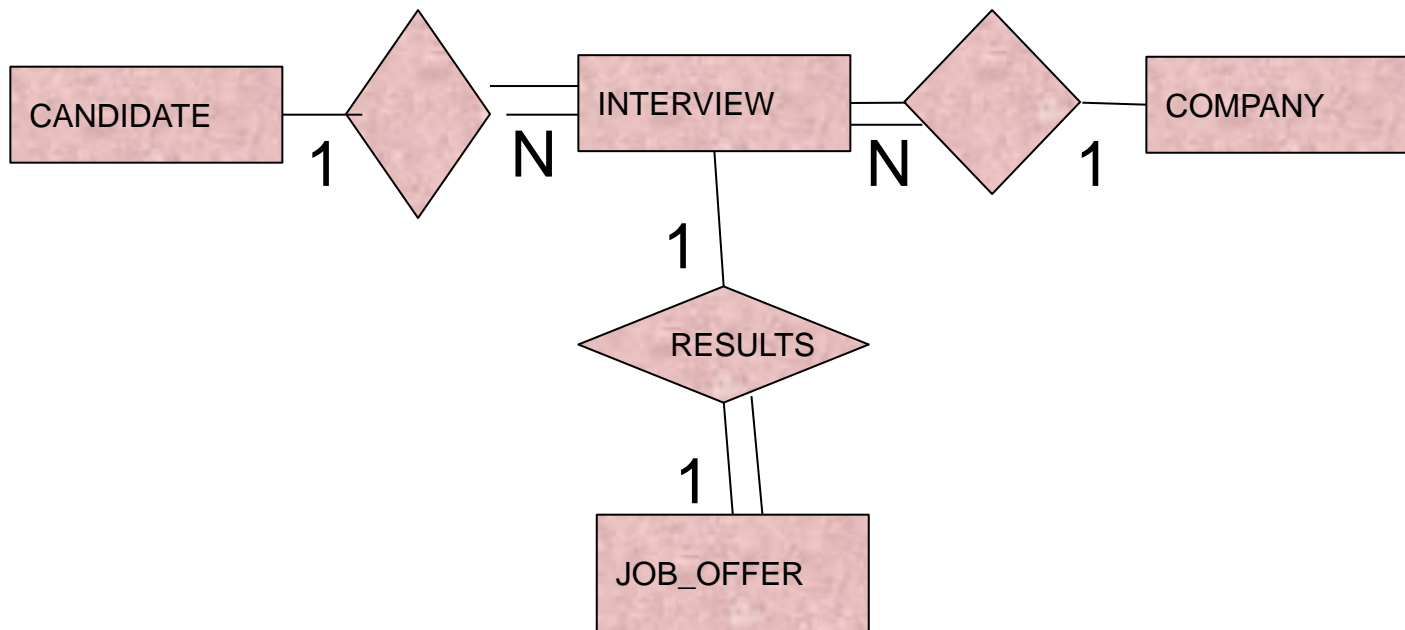
- The above schema is CORRECT and allows a better and convenient representation of the facts.

- We have the possibility of representing Interview as a Weak Entity type identified from Candiate and Company strong entity types if we prefer.

- This is known as "objectifying" the relationship type "Interview" into an entity type 'Interview".

Discussion of n-ary relationships (n > 2)

- If a particular binary relationship can be derived from a higher-degree relationship at all times, then it is redundant

- For example, the TAUGHT_DURING and OFFERED_DURING binary relationships in Figure 3.18 (see next slide) can be derived from the ternary relationship OFFERS (based on the meaning of the relationships). Hence they are redundant.

- However, the CAN_TEACH relationship type cannot be so inferred from OFFERS and hence is information-bearing and not redundant.

