

# CHAPTER 3

## Data Modeling Using the Entity-Relationship (ER) Model

Note: Slides, content, web links and end chapter questions are prepared from Pearson textbook (Elmasri & Navathe), and other Internet resources.

# Topics of Discussion

- A. Overview of Database Design Process
- B. Example Database (COMPANY)
- C. ER Model Concepts
  - A. Entities and Attributes
  - B. Entity Types, Value Sets, and Key Attributes
  - C. Relationships and Relationship Types
  - D. Weak Entity Types
  - E. Roles and Attributes in Relationship Types
- D. ER Diagrams - Notation
- E. ER Diagram for COMPANY Schema
- F. Alternative Notations, UML diagrams, others
- G. Relationships of Higher Degree

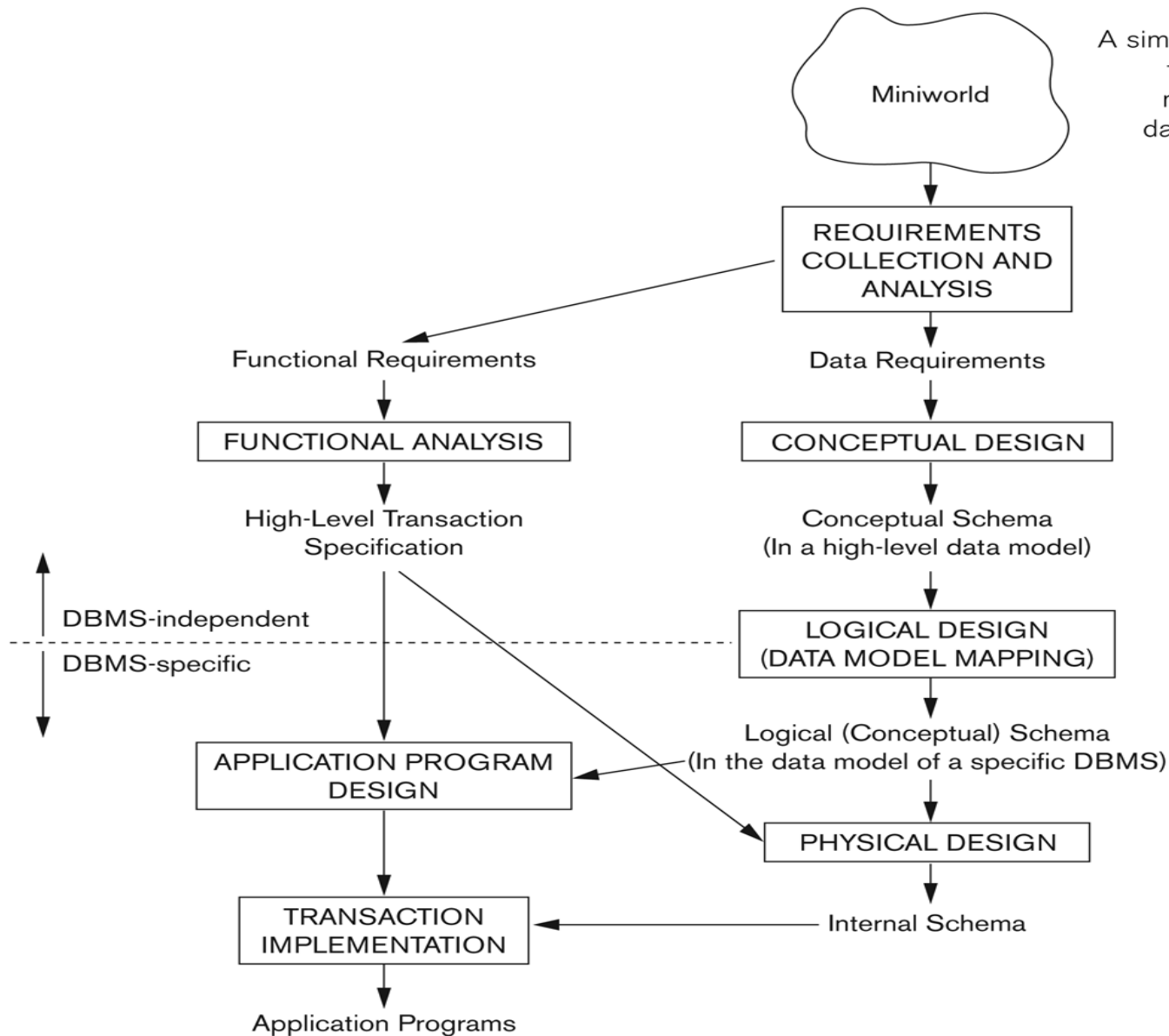
# Overview of Database Design Process

- Two main activities:
  - Database design
  - Applications design
- Focus in this chapter on database design (conceptual)
  - To design the conceptual schema for a database.
- Applications design focuses on the programs and interfaces that access the database
  - Generally considered part of “software engineering”.

# Overview of Database Design Process

**Figure 3.1**

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



# Overview of Database Design Process

- **Requirements collection and analysis:** During this step, the database designers interview prospective database users /clients to understand and document their data requirements.
- **Functional requirements:** In parallel with specifying the data requirements, to specify functional requirements. These consist of the operations (or transactions) that will be applied to the database. (We will not discuss these techniques here; they are usually covered in software engineering)
- **Conceptual design:** The conceptual schema is a concise description of the users' data requirements and includes detailed descriptions of the entity types, relationships, and constraints; these are expressed using the concepts provided by the high-level data model.

# Overview of Database Design Process

- **Logical design or data model mapping:** The actual implementation of the database, using a commercial DBMS Software. Most current commercial DBMSs use an implementation data model—such as the relational (SQL) model—so the conceptual schema is transformed from the high-level data model into the implementation data model.
- **Physical design:** during which the internal storage structures, file organizations, indexes, access paths, and physical design parameters for the database files are specified.

# Overview of Database Design Process

- We present basic ER (Entity Relationship) model concepts for conceptual schema design in this chapter.
  - Will present Enhanced Entity Relationship (EER) Diagrams (in Chapter 4)
- Use of Design Tools in the industry for designing and documenting large-scale designs. (Tools like ERWIN, Rational Rose, Oracle SQL Designer, etc..)
  - ERWIN from Erwin (now quest software)
  - SQL developer Data Modeler from Oracle
  - Rational Rose data modeler of IBM
  - MySQL Workbench
- The UML (Unified Modeling Language) Class Diagrams are also popular in the industry to document conceptual database designs.

# Example COMPANY Database

- We will 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

- 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 database 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 database keeps a record of name, sex, birthdate, and relationship to the employee.

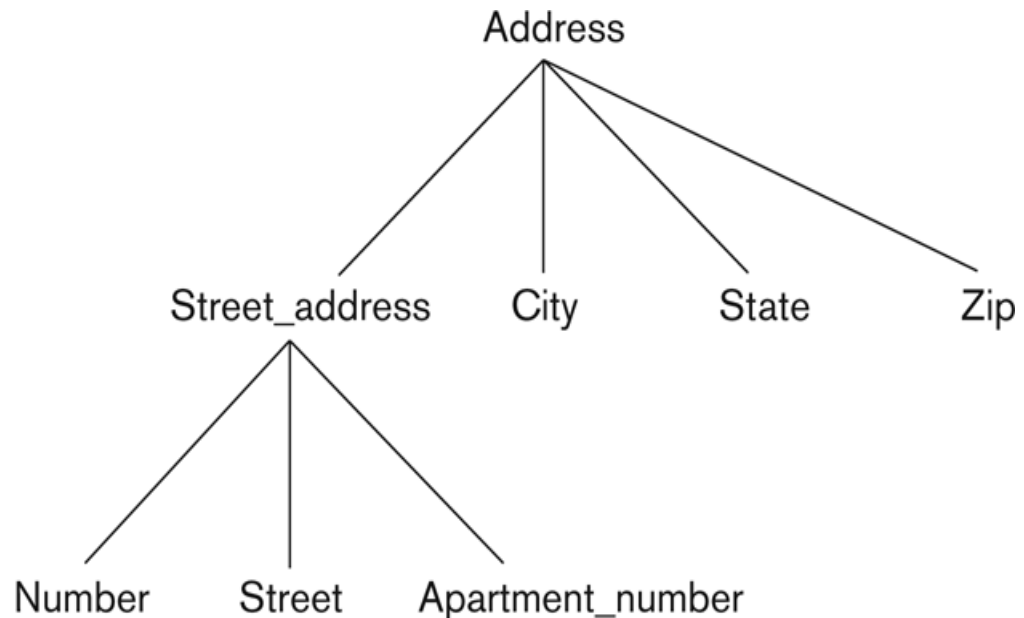
# ER Model Concepts

## ■ Entities and Attributes

- Entity is a basic concept for the ER model. Entities are specific things or objects (live or alive) 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 like- 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.
- 2) Composite
  - The attribute may be composed of several components. For example: Address(Apt#, House#, Street, City, State, ZipCode, Country),



**Figure 3.4**

A hierarchy of composite attributes.

