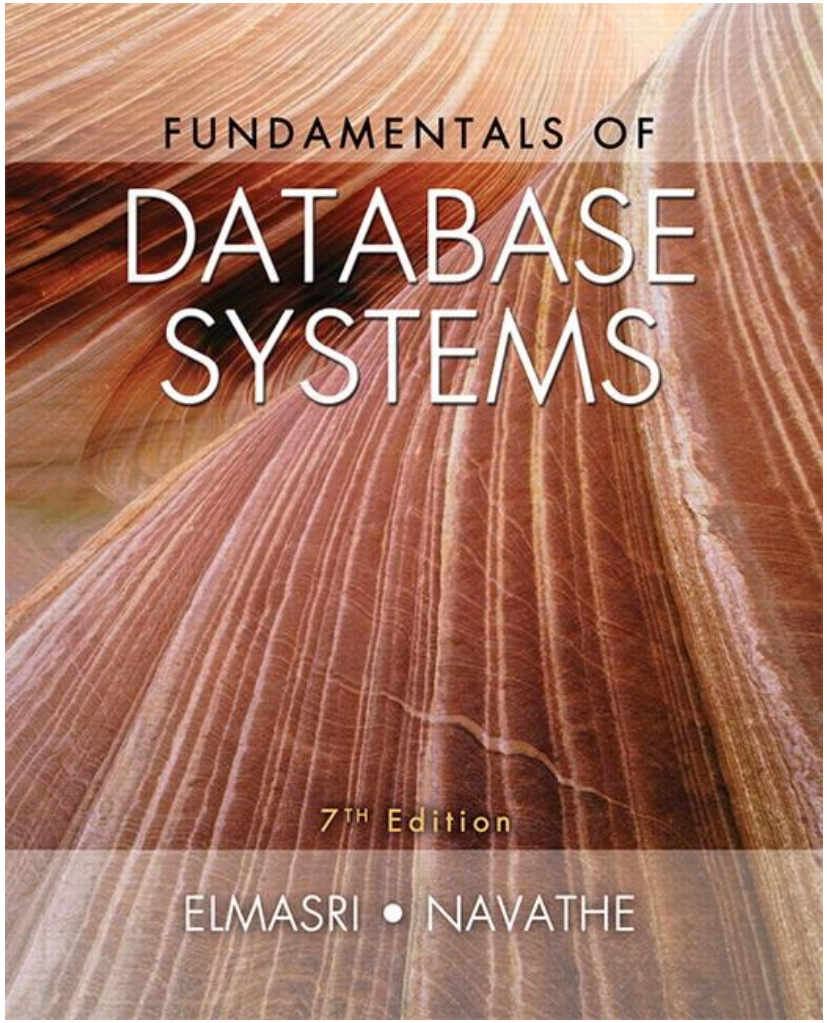


CHAPTER 3

Data Modeling Using the Entity-Relationship (ER) Model

Chapter 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 for COMPANY Schema
- Alternative Notations – UML class diagrams, others
- Relationships of Higher Degree

Example of an Entity-Relationship (ER) Model

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 **PROJECT**s. Each project has a **unique name, unique number** and is located at a single location.

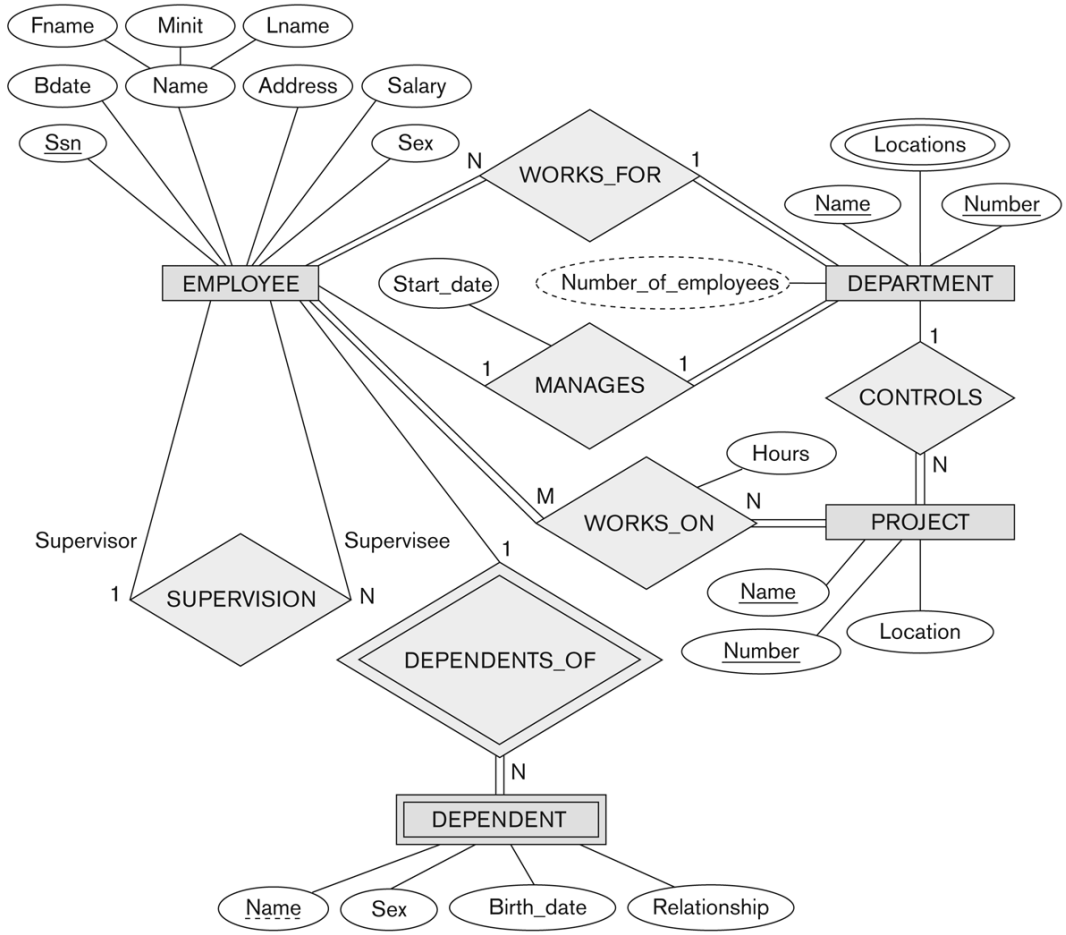
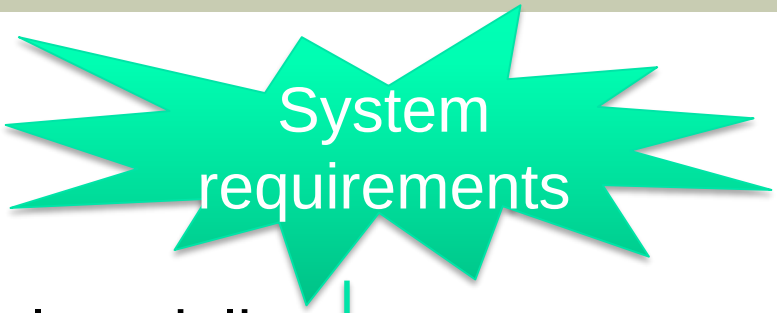


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

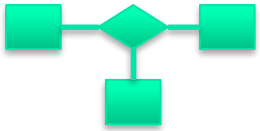
Database design: full picture



A university offers courses to students.
For every student we register name, surname, address, year and student ID.
Every course instance is given on a specific year by a professor,
for whom we store the income and the supervisor.
For each exam we save the date and the grade of the student.

