

22CS510 - Database Management Systems

Dr. Manimala.S

Associate Professor

Dept. of CS & E, SJCE, Mysuru

Unit II

**Data Modeling Using the Entity-Relationship
(ER) Model:**

**The Relational Data Model and Relational
Database Constraints:**

Unit II

Data Modeling Using the Entity-Relationship (ER)

Model: Using High- Level Conceptual Data Models for Database Design, A Sample Database Application, Entity Types, Entity Sets, Attributes, and Keys, Relationship Types, Relationship Sets, Roles, and Structural Constraints, Weak Entity Types, Refining the ER Design for the COMPANY Database, ER Diagrams, Naming Conventions, and Design Issues, Relationship Types of Degree Higher than Two, Relational Database Design Using ER-to-Relational Mapping

The Relational Data Model and Relational Database

Constraints: Relational Model Concepts, Relational Model Constraints and Relational Database Schemas, Update Operations, Transactions, and Dealing with Constraint Violations,

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
- Relationships of Higher Degree

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

Overview of Database Design Process

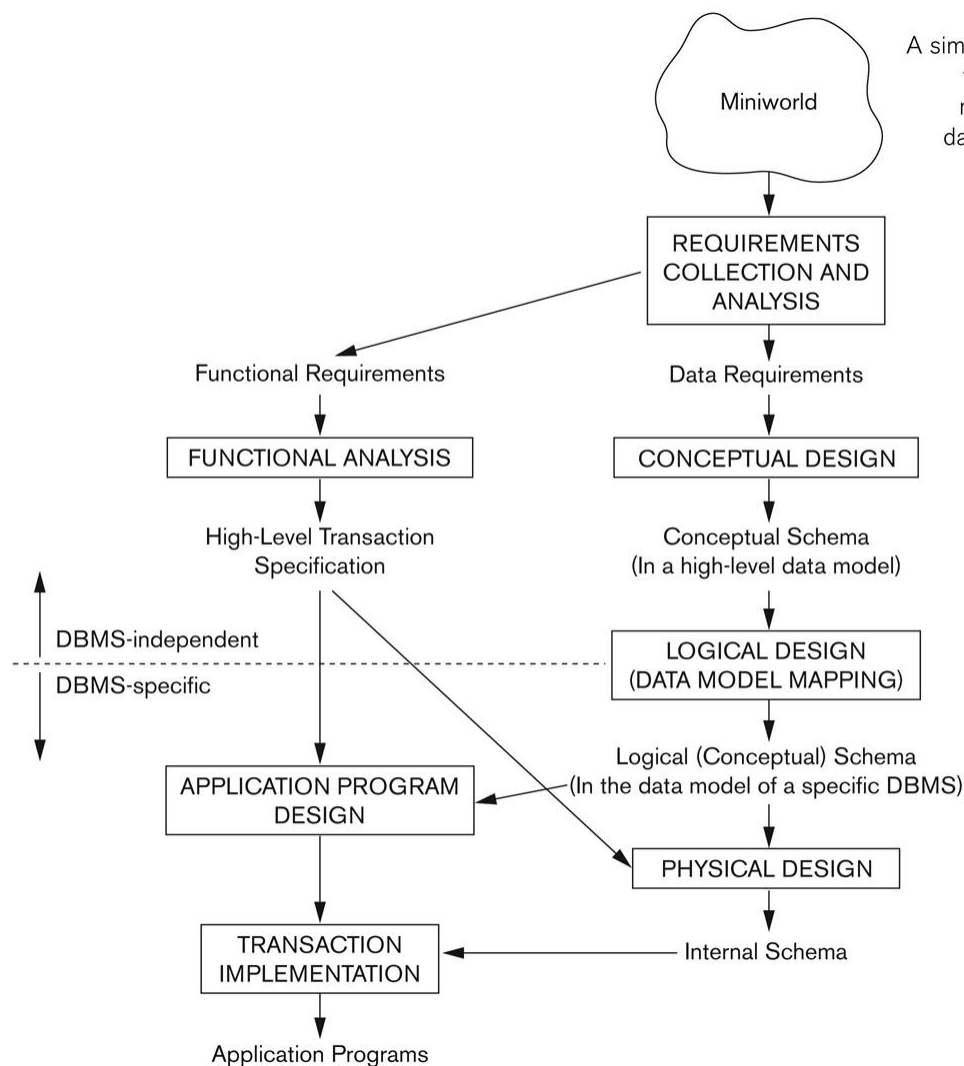


Figure 3.1

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)
- 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

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 (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}.