Another example of a ternary relationship

Figure 3.18
Another example of ternary versus binary relationship types.
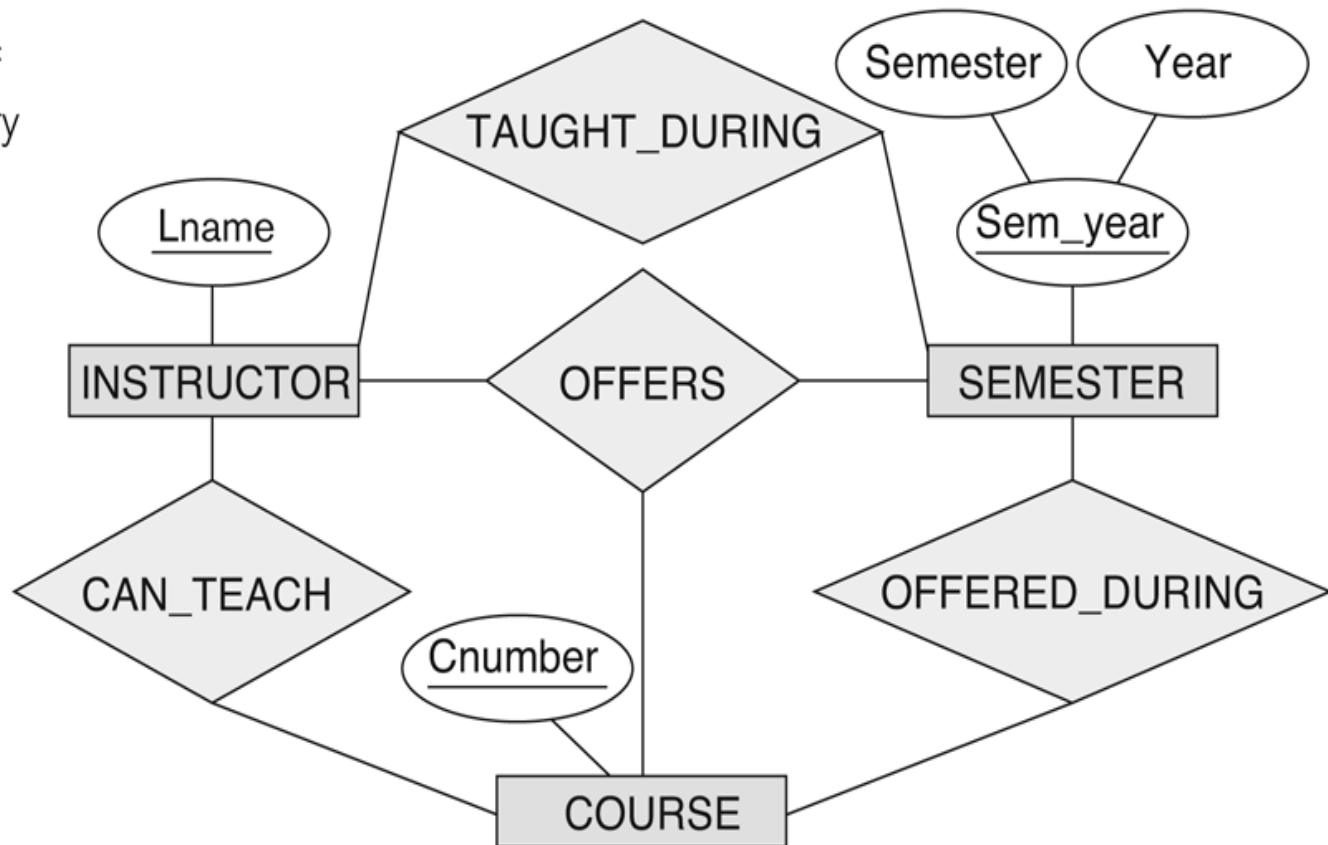
- The OFFERS ternary relationship type is CORRECT. However, TAUGHT_DURING and OFFERED_DURING are implied by OFFERS and hence are redundant and should be dropped.

Displaying constraints on higher-degree relationships

- The (min, max) constraints can be displayed on the edges – however, they do not fully describe the constraints between the pairs of entity types (three such pairs) when considered two at a time.