# Types of Attributes

- 3) 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}.
- 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

# Types of Attributes

- 4) NULL value
  - In some cases, a particular entity may not have an applicable value for an attribute.
  - For example, the apartment\_number attribute of an address –Figure 3.4 - applies only to addresses that are in apartment buildings and not to other types of residences, such as single-family homes.
  - In case of another example, a college\_degrees attribute applies only to people with college degrees.
- For such situations, a special value called NULL is created. An address of a single-family home would have NULL for its apartment\_number attribute.
  - NULL can also be used if we do not know the value of an attribute for a particular entity—for example, if we do not know the home phone number of 'John Smith'.

# Entity Types and Key Attributes

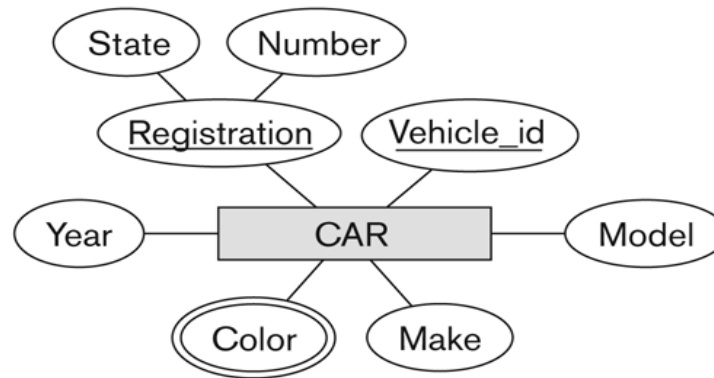
- Entities with the same basic attributes are grouped or typed into an entity type.
  - For example, the entity type EMPLOYEE and PROJECT.
- An attribute of an entity type for which each entity must have a unique value is called a key attribute of the entity type.
  - For example, SSN of EMPLOYEE.

# Entity Types and Key Attributes

- A key attribute may be composite.
  - VehicleTagNumber (in Figure 3.7) 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
  - (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)



**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}

CAR<sub>1</sub>  
((ABC 123, TEXAS), TK629, Ford Mustang, convertible, 2004 {red, black})

CAR<sub>2</sub>  
((ABC 123, NEW YORK), WP9872, Nissan Maxima, 4-door, 2005, {blue})

CAR<sub>3</sub>  
((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**  
(Simply as records or rows in commercial RDBMS)
- 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



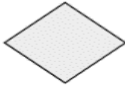
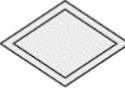




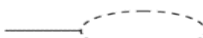
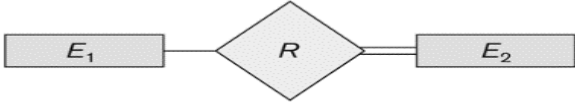
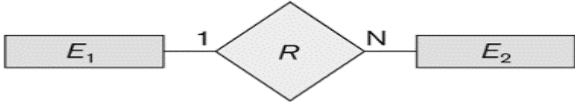
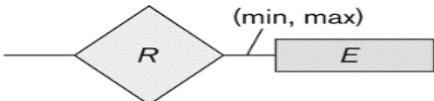
- Each simple attribute is associated with a value set
  - E.g., Lastname has a value which is a character string of upto 15 characters.
  - Date has a value consisting of MM-DD-YYYY where each letter is an integer
- A **value set** specifies the set of values associated with an attribute
- Value sets are similar to data types in most programming languages – e.g., integer, character (n), real, float, Boolean, enumerated type, subrange, and so on.

# Drawing ER conceptual design

- 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

# Drawing ER conceptual design

**Figure 3.14**  
Summary of the  
notation for ER  
diagrams.

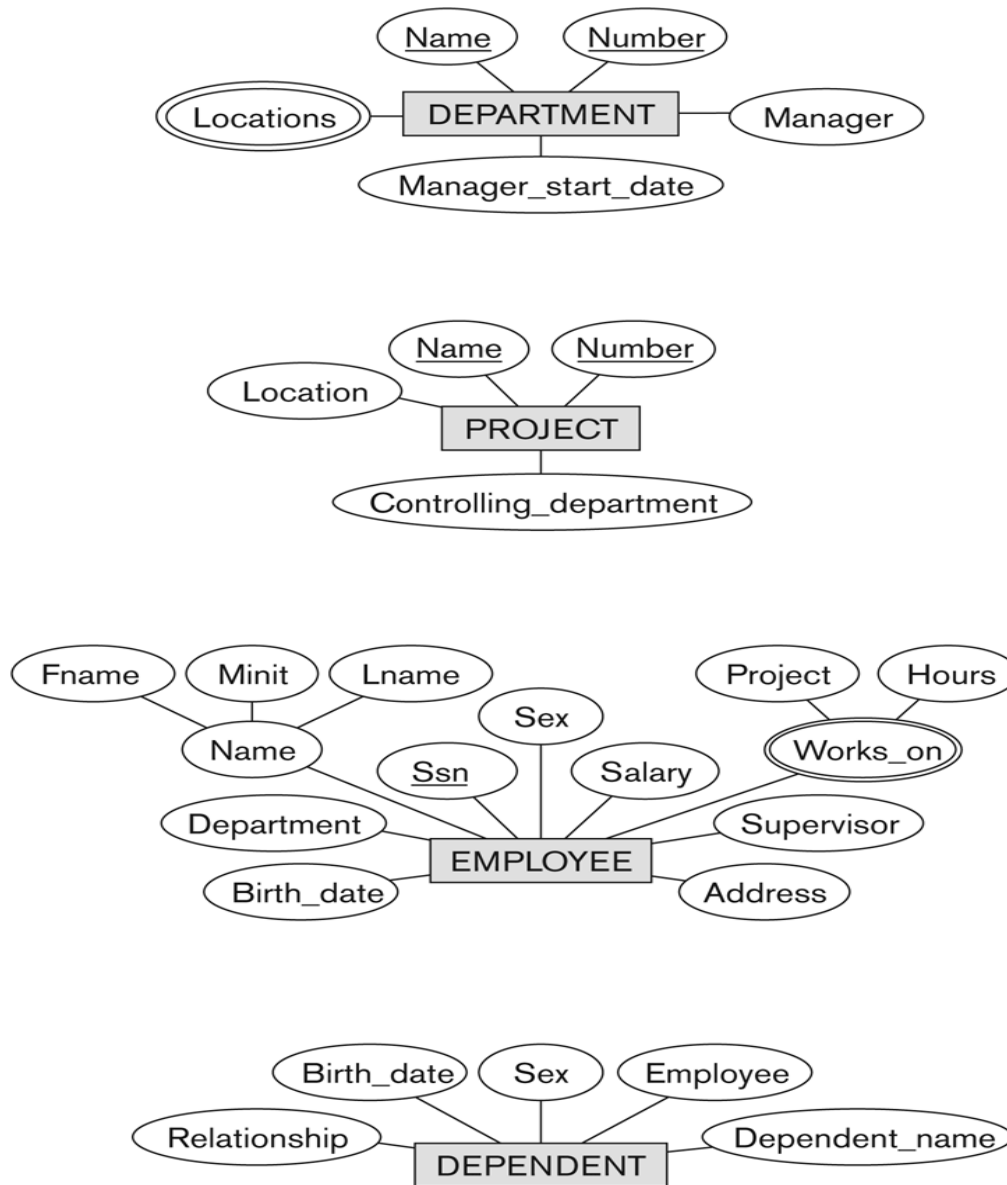
Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Identifying Relationship
	Attribute
	Key Attribute
	Multivalued Attribute
	Composite Attribute
	Derived Attribute
	Total Participation of $E_2$ in $R$
	Cardinality Ratio 1: N for $E_1:E_2$ in $R$
	Structural Constraint (min, max) on Participation of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

(Note: For getting the feel that you are a data designer and so first you interviewed others (client) and collected requirements that are on slide 8 and 9 and on the base of that you start drawing in the next slide)