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

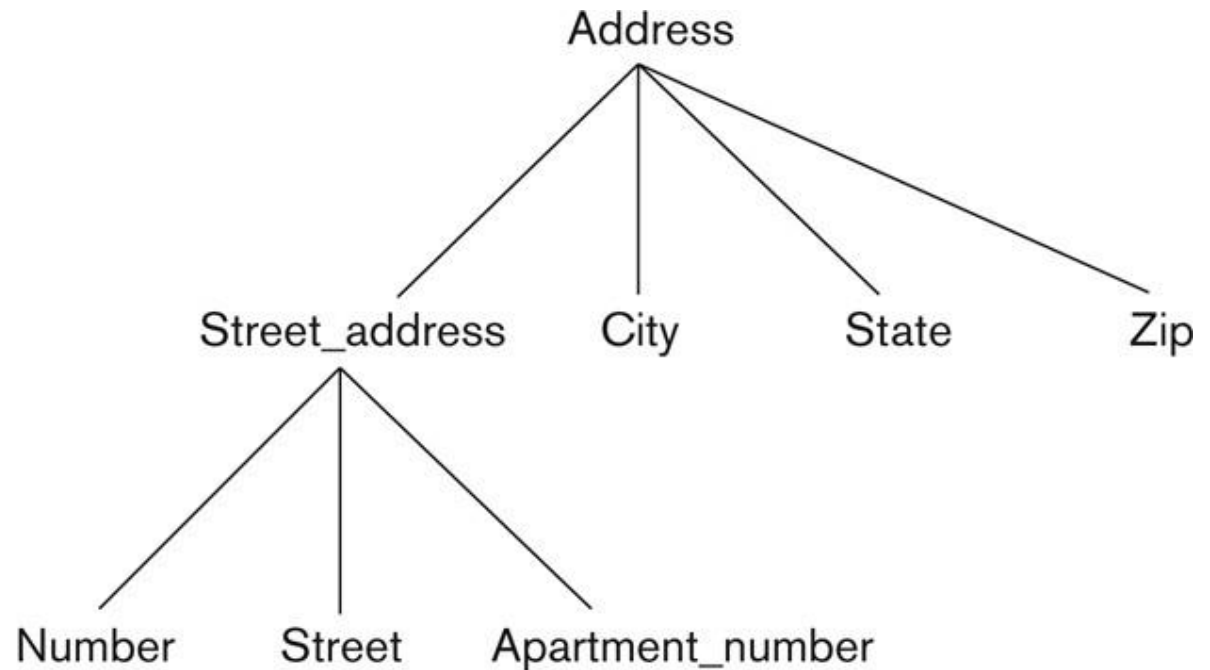


Figure 3.4

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 (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**
- 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

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.








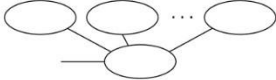



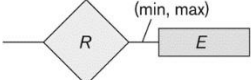
- We refer to the value of attribute A for entity e as $A(e)$.

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
	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

Entity Type CAR with two keys and a corresponding Entity Set

(a)

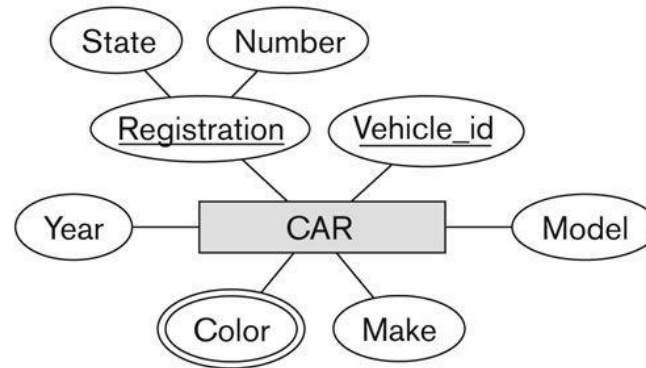


Figure 3.7

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}

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})