Natural language

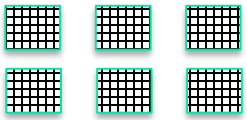
Conceptual modeling



More formal
Not ambiguous

Entity-Relationship diagram

Logical modeling



No implementation or DBMS specific details

Relational model

Physical design

Overview of Database Design Process

- Two main activities:
  - Database design
  - Applications design
- Focus in this chapter on conceptual database design
  - 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

Why a conceptual model

- More formal than natural language.
  - Avoid misconceptions and multiple interpretations.
- Implementation independent (of DBMS).
  - Less technical details.
- High-level description.
  - Easier to understand for people without a technical background (e.g. customers).
- Can be used as documentation.
- Comes with model transformations to be mapped to an implementation data model.
- CASE Tools facilitate some automated generation of databases using ERs

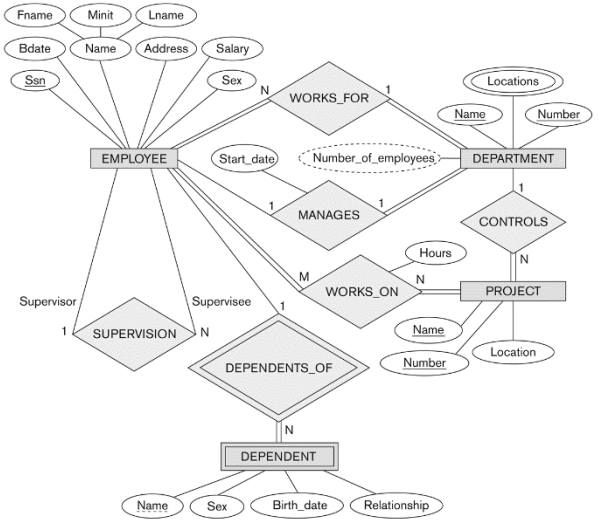


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

Methodologies for Conceptual Design

- Entity Relationship (ER) Diagrams (This Chapter)
- Enhanced Entity Relationship (EER) Diagrams (Chapter 4)
- Use of Design Tools in industry for designing and documenting large scale designs
- The UML (Unified Modeling Language) Class Diagrams are popular in industry to document conceptual database designs.

The ER model

- Originally proposed in:
  - Chen, P. P. S., “The entity-relationship model: towards a unified view of data”, ACM TODS, 1, 1 1976, p. 9-36.

Example COMPANY Database

- 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

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 **PROJECT**s. Each project has a **unique name, unique number** and is located at a single location.

Each employee may *have* a number of **DEPENDENT**s. For each **dependent**, the DB keeps a record of **name, sex, birthdate**, and relationship to the employee.

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.

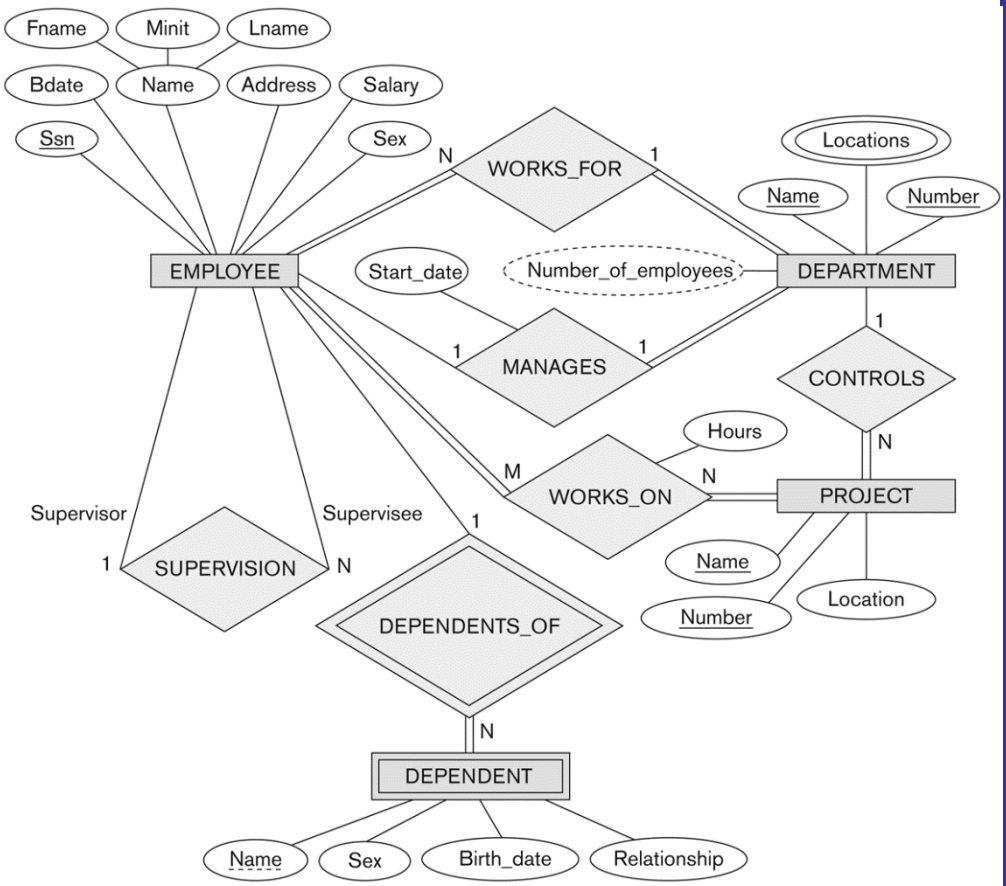


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

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.

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}.

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 (Note: this is different from the relational schema where only one “primary key is underlined).

Entity Set

- Each entity type will have a collection of entities stored in the database
  - Called the **entity set** or sometimes **entity collection**

(a)

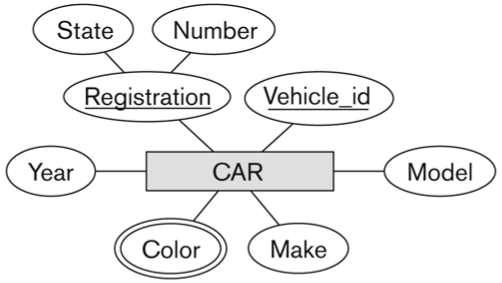


Figure 3.7
The CAR entity type with two key attributes, Registration and Vehicle_id. (a) ER diagram notation. (b) Entity set with three entities.

(b)

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

CAR_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 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

Attributes and Value Sets

- Value sets are similar to data types in most programming languages – e.g., integer, character (n), real, bit
- 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
- See the full ER notation in advance on the next slide