- Overall, the constraint specification is difficult and possibly ambiguous when we consider relationships of a degree higher than two.
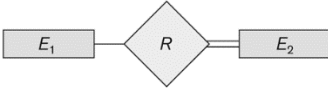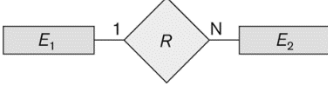
BOTTOM LINE:
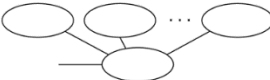
- Avoid using relationship types of degree higher than 2 as much as possible.

- Make sure that one instance of a ternary relationship must be related to one instance of each participating entity type.

- If a relationship type has further relationship types with other entities, make it into an entity type (objectifying the relationship type).

Alternative diagrammatic notation

- ER diagrams is one popular example for displaying database schemas

- Many other notations exist in the literature and in various database design and modeling tools

- Appendix A illustrates some of the alternative notations that have been used

- UML class diagrams is representative of another way of displaying ER concepts that is used in several commercial design tools

Summary of our notation for ER diagrams

Figure 3.14
Summary of the notation for ER diagrams.

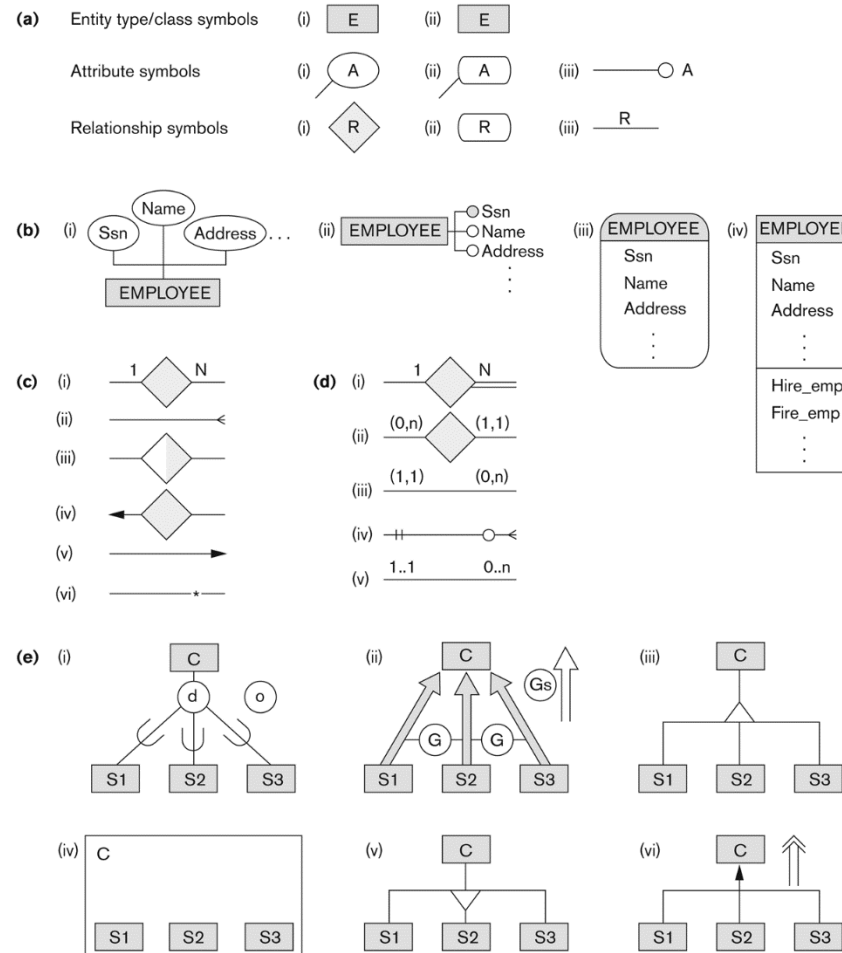| Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Indentifying Relationship
	Attribute
	Key Attribute
	Multivalued Attribute
	Composite Attribute
	Derived Attribute
E_1 — R — E_2	Total Participation of E_2 in R
E_1 —1 R N— E_2	Cardinality Ratio 1: N for E_1:E_2 in R
R — (min, max) E	Structural Constraint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

Other alternative diagrammatic notations

Figure A.1
Alternative notations. (a) Symbols for entity type/class, attribute, and relationship. (b) Displaying attributes. (c) Displaying cardinality ratios. (d) Various (min, max) notations. (e) Notations for displaying specialization/generalization.