⋮

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

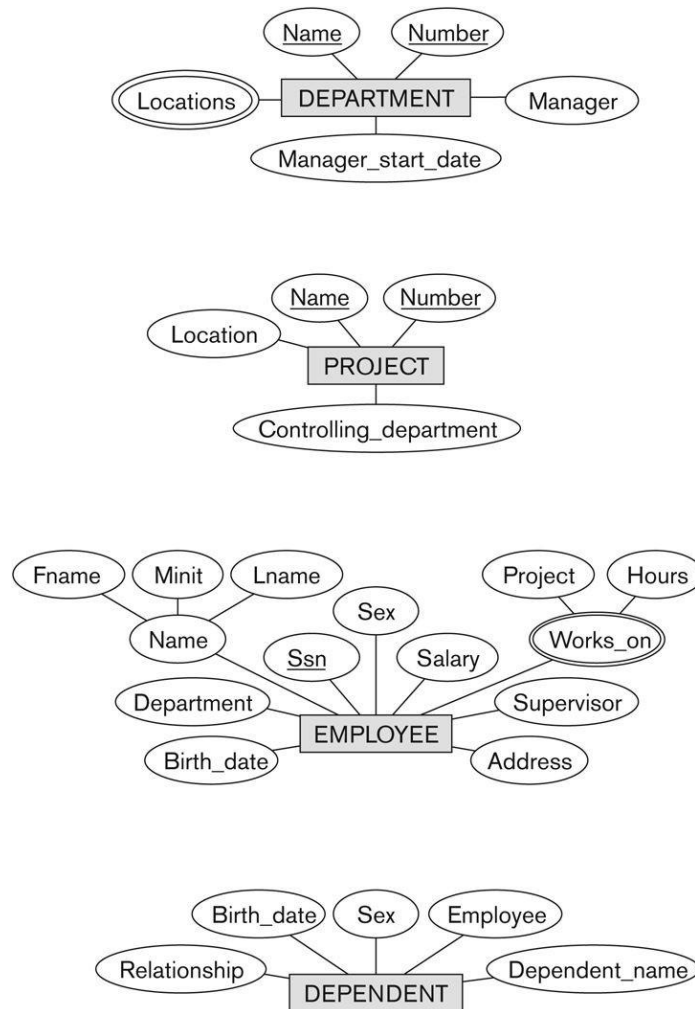


Figure 3.8

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 N:1 relationship between EMPLOYEE and DEPARTMENT