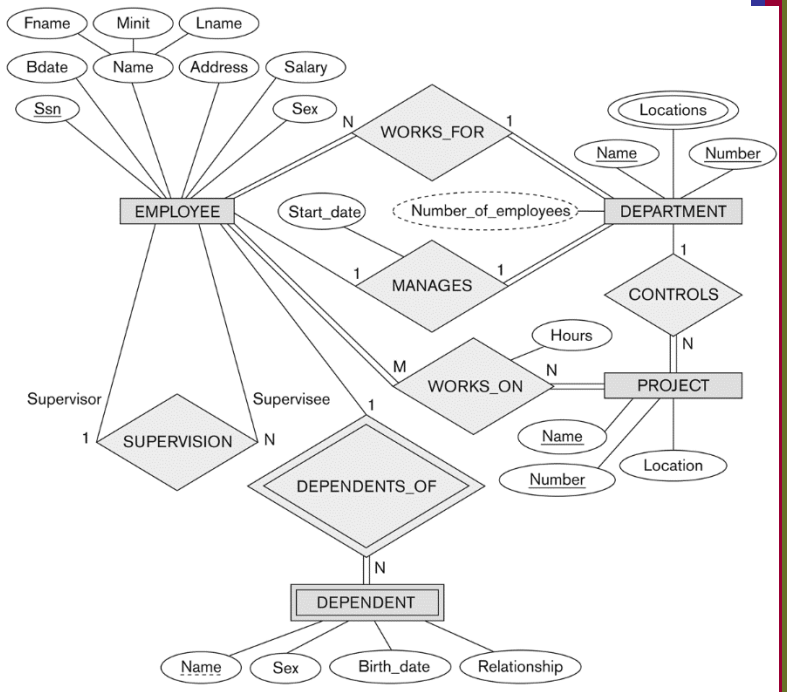
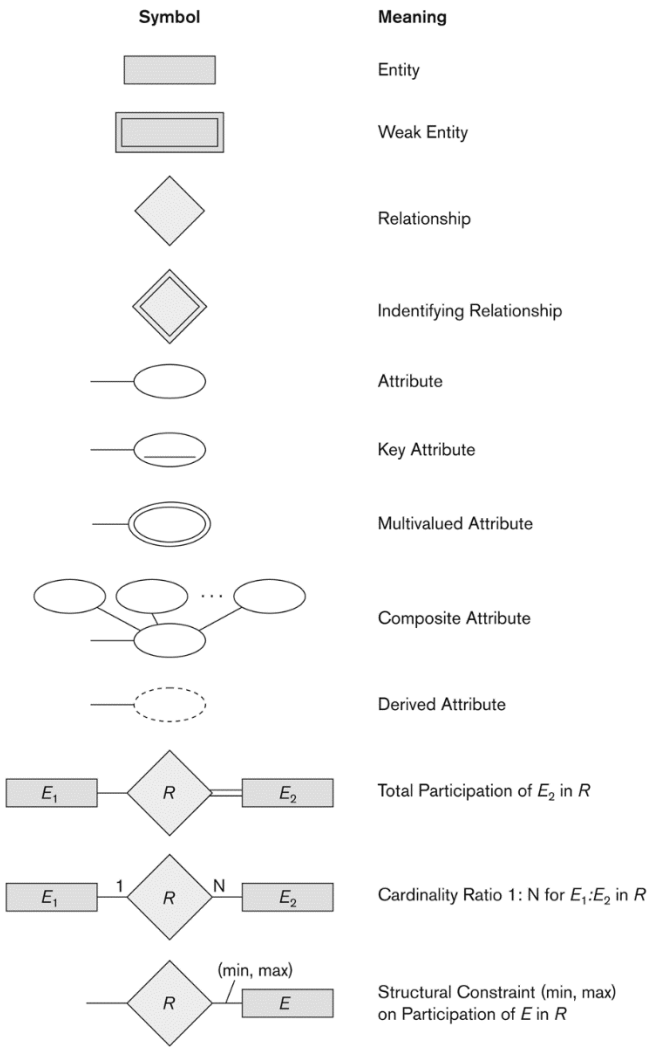


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

NOTATION for ER diagrams

Figure 3.14
Summary of the notation for ER diagrams.



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

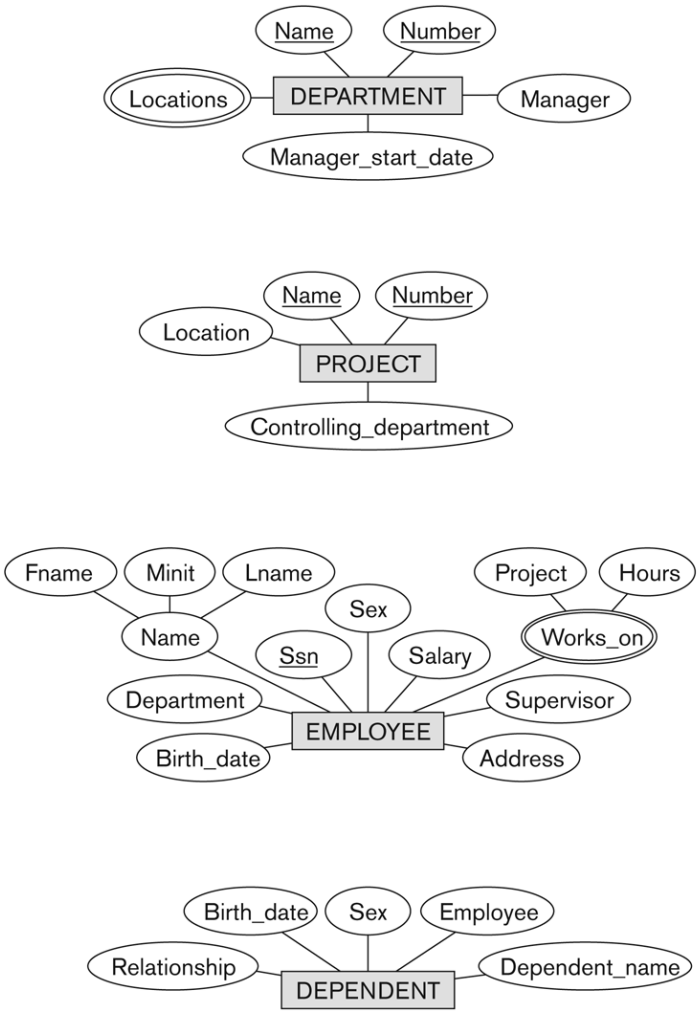


Figure 3.8
Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

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 1:N relationship between EMPLOYEE and DEPARTMENT

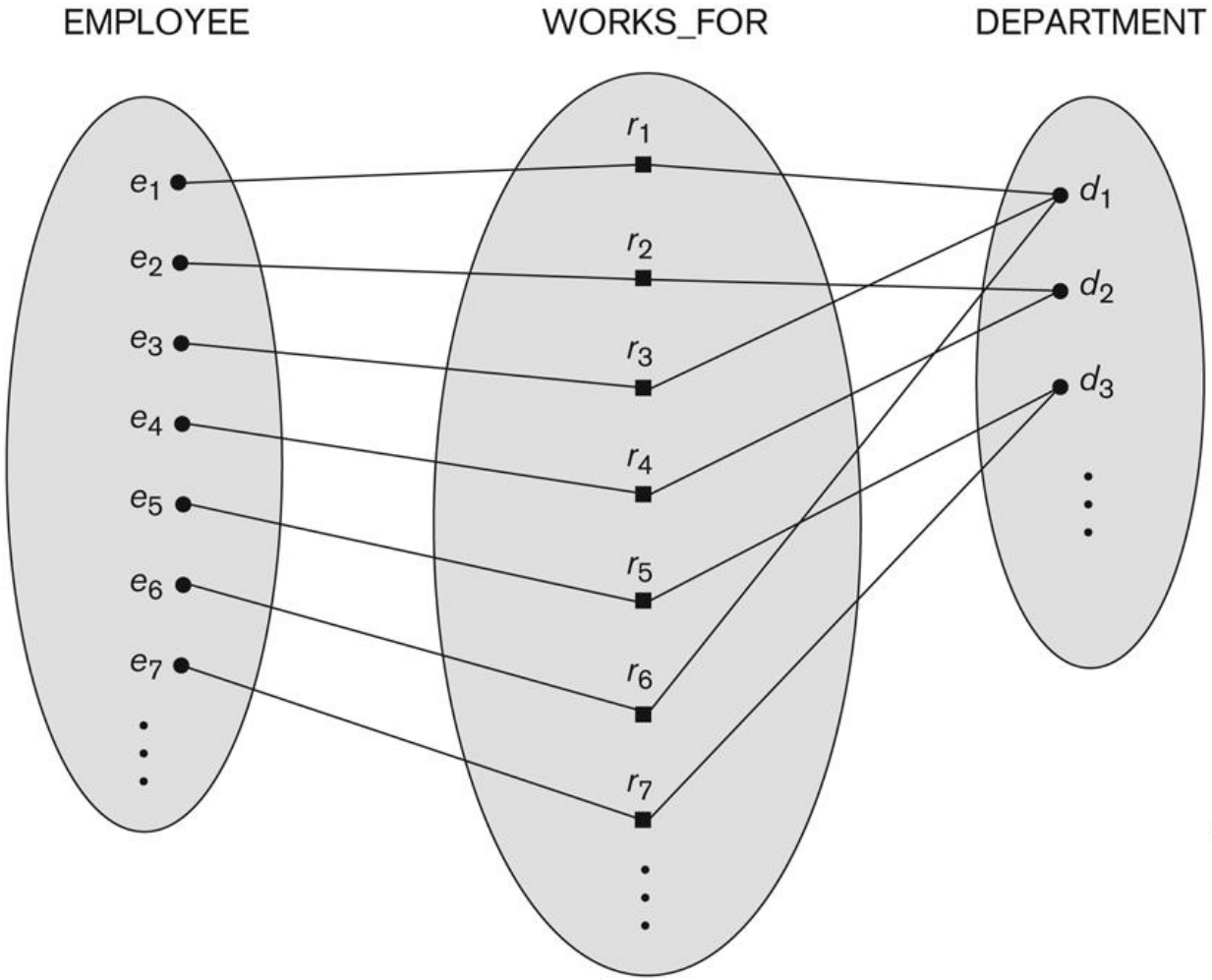


Figure 3.9
Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT.