UML class diagram for COMPANY database schema


Figure 3.16
The COMPANY conceptual schema in UML class diagram notation.
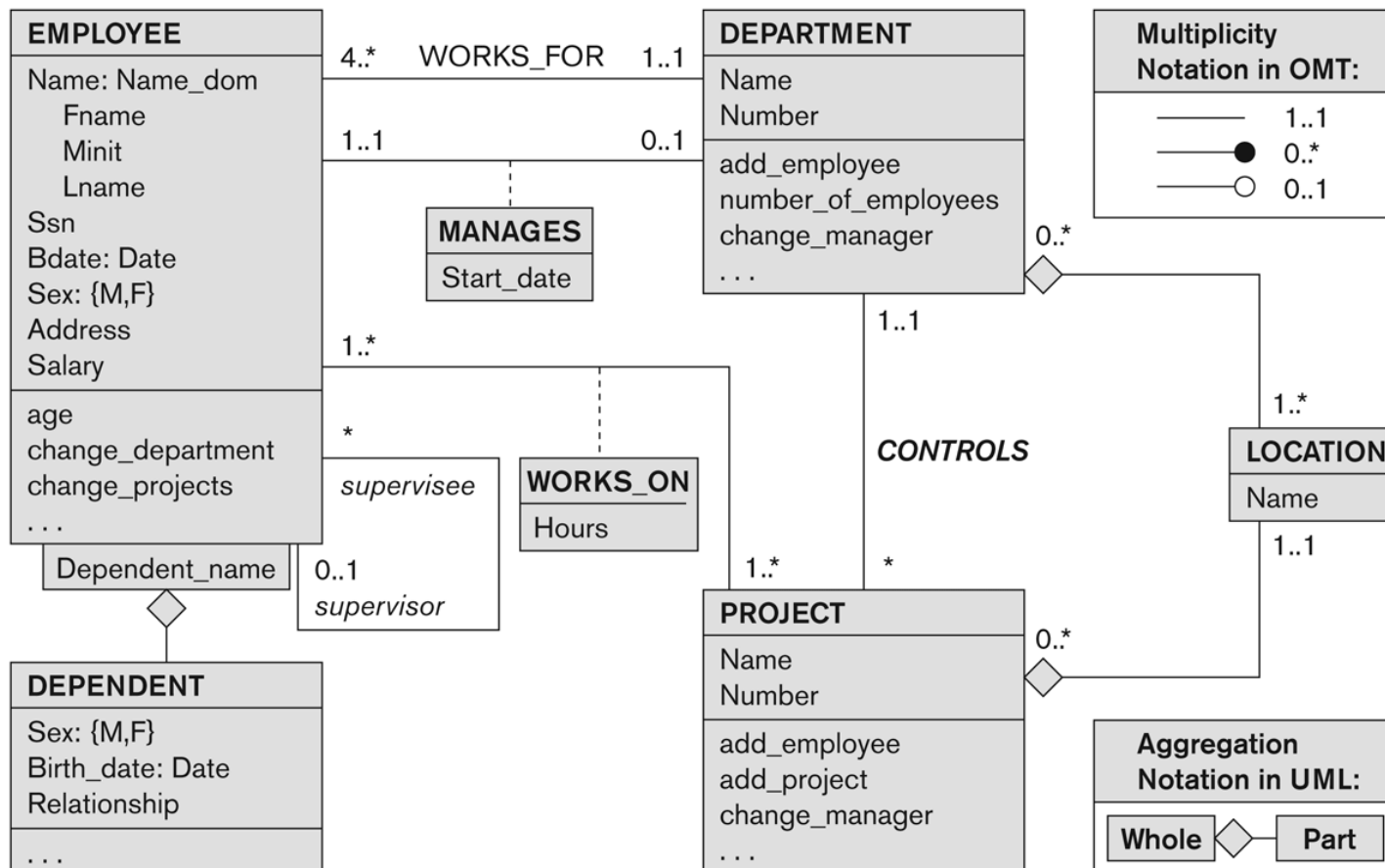
Another Example: A UNIVERSITY Database

- To keep track of the enrollments in classes and student grades, another database is to be designed.