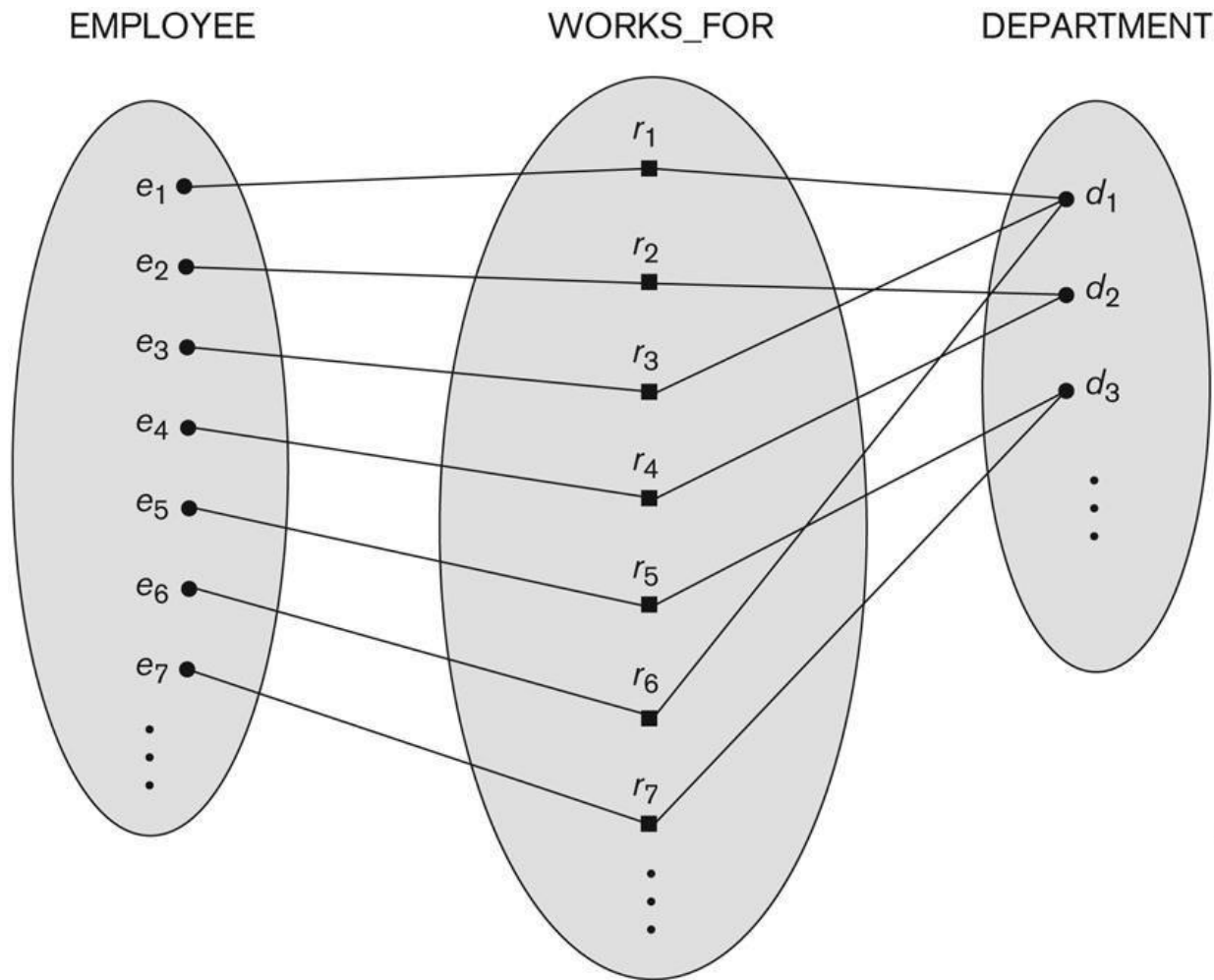


Figure 3.9

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