# Initial Design of Entity Types:



**Figure 3.8**  
Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

# Refining the initial design by introducing **relationships**

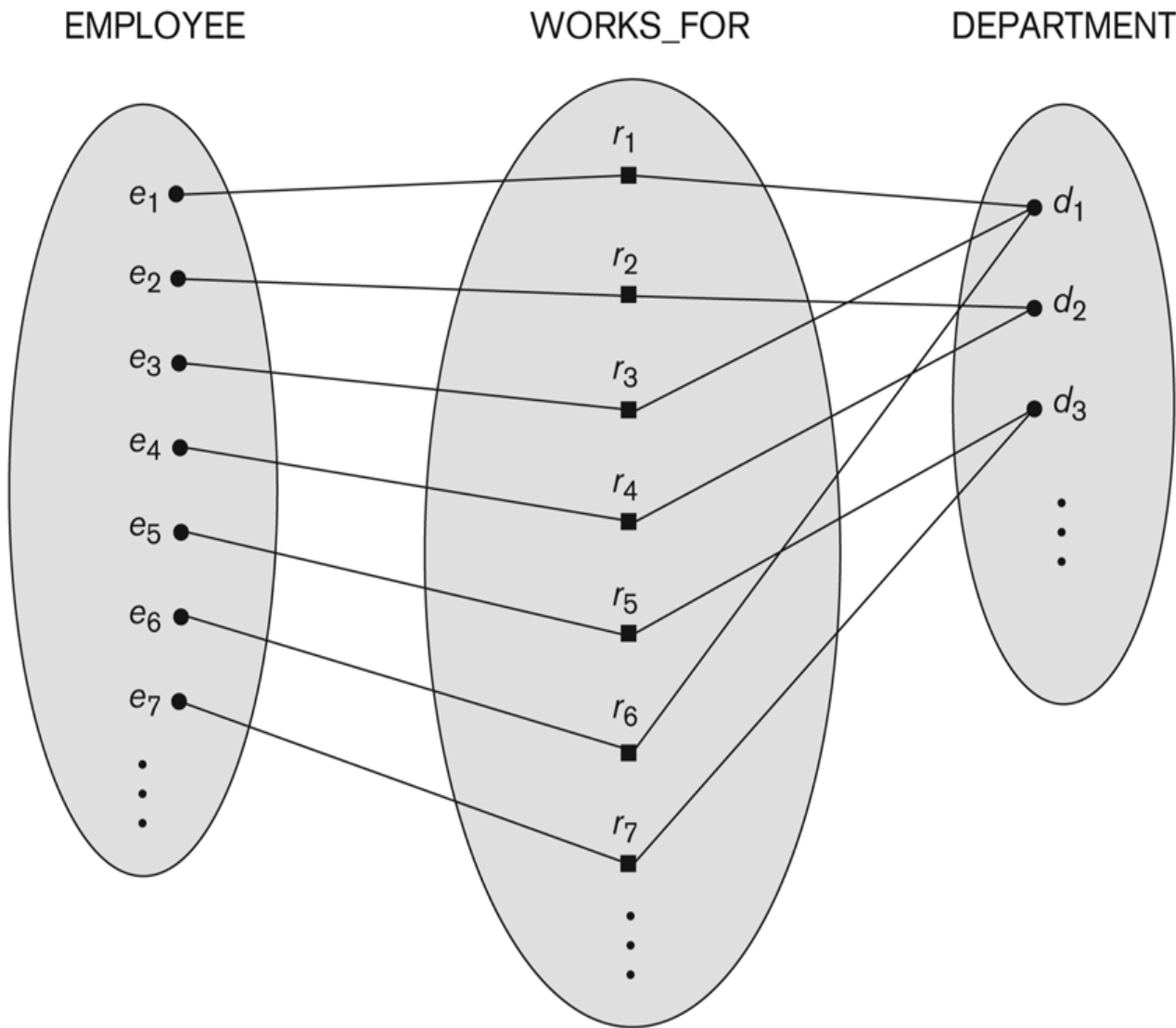
- The initial design is typically not complete
- Some aspects in the requirements will be represented as **relationships**
- ER model has three main concepts:
  - Entities (and their entity types and entity sets)
  - Attributes (simple, composite, multivalued)
  - Relationships (and their relationship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

# Relationships and Relationship Types

- 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.



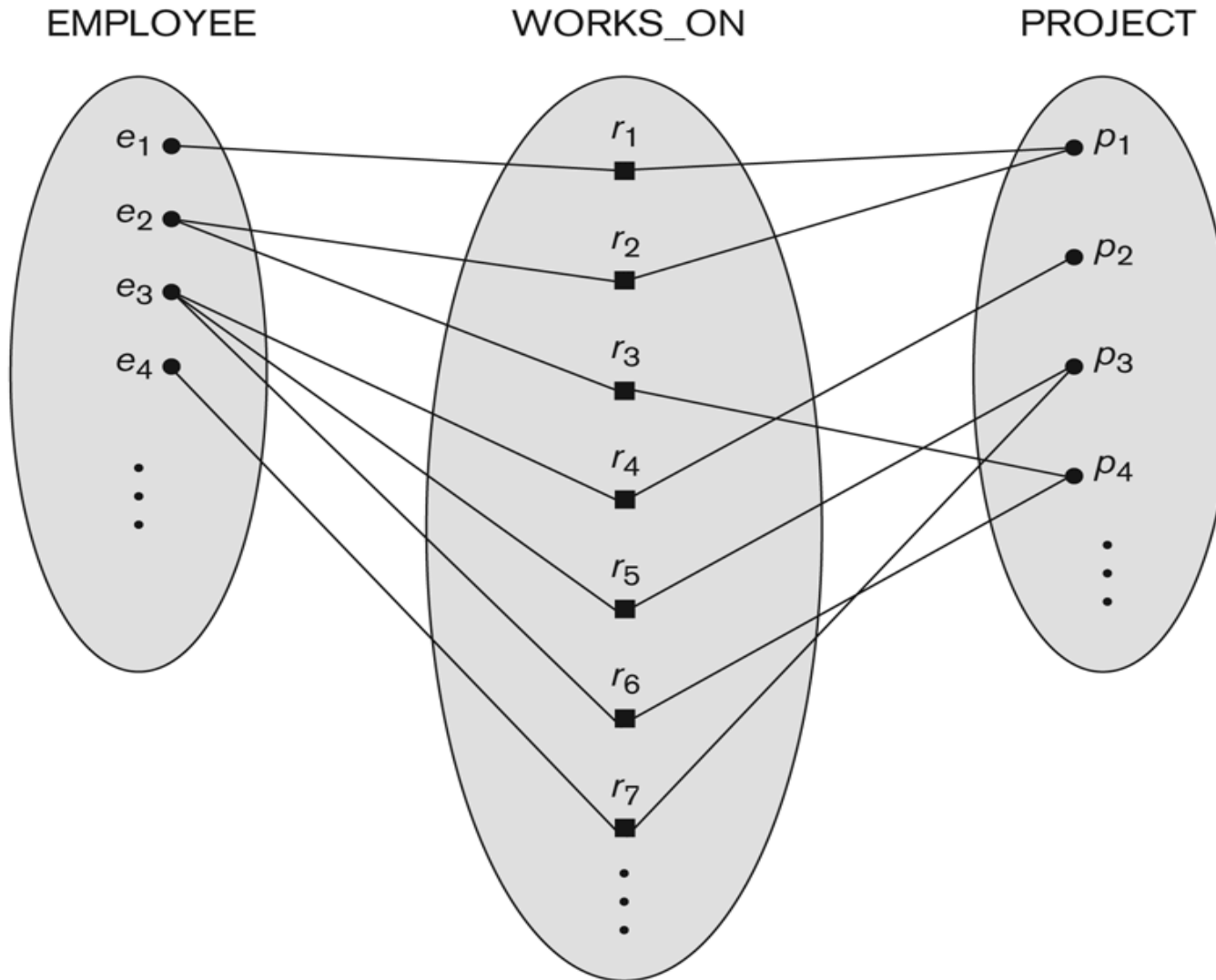
# Relationship instances of the WORKS\_FOR N:1 relationship.



**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 WORKS\_ON relationship



**Figure 3.13**  
An M:N relationship,  
WORKS\_ON.

# Relationship type vs. relationship set

- 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
- Previous figures displayed the relationship sets
- Each instance in the set relates individual participating entities – one from each participating entity type

# Refining the COMPANY database schema by introducing relationships

- By examining the requirements, six relationship types are identified
- All are *binary* relationships (i.e., 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)

# ER DESIGN – Relationship Types are:

WORKS\_FOR, MANAGES, WORKS\_ON, CONTROLS, SUPERVISION, DEPENDENTS\_OF

- 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 the relationship type is not shown with an arrow. The name should typically be readable from left to right and top to bottom.

# ER DESIGN – Relationship Types are:

WORKS\_FOR, MANAGES, WORKS\_ON, CONTROLS, SUPERVISION, DEPENDENTS\_OF

- In the refined design, some attributes from the initial entity types are refined into relationships:
  - Manager of DEPARTMENT -> MANAGES
  - Works\_on of EMPLOYEE -> WORKS\_ON
  - Department of EMPLOYEE -> WORKS\_FOR etc
- In general, more than one relationship type can exist between the same participating entity types
  - MANAGES and WORKS\_FOR are distinct relationship types between EMPLOYEE and DEPARTMENT
  - Different meanings and different relationship instances.