Relationship instances of the M:N WORKS_ON relationship between EMPLOYEE and PROJECT

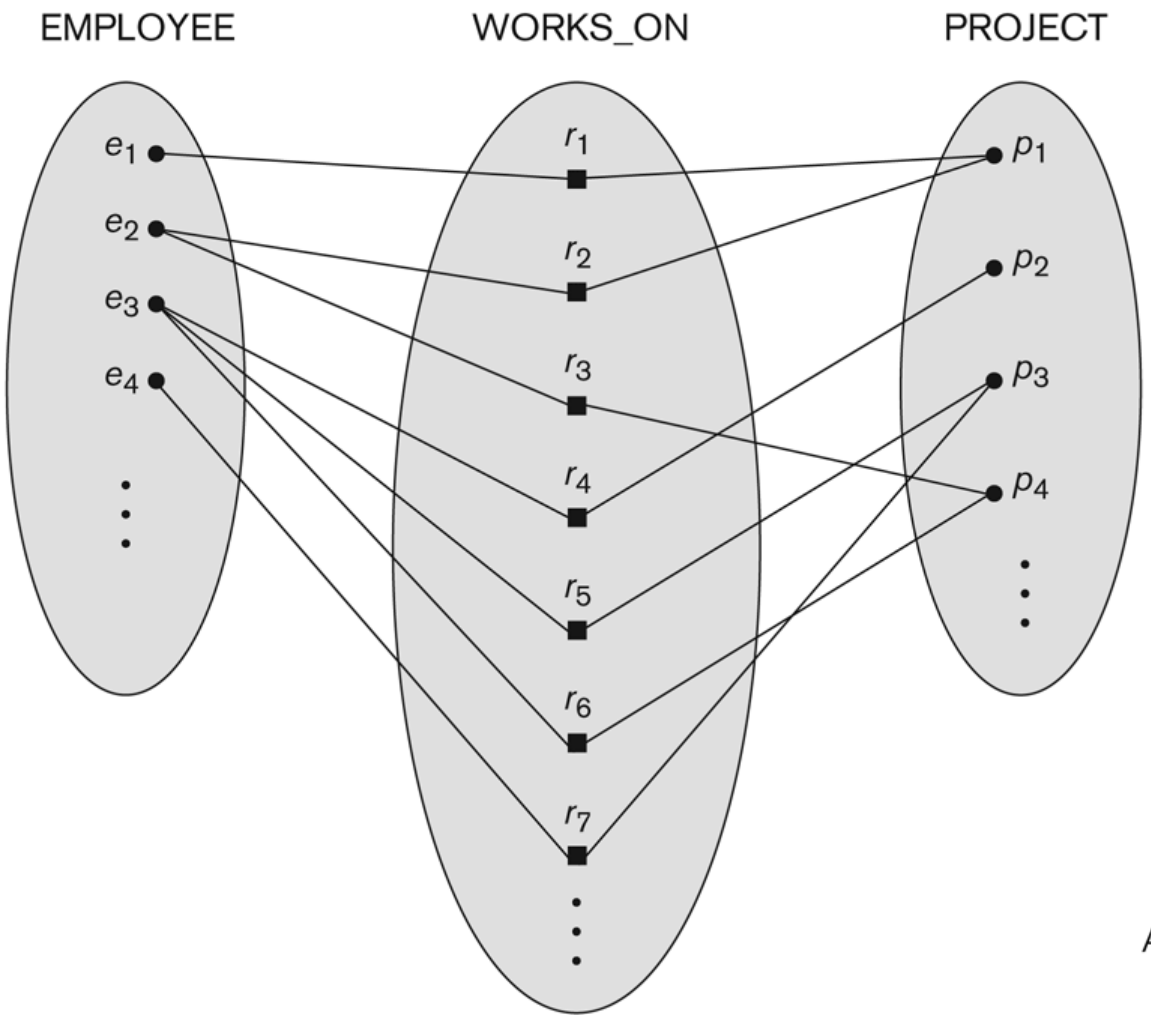


Figure 3.13
An M:N relationship, WORKS_ON.

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 the relationship type is not shown with an arrow. The name should be typically be 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)

ER DIAGRAM – Relationship Types are:

WORKS_FOR, MANAGES, WORKS_ON, CONTROLS, SUPERVISION, DEPENDENTS_OF

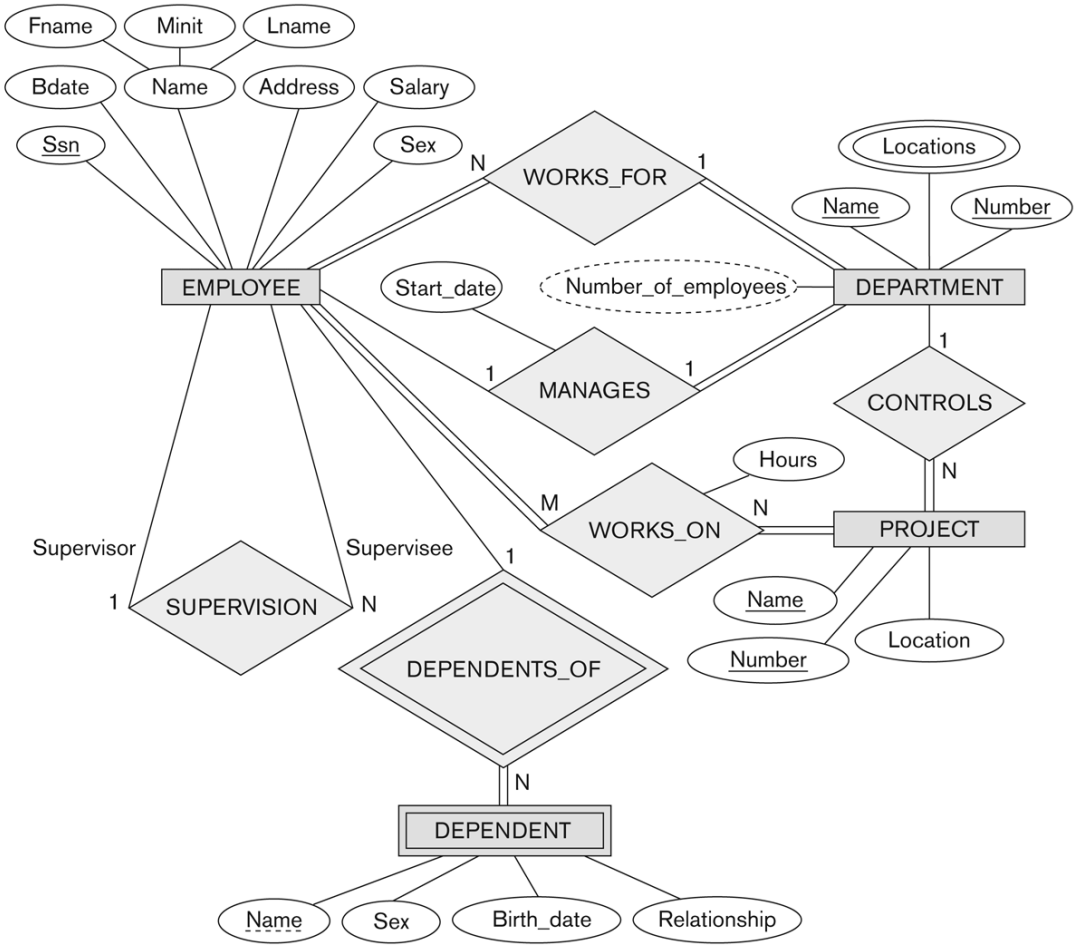


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

Discussion on Relationship Types

- 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) 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)