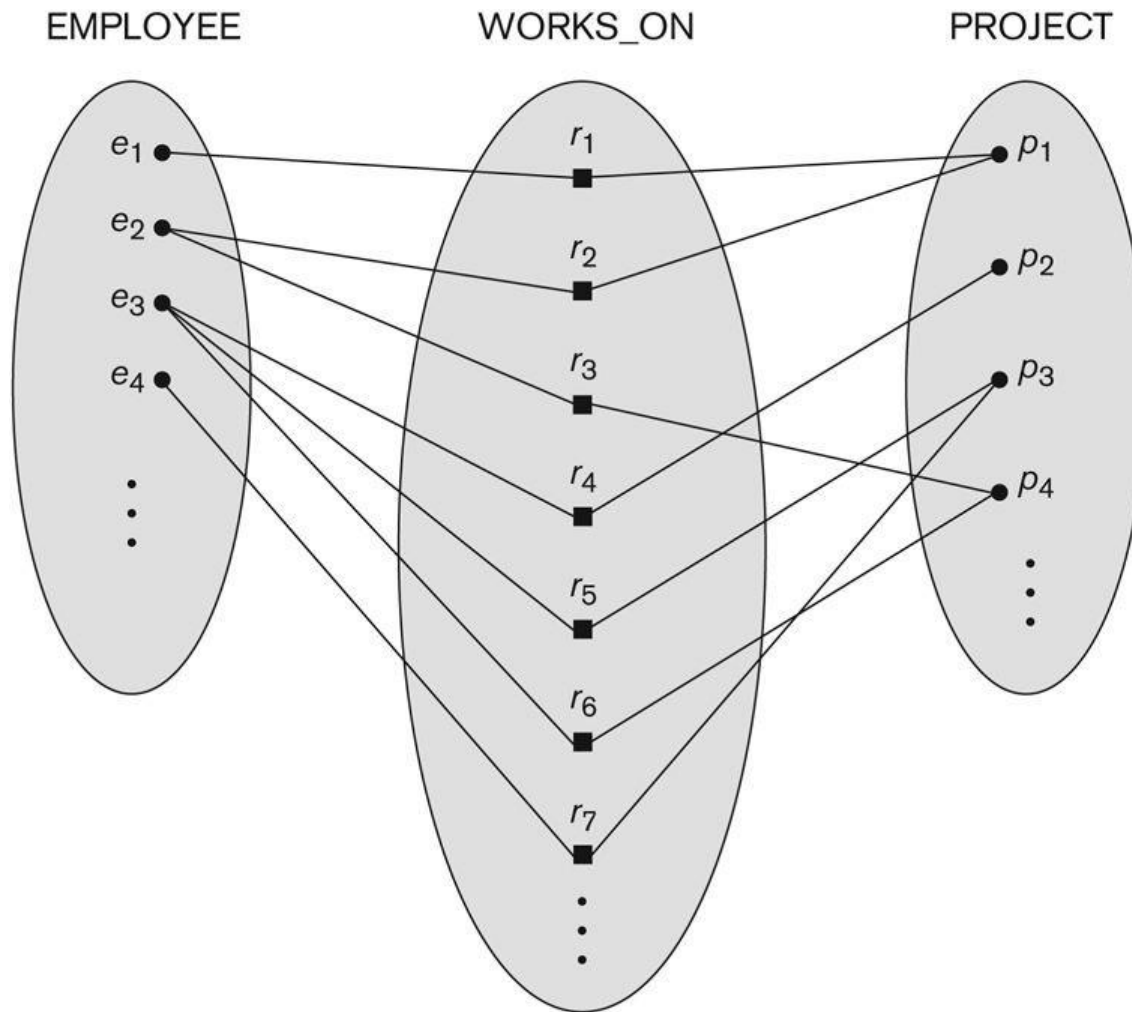


Figure 3.13
An M:N relationship,
WORKS_ON.