# Constraints on Relationships

- Constraints on Relationship Types  
(also known as ratio constraints)
  - Cardinality Ratio (specifies *maximum* participation)
    - One-to-one (1:1)
    - One-to-many (1:N)
    - 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)

# 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):
  - 1) total (called existence dependency) or
  - 2) partial.
  - Total has shown by a double line, partial by a single line.
  - (NOTE: These are easy to specify for Binary Relationship Types).



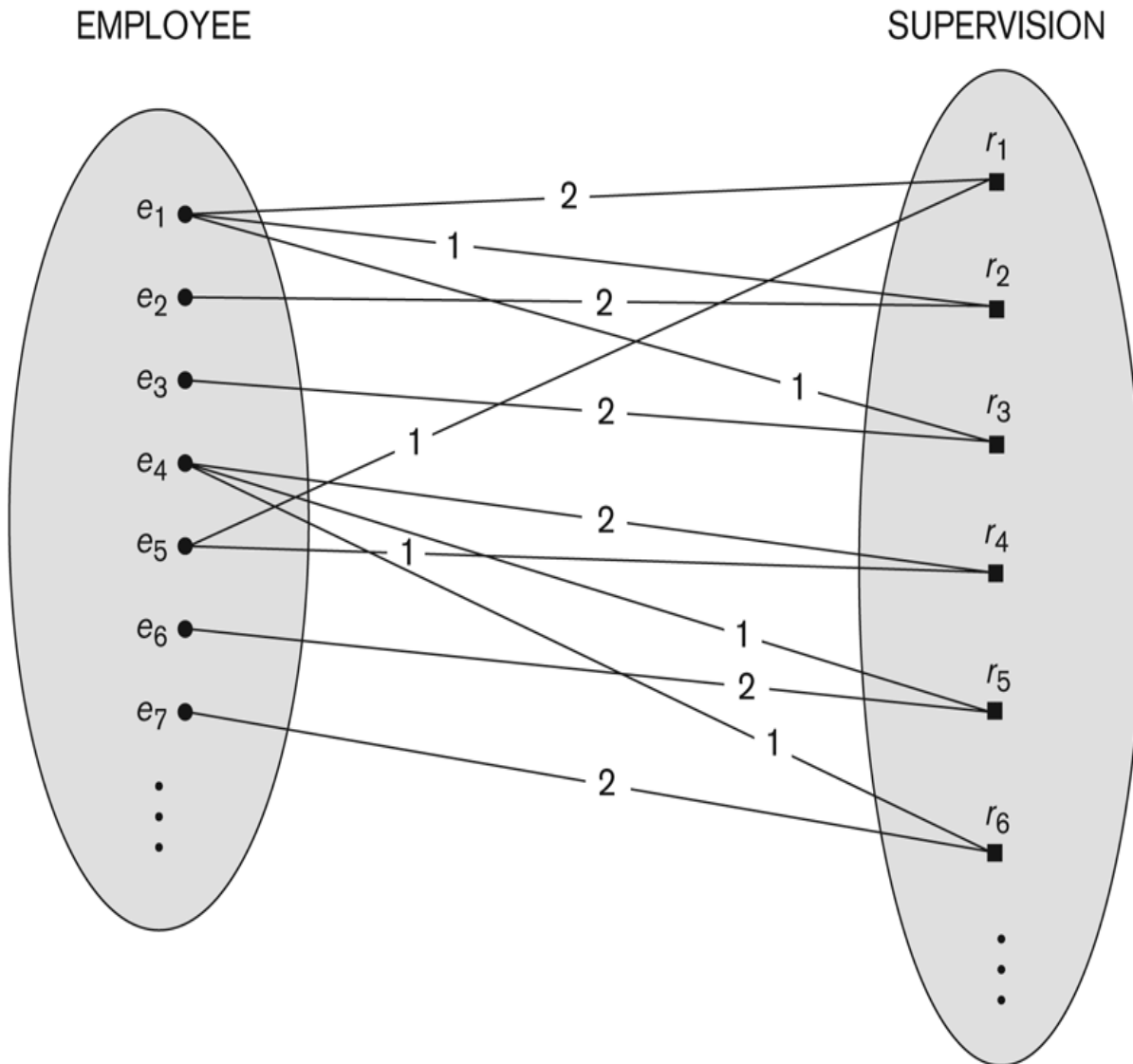
# Recursive Relationship Type

- A relationship type between the same participating entity type in **distinct roles**, known as **recursive** relationship.
- Also called a **self-referencing** relationship type.
- For 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

# Recursive Relationship Type

- In 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.

# Supervision : A Recursive Relationship



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

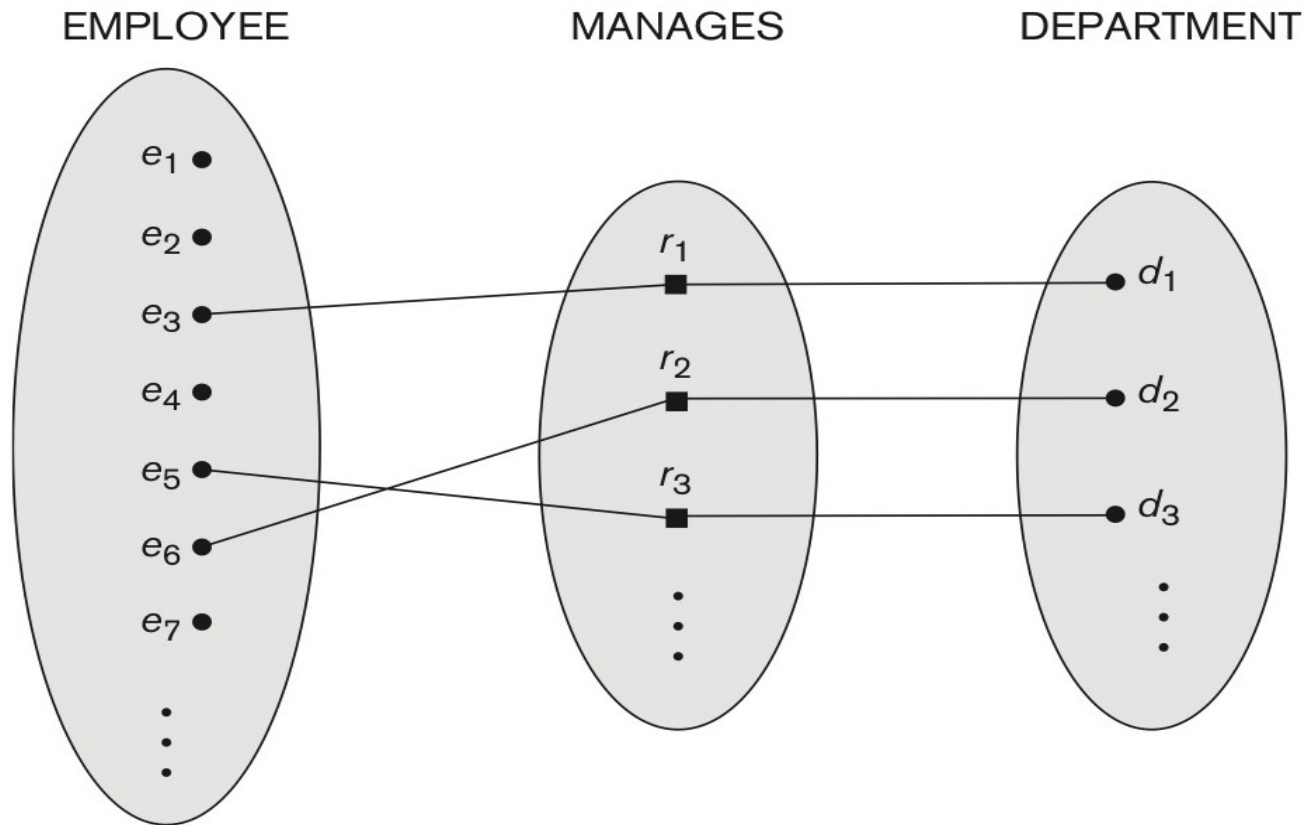
# Weak Entity Types

- An entity that does not have a key attribute and in that case, it 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.
  - DEPENDENT is a *weak entity type*
  - EMPLOYEE is its identifying entity type via the identifying relationship type DEPENDENT\_OF
  - Name of DEPENDENT is the partial key and Ssn of EMPLOYEE as identifying. That combination will be a work as a key attribute.