Many-to-one (N:1) Relationship

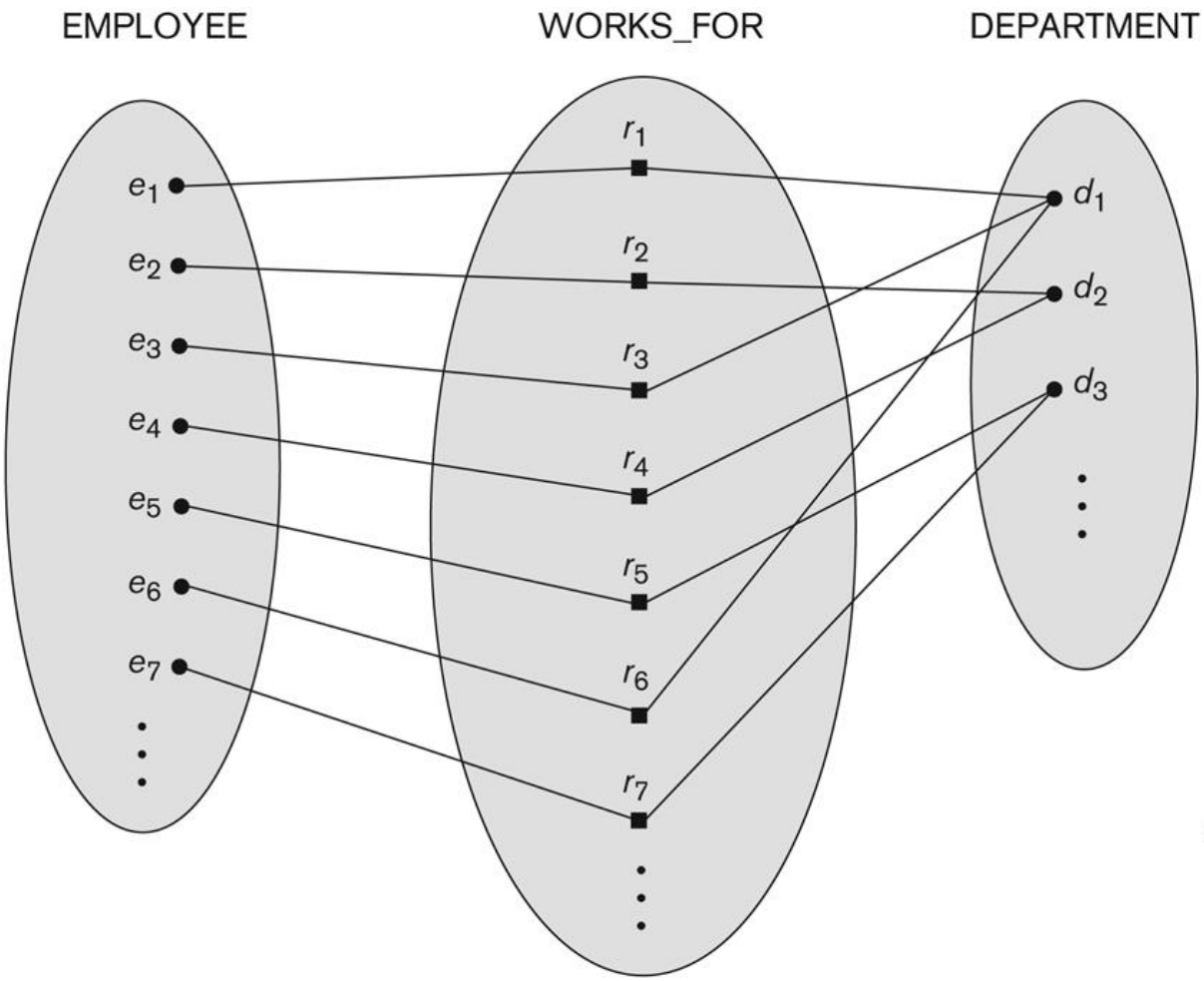


Figure 3.9
Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT.

Many-to-many (M:N) Relationship

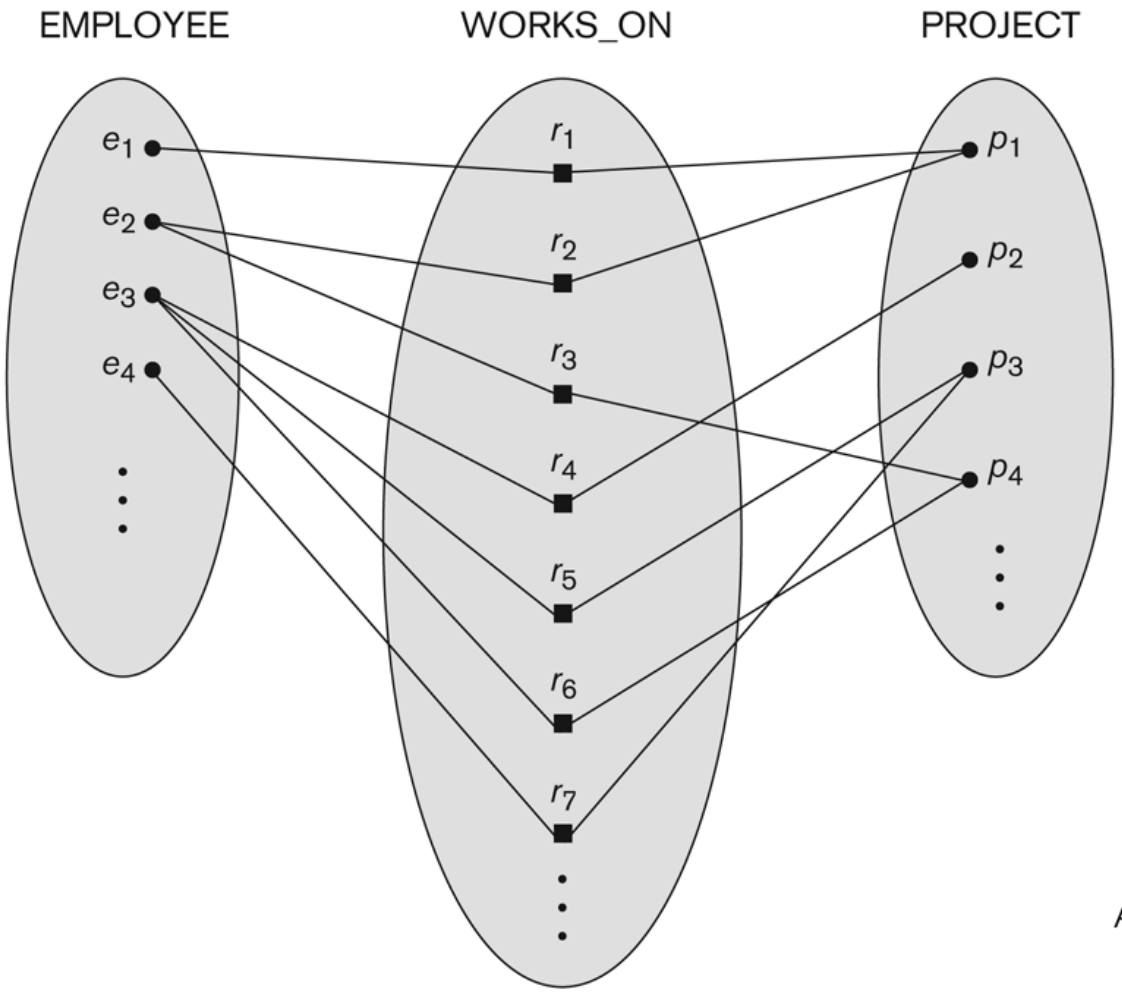


Figure 3.13
An M:N relationship, WORKS_ON.