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

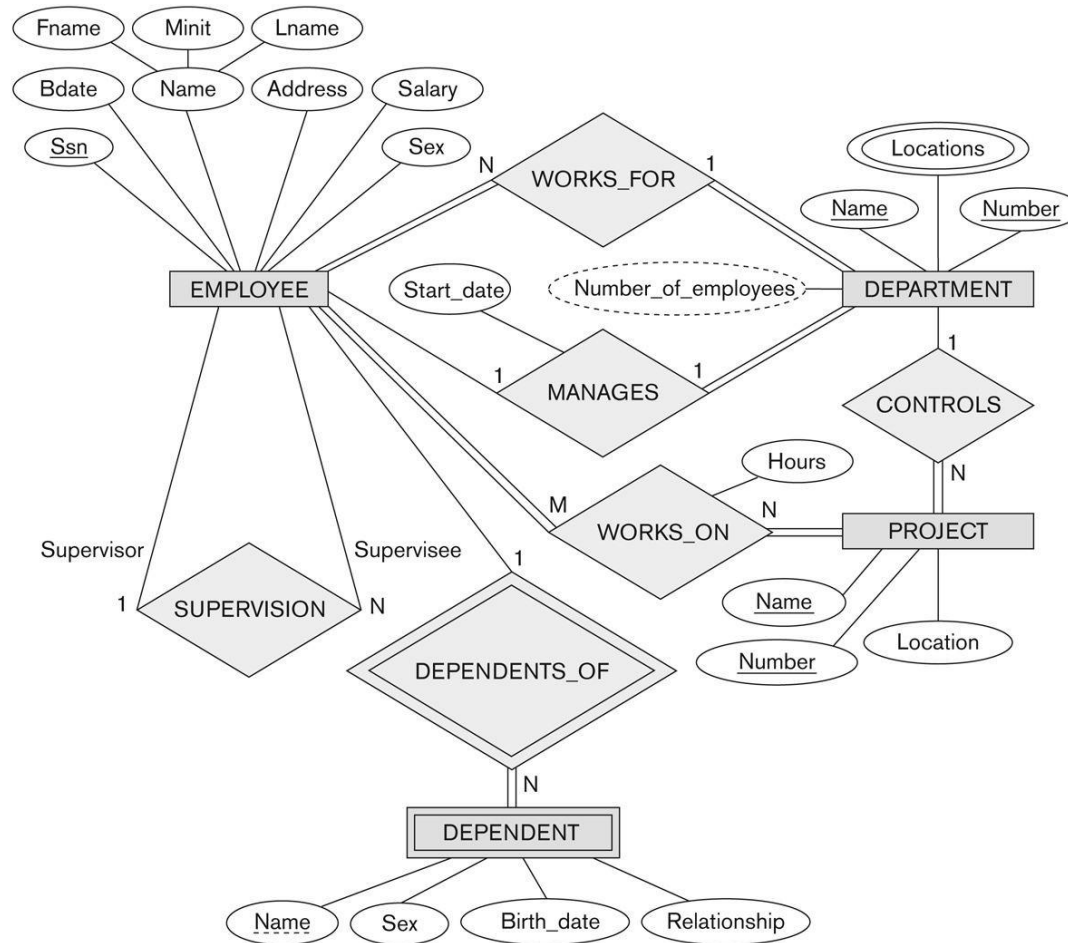


Figure 3.2

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