# 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.
  - Another example is to include the date on which a manager started managing a department via an attribute Start\_date for the MANAGES relationship type in Figure 3.12 (next slide)
- Note: 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

# Attributes of Relationship types

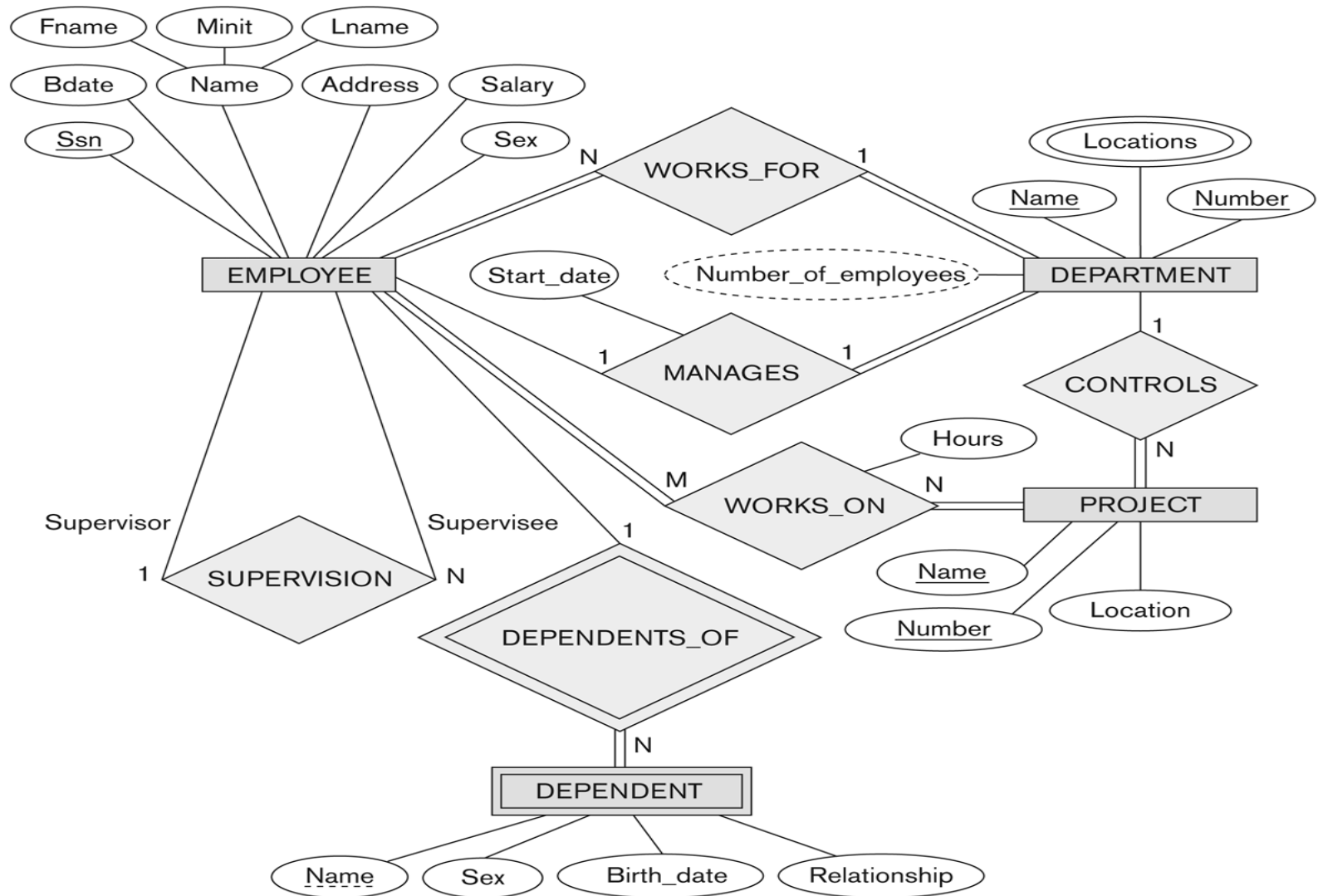


**Figure 3.12** A 1:1 relationship, MANAGES.

# Final ER Design :

- In next slide, we have following six relationships, finally.
  - MANAGES, which is a 1:1(one-to-one) relationship type between EMPLOYEE and DEPARTMENT.
  - WORKS\_FOR, a 1:N (one-to-many) relationship type between DEPARTMENT and EMPLOYEE.
  - CONTROLS, a 1:N relationship type between DEPARTMENT and PROJECT.
  - SUPERVISION, a 1:N relationship type between EMPLOYEE (in the supervisor role) and EMPLOYEE (in the supervisee role)
  - WORKS\_ON, determined to be an M:N (many-to-many) relationship type with attribute Hours, after the users indicate that a project can have several employees working on it.
  - DEPENDENTS\_OF, a 1:N relationship type between EMPLOYEE and DEPENDENT

# Final ER Design :



**Figure 3.2**

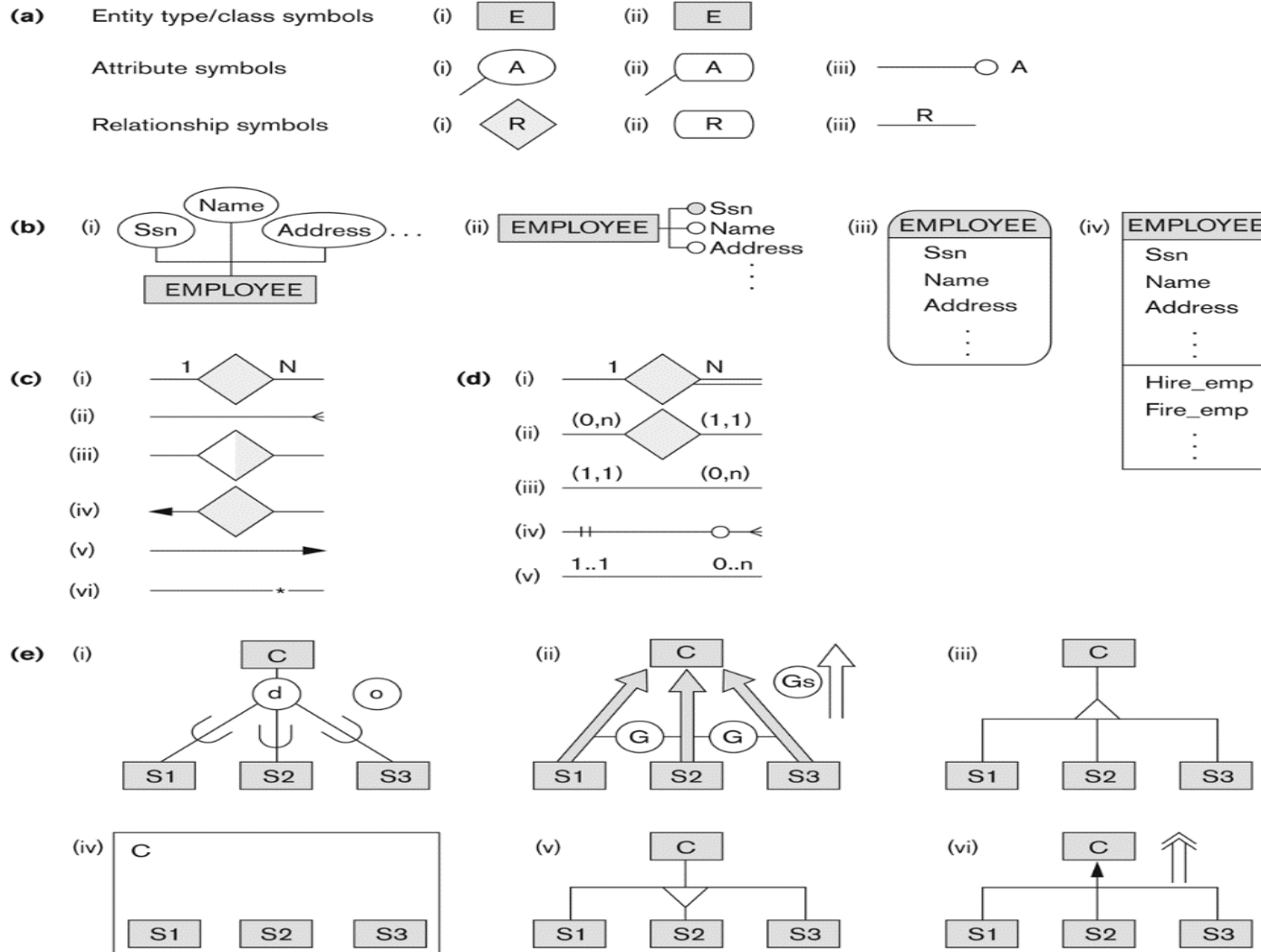
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.



# 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.
- Figure A-1, 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.