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 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`

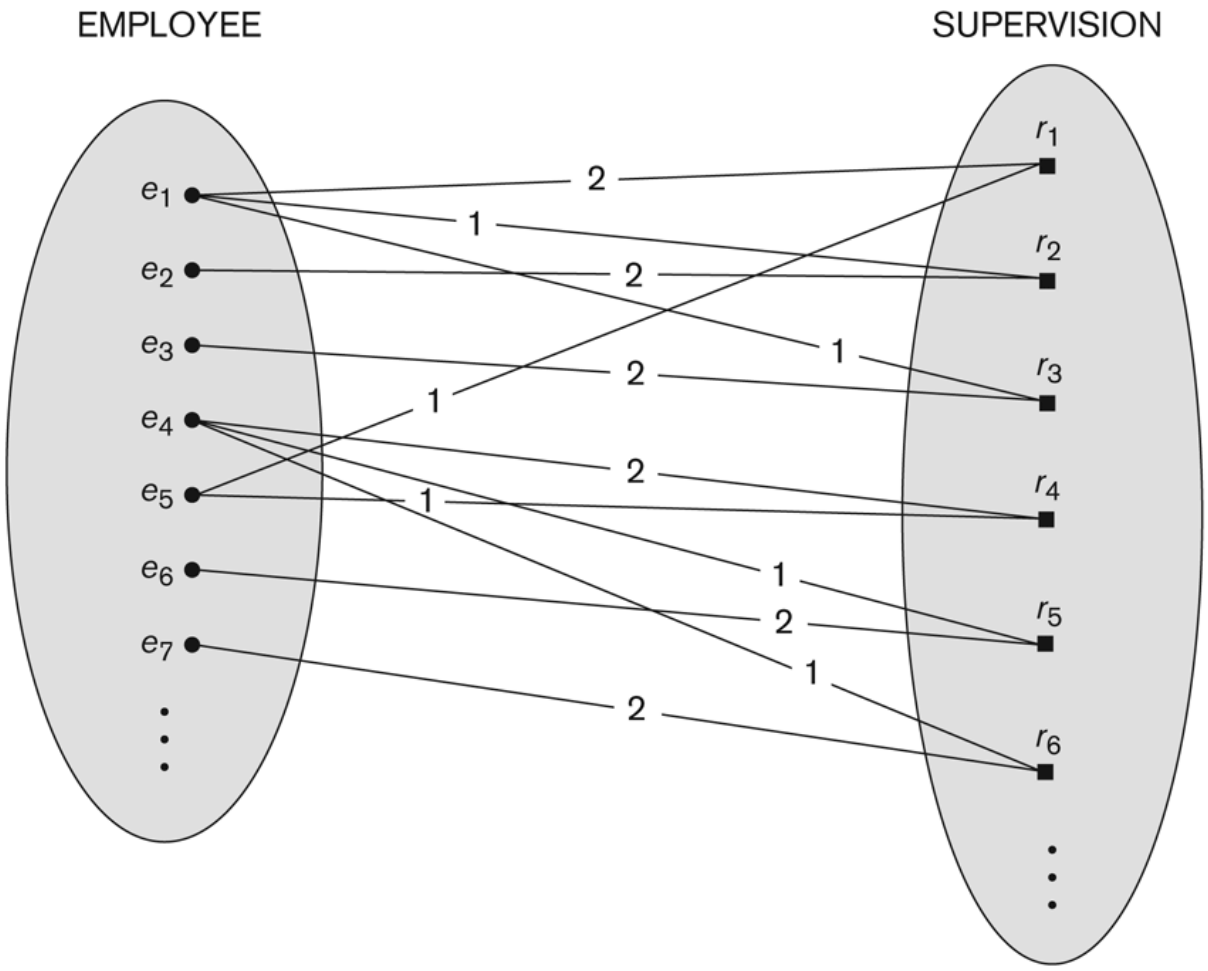


Figure 3.11
A recursive relationship SUPERVISION between EMPLOYEE in the *supervisor* role (1) and EMPLOYEE in the *subordinate* role (2).