- It keeps track of the COLLEGEs, DEPARTMENTs within each college, the COURSEs offered by departments, and SECTIONs of courses, INSTRUCTORs who teach the sections etc.

- These entity types and the relationships among these entity types are shown on the next slide in Figure 3.20.

UNIVERSITY database conceptual schema

©2016 Ramez Elmasri and Shamkant B. Navathe

Slide 3- 69
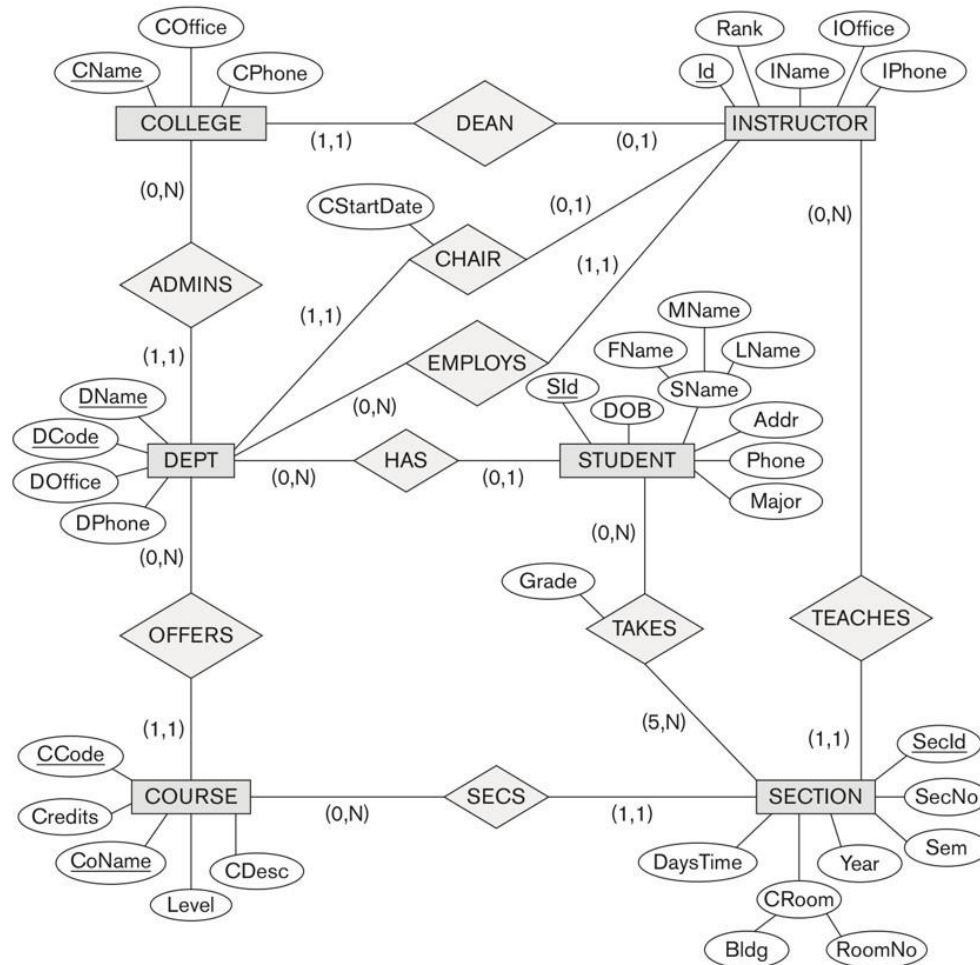
Chapter Summary

- ER Model Concepts: Entities, attributes, relationships

- Constraints in the ER model

- Using ER in step-by-step mode conceptual schema design for the COMPANY database

- ER Diagrams - Notation

- Binary Relationship types and those of higher degree.

- Alternative Notations – UML class diagrams, others