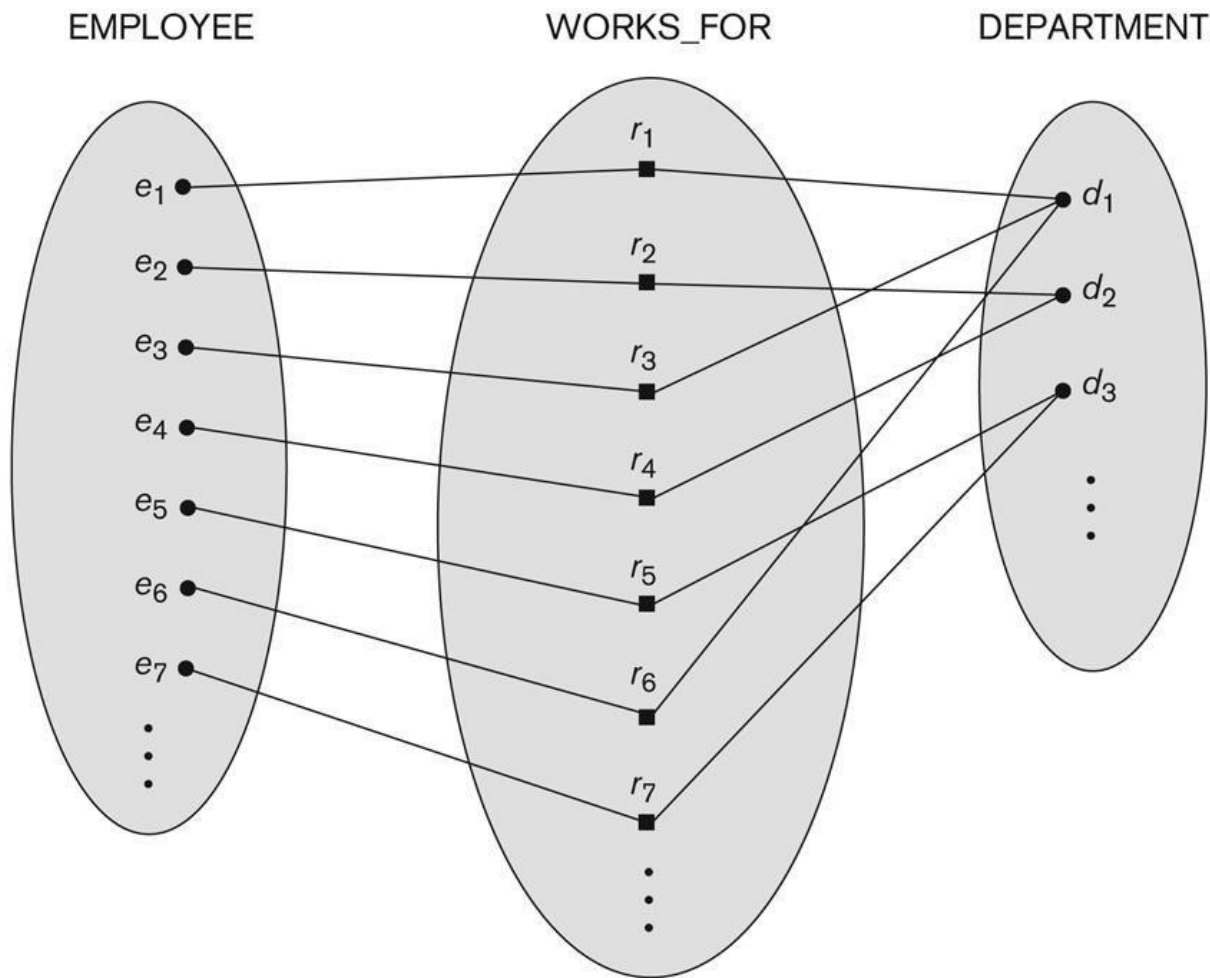


Figure 3.9

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

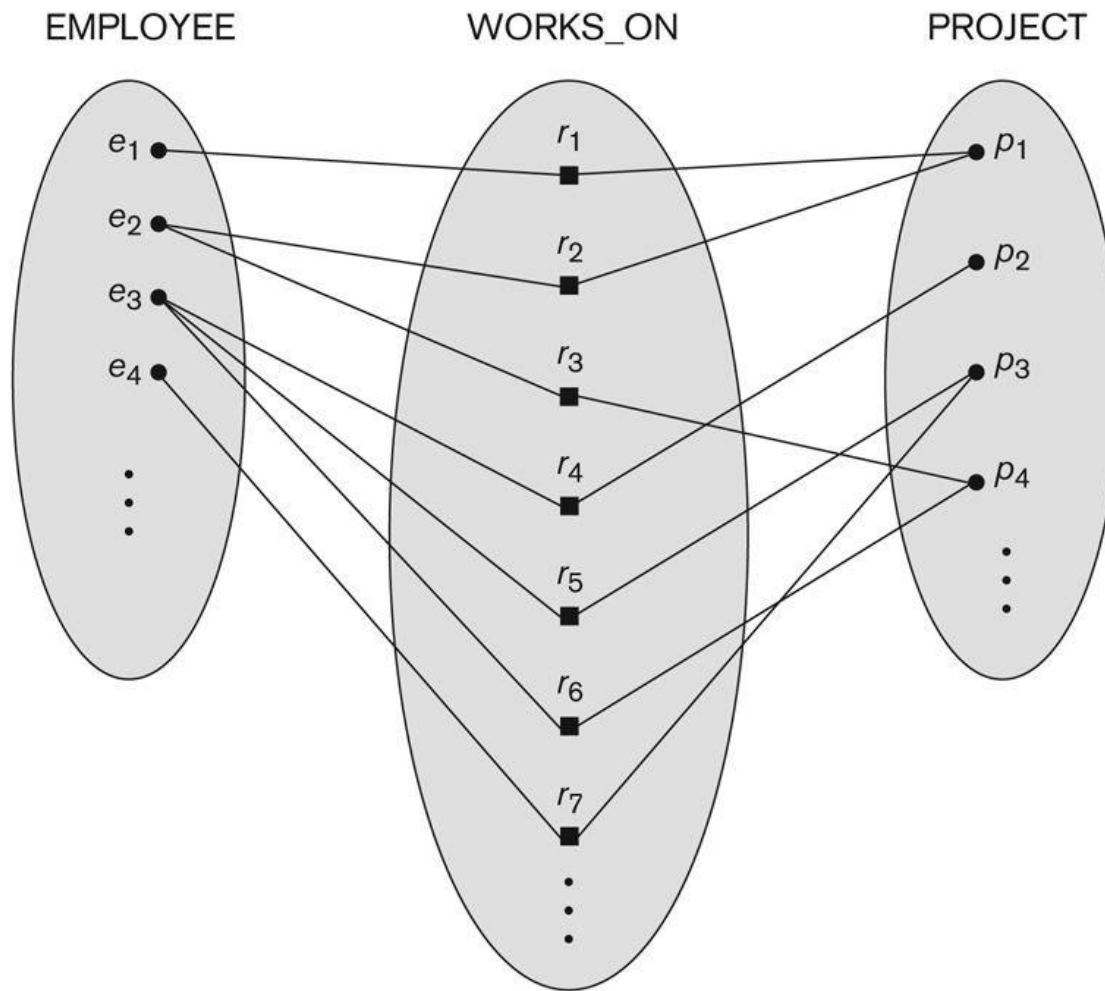


Figure 3.13
An M:N relationship,
WORKS_ON.