# Other alternative diagrammatic notations



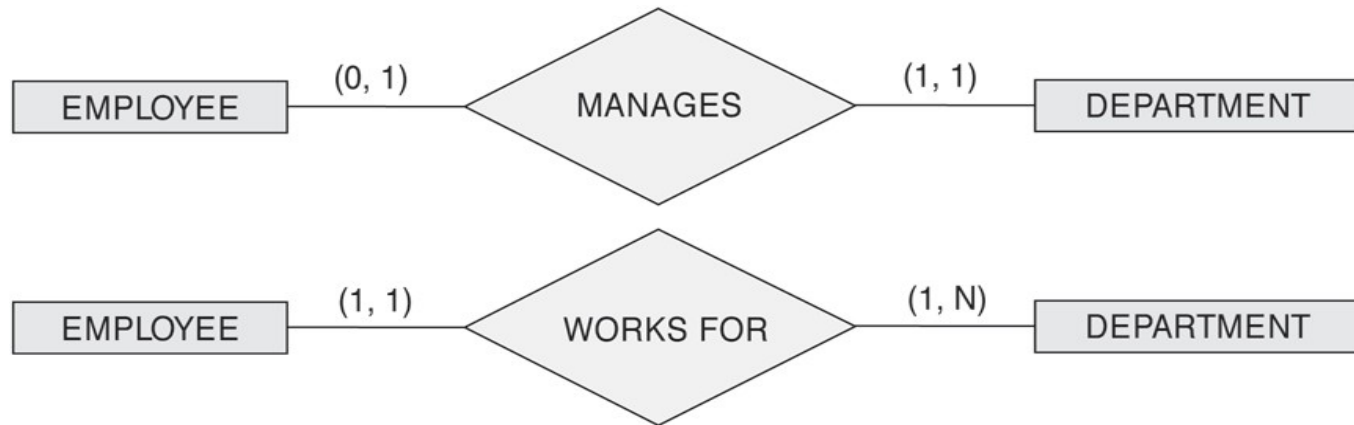
**Figure A.1**

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.

# 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  $\min \leq \max$ ,  $\min \geq 0$ ,  $\max \geq 1$ .
- 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.

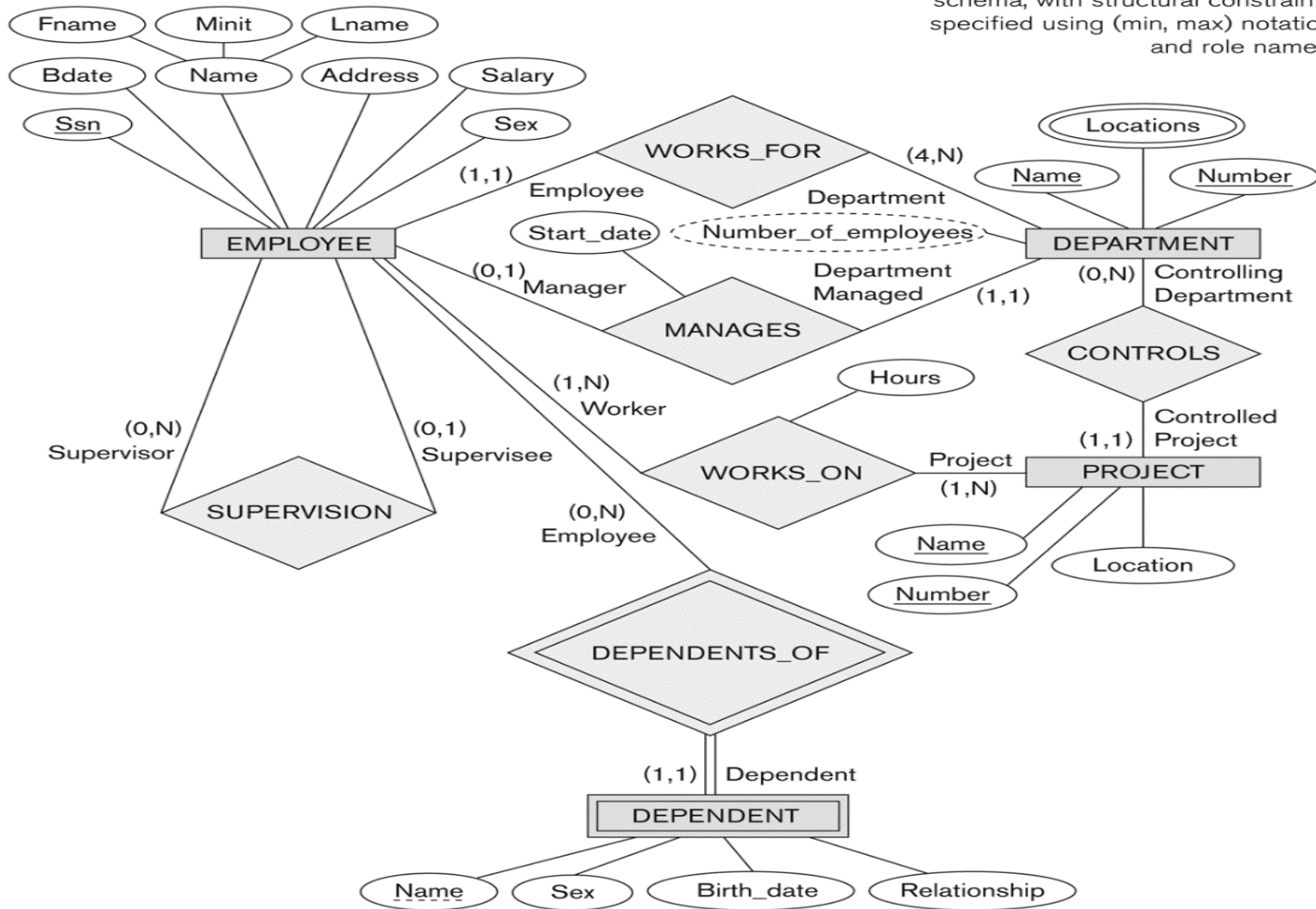
- (0,1): An employee manage **0** department or **1** department.
- (1,1): Department manage by **1** employee.
- (1,1): Employee works for **1** department.
- (1,N): Department can have **1** employee or **N** number of employees.