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

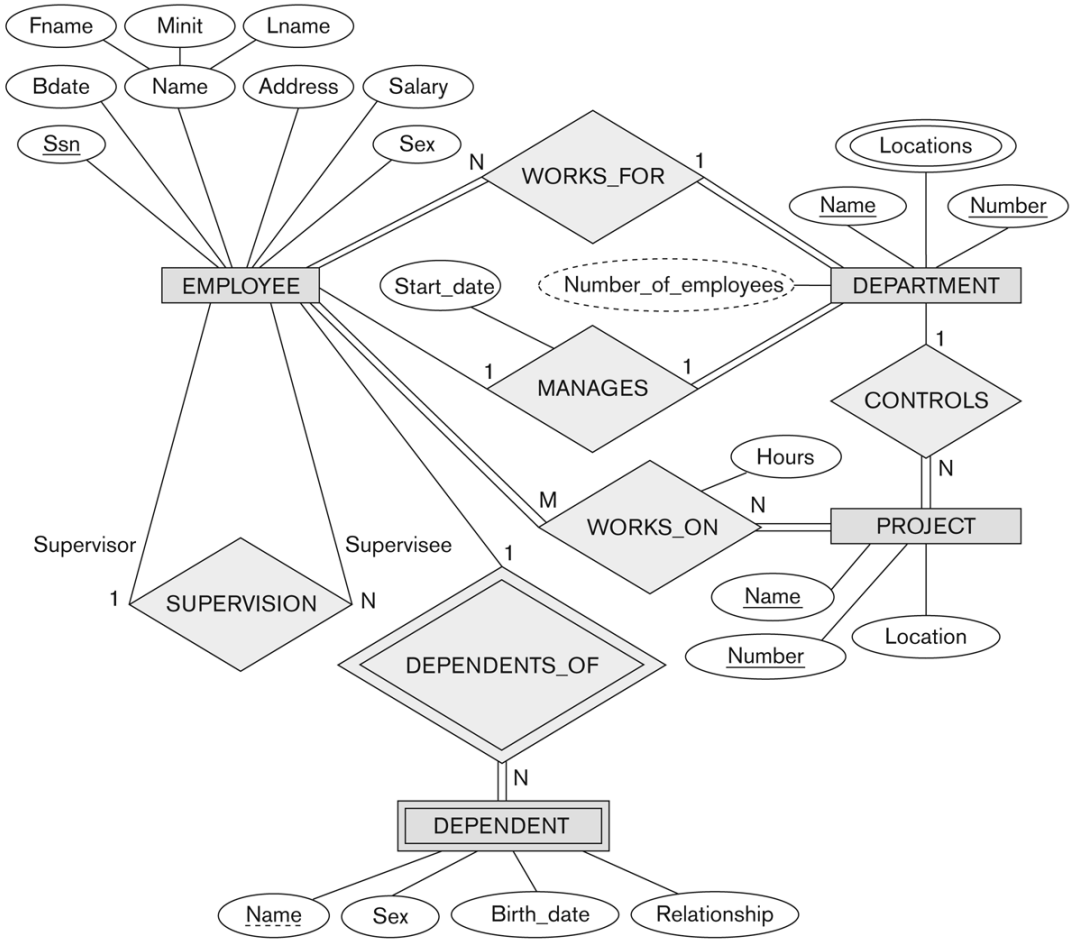


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

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

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

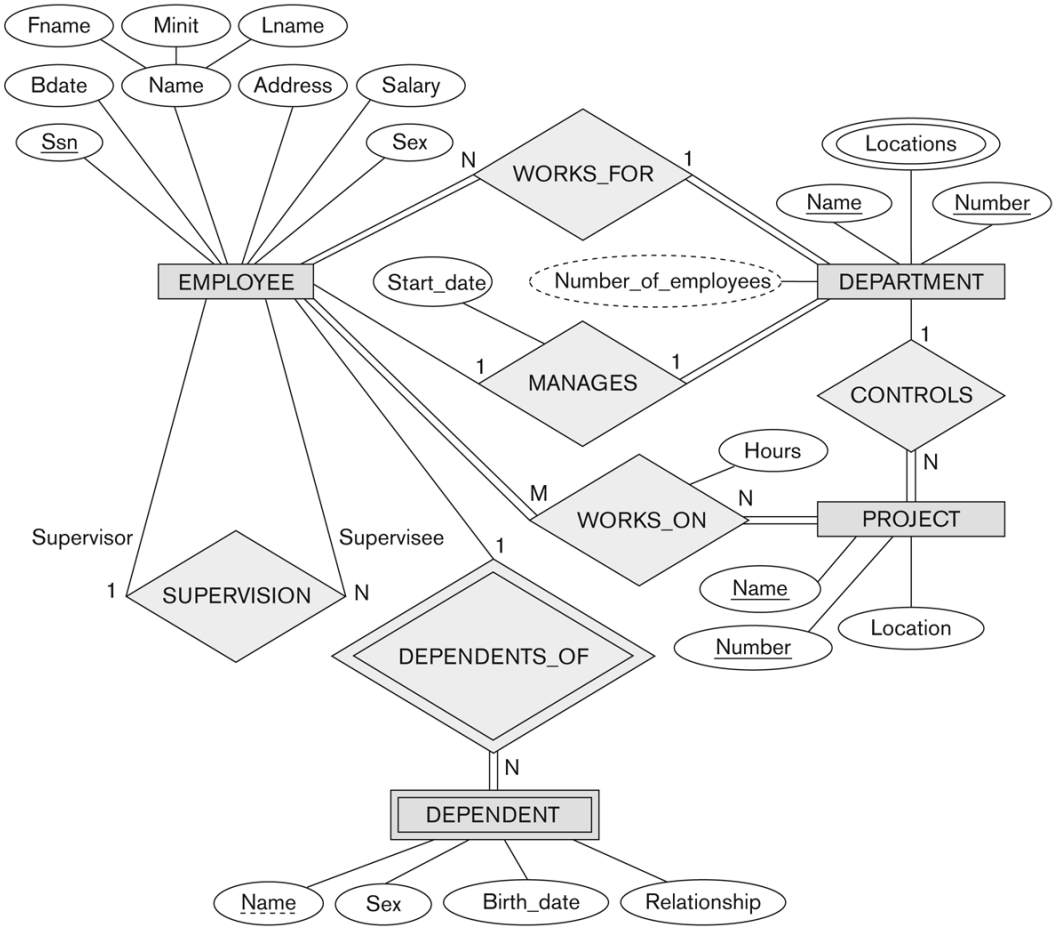


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

Notation for Constraints on Relationships

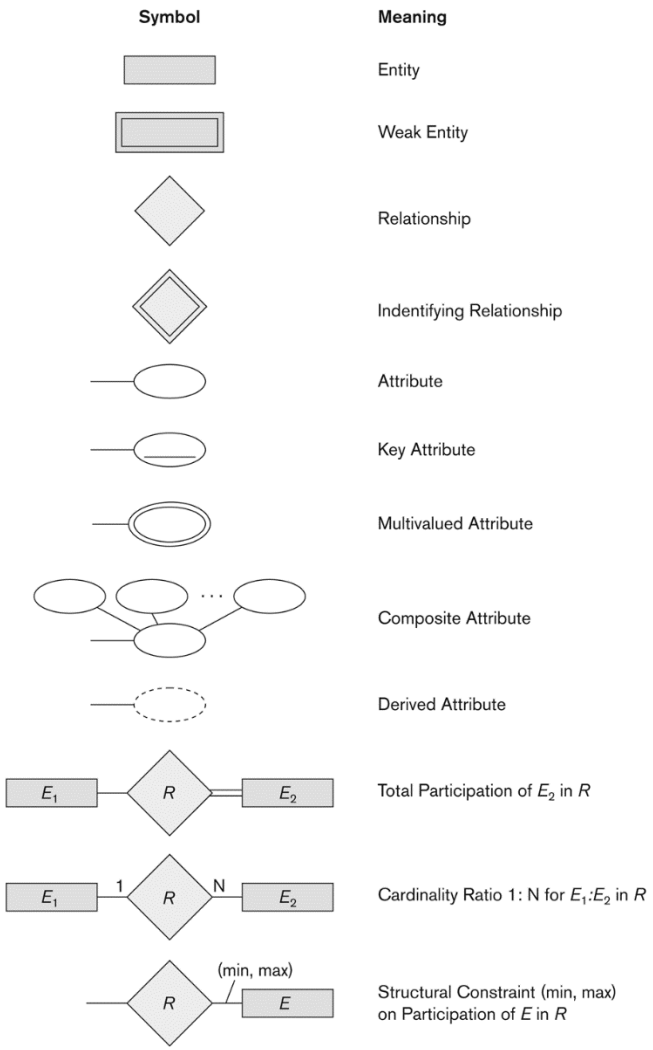
- Cardinality ratio (of a binary relationship): 1:1, 1:N, N:1, or M:N
  - Shown by placing appropriate numbers on the relationship edges.
- Participation constraint (on each participating entity type): total (called existence dependency) or partial.
  - Total shown by double line, partial by single line.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

Alternative 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 notation for ER diagrams

Figure 3.14
Summary of the notation for ER diagrams.

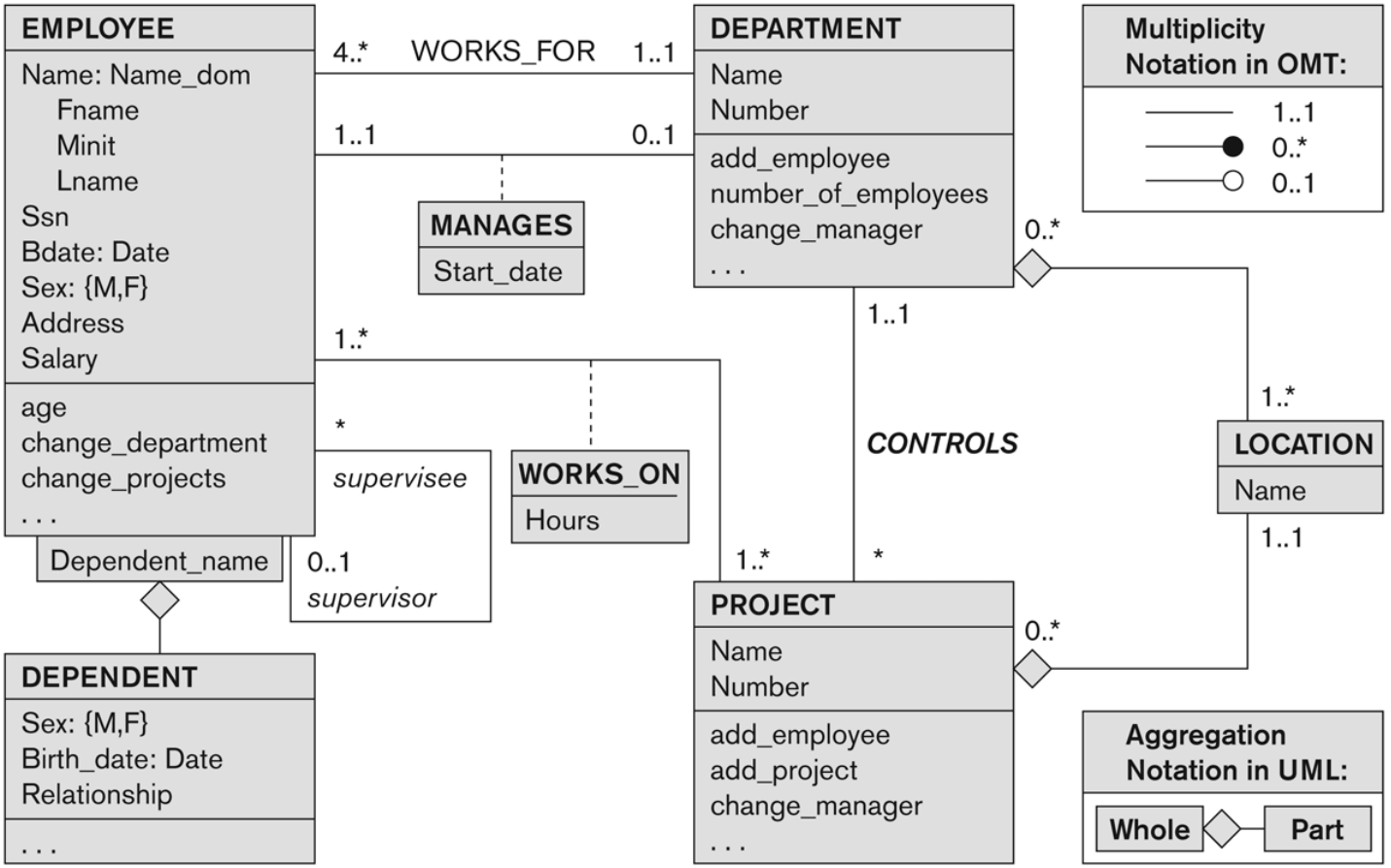


UML class diagrams

- 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 schema

Figure 3.16
The COMPANY conceptual schema in UML class diagram notation.