# COMPANY ER Schema Design

## using (min, max) notation

**Figure 3.15**

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



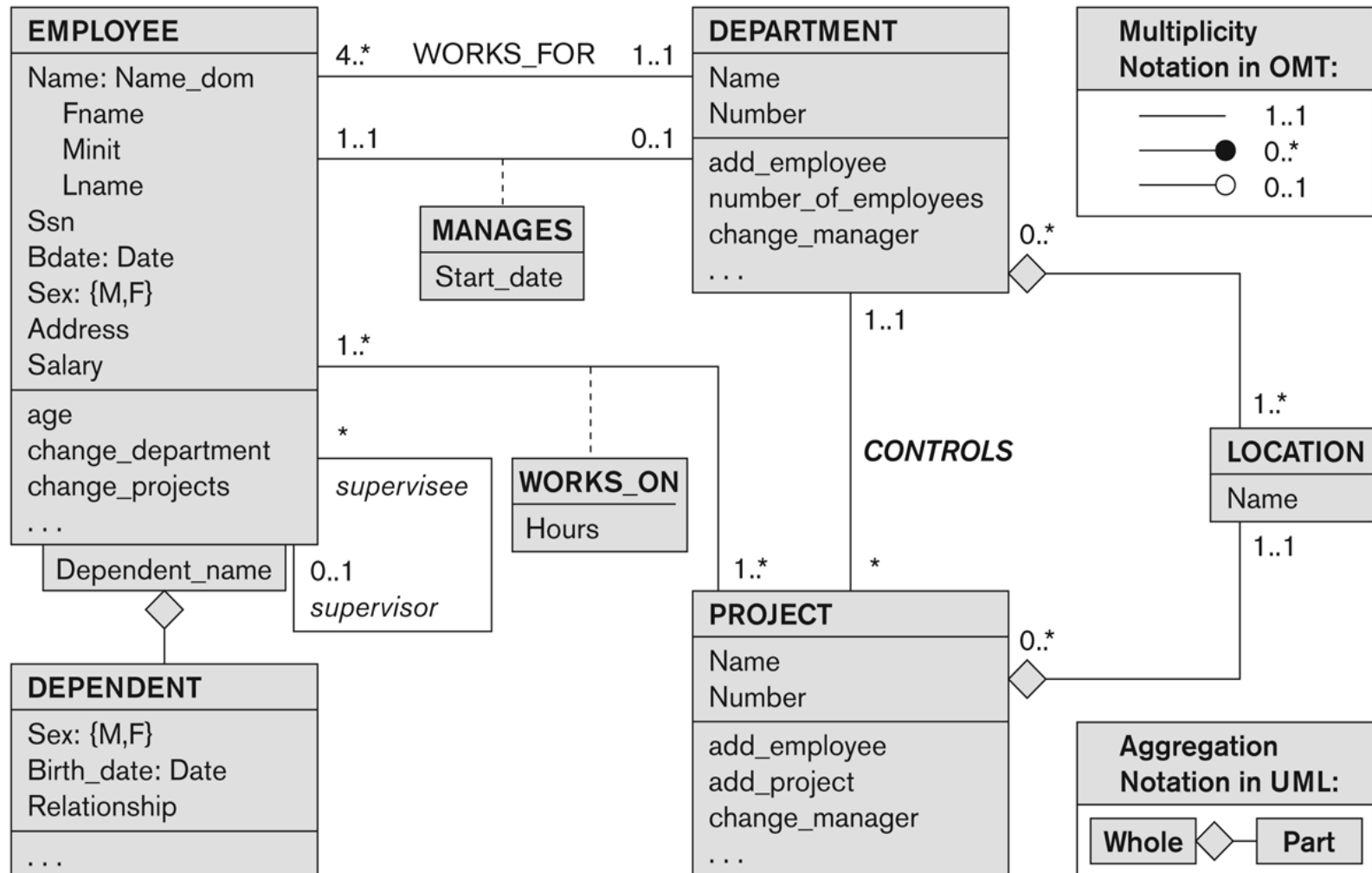
# UML class diagrams

- UML Class Diagrams are another highly used Alternative Notations in Industry.
- Represent classes (similar to entity types) as large rounded boxes with three sections:
  - Top section includes entity type (class) name
  - Second section includes attributes
  - Third section includes class operations (operations are not in basic ER model)
- Relationships (called associations) represented as lines connecting the classes
  - Other UML terminology also differs from ER terminology
- Used in database design and object-oriented software design.
- UML has many other types of diagrams for software design.

# UML class diagram for COMPANY database

**Figure 3.16**

The COMPANY conceptual schema in UML class diagram notation.



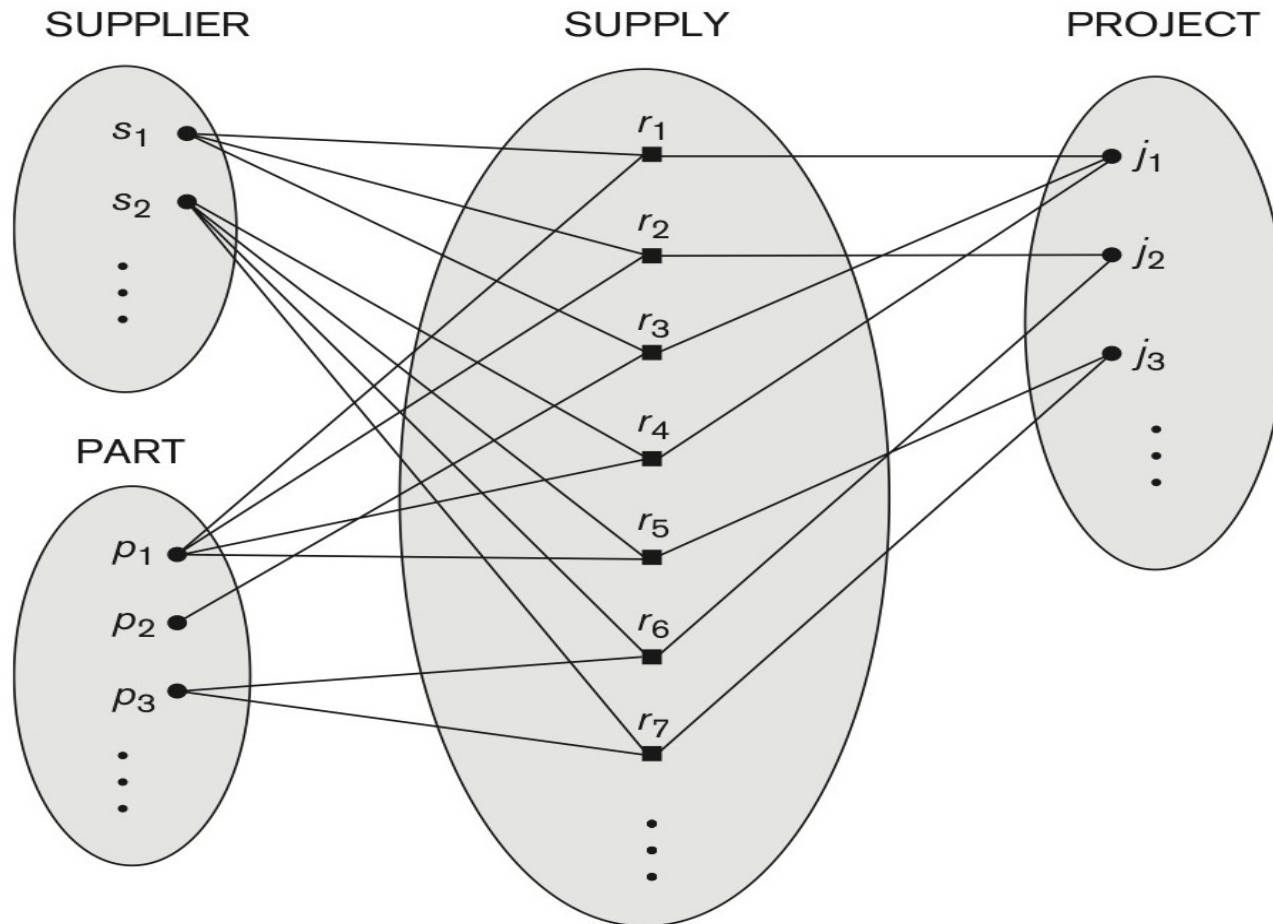
# Relationships of Higher Degree

- 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
- Constraints are harder to specify for higher-degree relationships ( $n > 2$ ) than for binary relationships.



# Relationships of Higher Degree

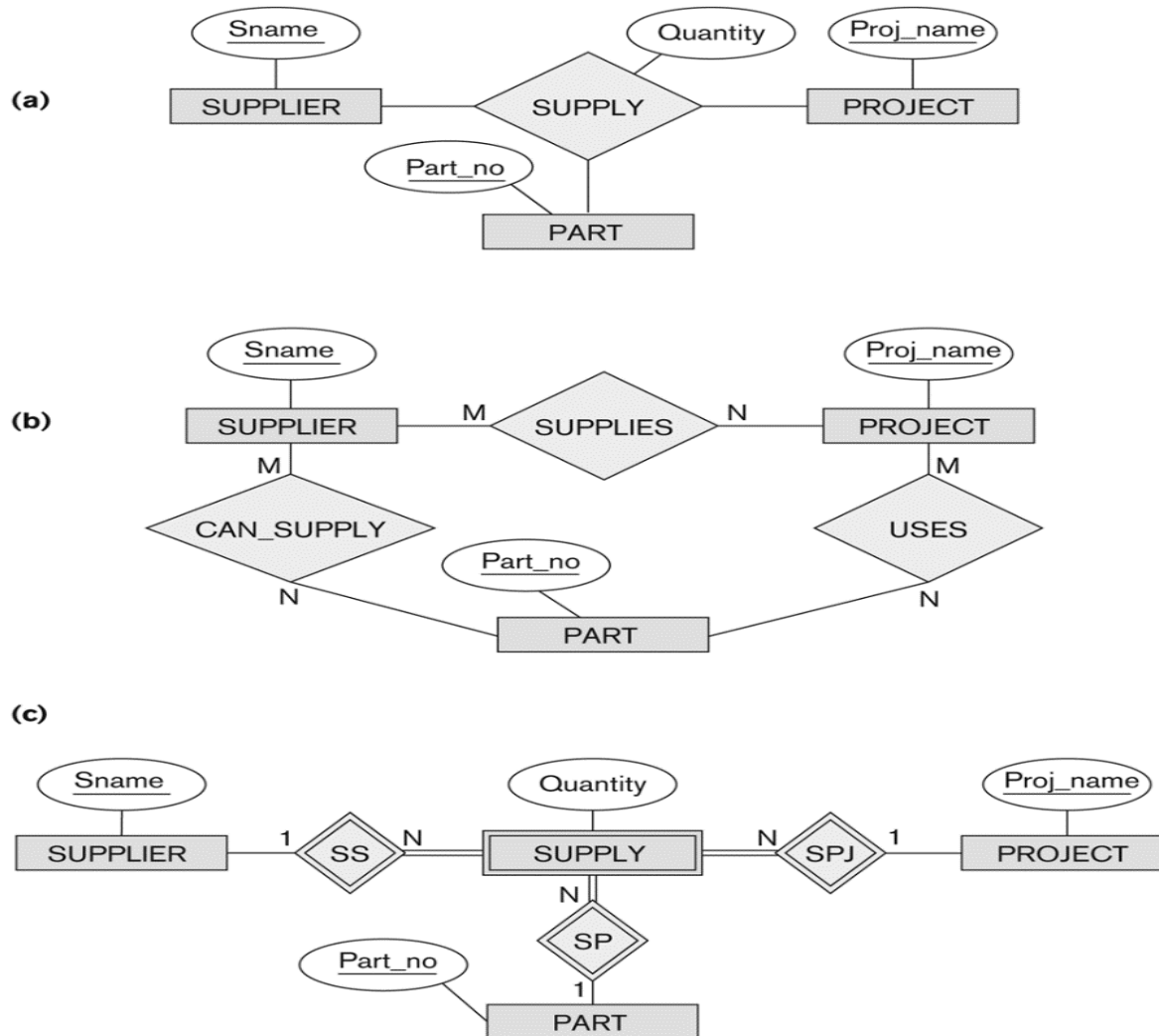
- An example of a ternary relationship is SUPPLY, shown in following figure, where each relationship instance  $r_i$  associates three entities—a supplier  $s$ , a part  $p$ , and a project  $j$ —whenever  $s$  supplies part  $p$  to project  $j$ .



# 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 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 can be represented as a weak entity 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.

# Another Example: A UNIVERSITY Database

- Suppose, to keep track of the enrollments in classes and student grades, the UNIVERSITY database need to design.
- University have COLLEGES, DEPARTMENTS within each college, the COURSEs offered by departments, and SECTIONS of courses, INSTRUCTORS who teach the sections etc.
- To design database, entity types and the relationships among these entity types, you might have interviewed them and collected requirements as follow.
  - The university is organized into colleges (COLLEGE), and each college has a unique name (CName), a main office (COffice) and phone (CPhone), and a particular faculty, who is dean of the college.

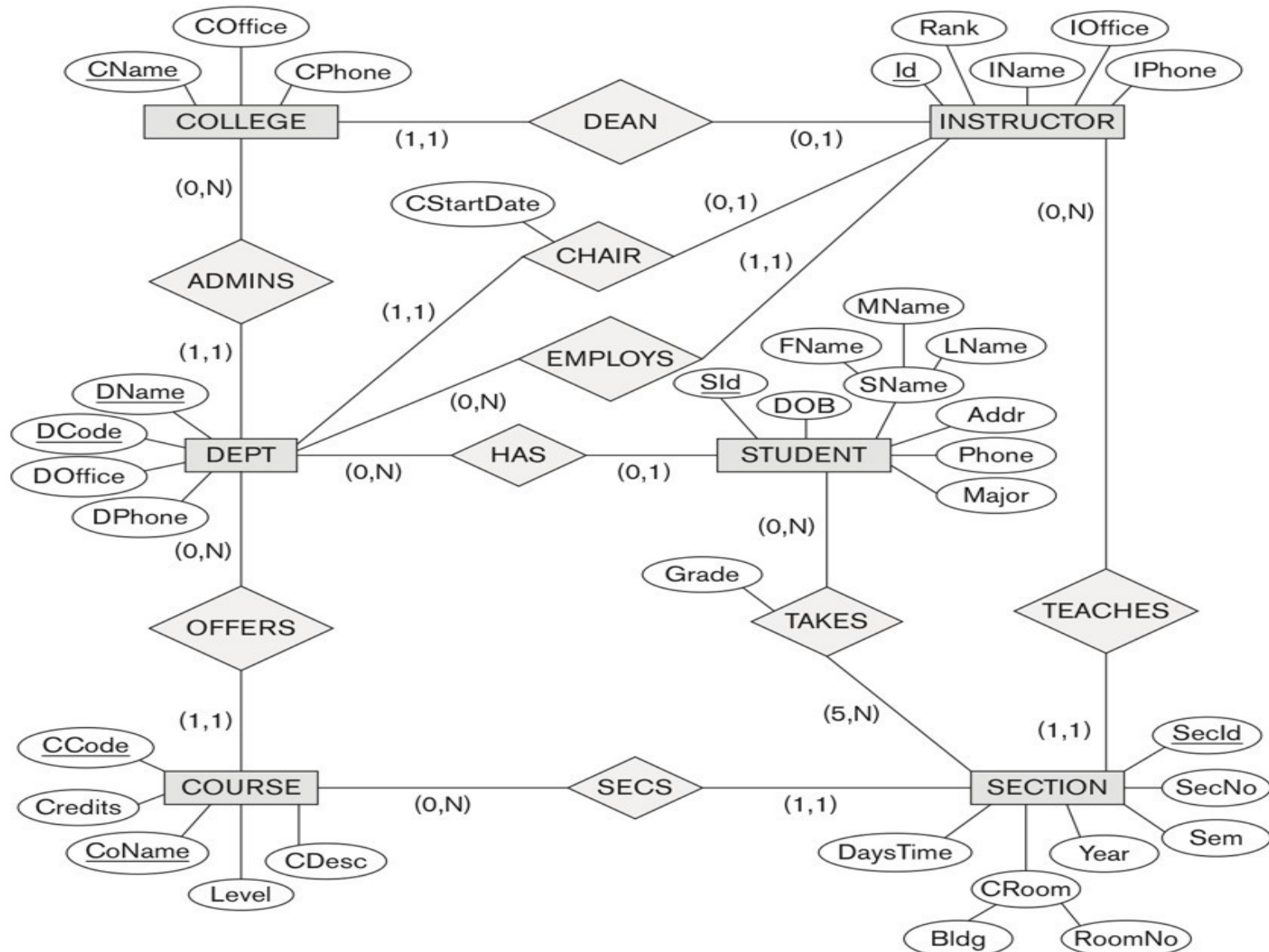
# Another Example: A UNIVERSITY Database

- Each college administers several academic departments (DEPT). Each department has a unique name (DName), a unique code number (DCode), a main office (DOffice) and phone (DPhone), and a particular faculty, who chairs the department. We keep track of the start date (CStartDate) when that faculty member began chairing the department.
- A department offers a number of courses (COURSE), each of which has a unique course name (CoName), a unique code number (CCode), a course level (Level: this can be coded as 1 for freshman level, 2 for sophomore, 3 for junior, 4 for senior, 5 for MS level, and 6 for PhD level), a course credit hours (Credits), and a course description (CDesc).

# Another Example: A UNIVERSITY Database

- The database also keeps track of instructors (INSTRUCTOR); and instructor has a unique id (Id), name (IName), office (IOffice), phone (IPhone), and rank (Rank); in addition, each instructor works for one primary academic department.
- The database store each (STUDENT) data student's name (SName, composed of first name (FName), middle name (MName), last name (LName)), student id (Sid, unique for every student), address (Addr), phone (Phone), major code (Major), and date of birth (DoB).
- Courses are offered as sections (SECTION). Each section has a unique id (SecId). A section also has a section number (SecNo). semester (Sem), year (Year), classroom (Croom), and days/times (DaysTime).

# UNIVERSITY database ER conceptual schema



# Some of the Automated Database Design Tools

(Note: Not all may be on the market now, explore internet and check.)

COMPANY	TOOL	FUNCTIONALITY
Embarcadero Technologies	ER Studio	Database Modeling in ER and IDEF1X
	DB Artisan	Database administration, space and security management
Oracle	Oracle SQL Designer	Database modeling, application development
Popkin Software	System Architect 2001	Data modeling, object modeling, process modeling, structured analysis/design
Platinum (Computer Associates)	Enterprise Modeling Suite: Erwin, BPWin, Paradigm Plus	Data, process, and business component modeling
Persistence Inc.	Pwertier	Mapping from O-O to relational model
Rational (IBM)	Rational Rose	UML Modeling & application generation in C++/JAVA
Resolution Ltd.	Xcase	Conceptual modeling up to code maintenance
Sybase	Enterprise Application Suite	Data modeling, business logic modeling
Visio	Visio Enterprise	Data modeling, design/reengineering Visual Basic/C++



# Data Modeling Tools (Additional Material )

- A number of popular tools that cover conceptual modeling and mapping into relational schema design.
  - Examples: ERWin, S- Designer (Enterprise Application Suite), ER- Studio, Rational rose etc.
- POSITIVES:
  - Serves as documentation of application requirements, easy user interface - mostly graphics editor support
- NEGATIVES:
  - Most tools lack a proper distinct notation for relationships with relationship attributes
  - Mostly represent a relational design in a diagrammatic form rather than a conceptual ER-based design.

# End Chapter Questions

- 3.1. Discuss the phases of database design process.
- 3.3. Define the following terms: *entity*, *attribute*, *attribute value*, *relationship instance*, *composite attribute*, *multivalued attribute*, *derived attribute*, *complex attribute*, *key attribute*, and *value set (domain)*.
- 3.4. What is an entity type? What is an entity set? Explain the differences among an entity, an entity type, and an entity set.
- 3.6. What is a relationship type? Explain the differences among a relationship instance, a relationship type, and a relationship set.

# End Chapter Questions

- 3.12. When is the concept of a weak entity used in data modeling? Define the terms *owner entity type*, *weak entity type*, *identifying relationship type*, and *partial key*.
- 3.14. Discuss the conventions for displaying an ER schema diagram.