Other alternative diagrammatic notations

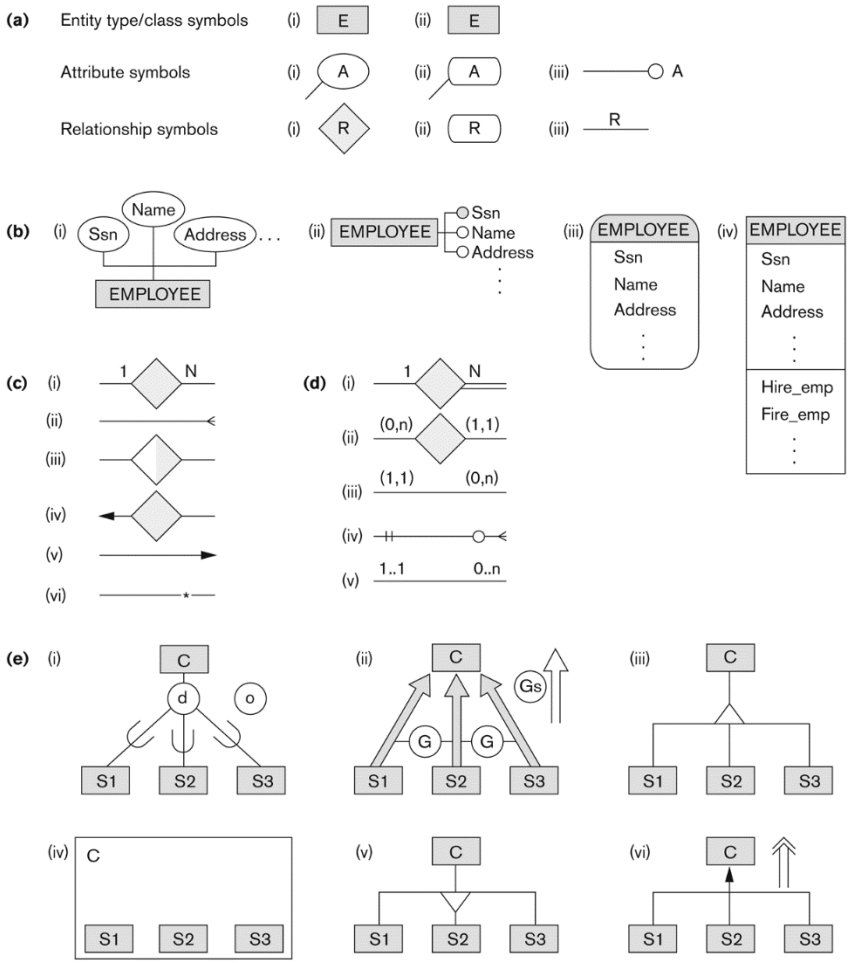


Figure A.1
Alternative notations. (a) Symbols for entity type/class, attribute, and relationship. (b) Displaying attributes. (c) Displaying cardinality ratios. (d) Various (min, max) notations. (e) Notations for displaying specialization/generalization.

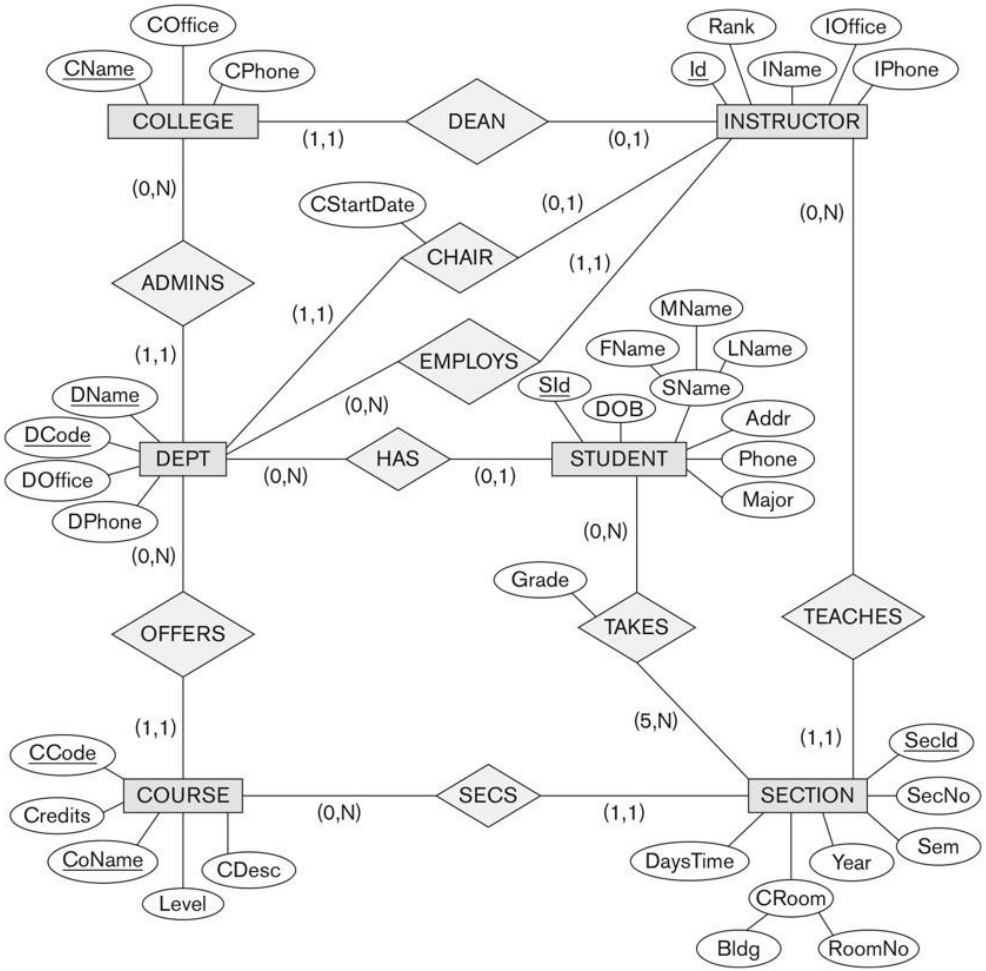
Relationships of Higher Degree

- Relationship types of degree 2 are called binary
- Relationship types of degree 3 are called ternary and of degree n are called n-ary
- Constraints are harder to specify for higher-degree relationships (n > 2) than for binary relationships

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



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, 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

Some of the Automated Database Design Tools (Note: Not all may be on the market now)

COMPANY	TOOL	FUNCTIONALITY
Embarcadero Technologies	ER Studio	Database Modeling in ER and IDEF1X
	DB Artisan	Database administration, space and security management
Oracle	Developer 2000/Designer 2000	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++

Extended Entity-Relationship (EER) Model (in the next chapter)

- The entity relationship model in its original form did not support the specialization and generalization abstractions
- Next chapter illustrates how the ER model can be extended with
  - Type-subtype and set-subset relationships
  - Specialization/Generalization Hierarchies
  - Notation to display them in EER diagrams

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
- Alternative Notations – UML class diagrams, others
- Binary Relationship types and those of higher degree.