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`

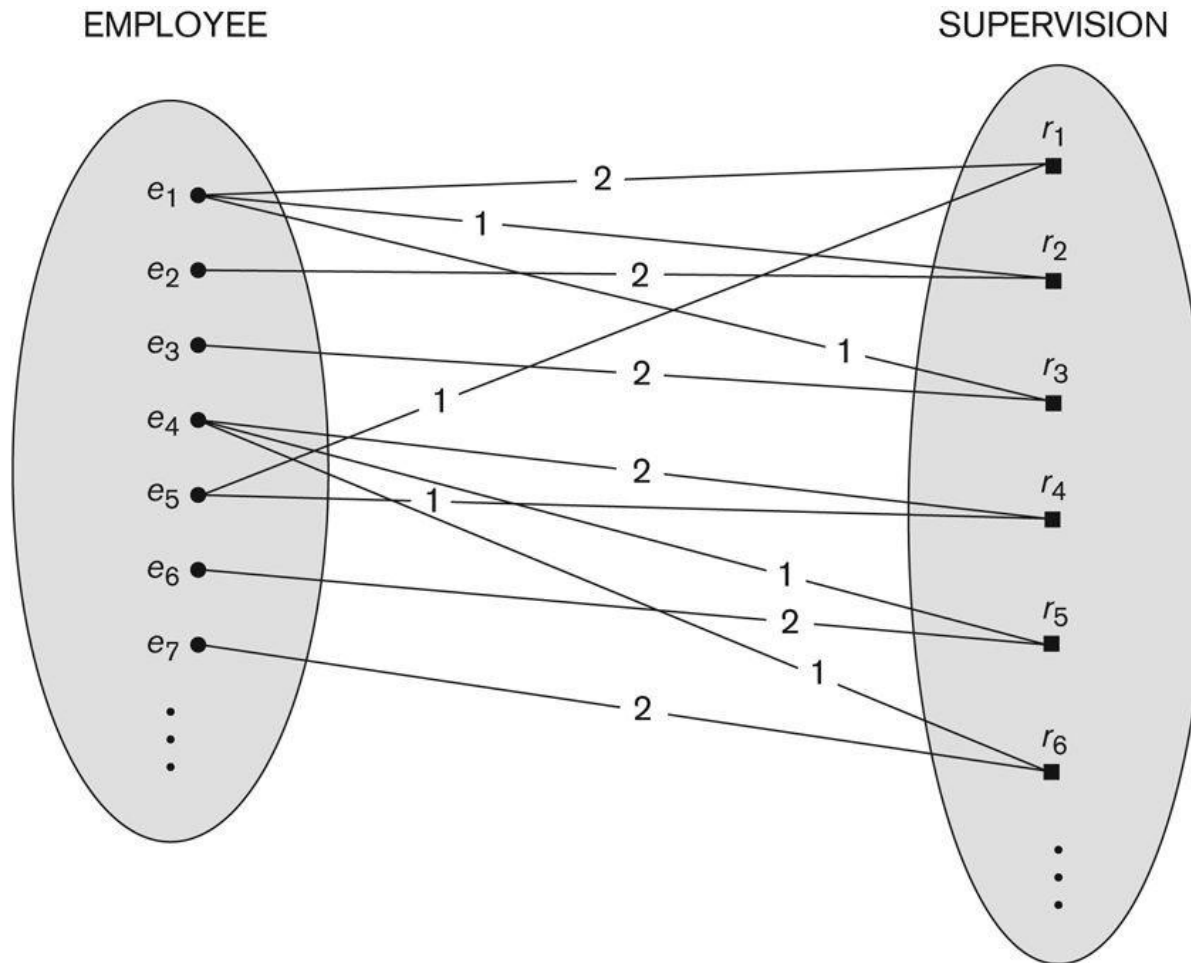


Figure 3.11

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)

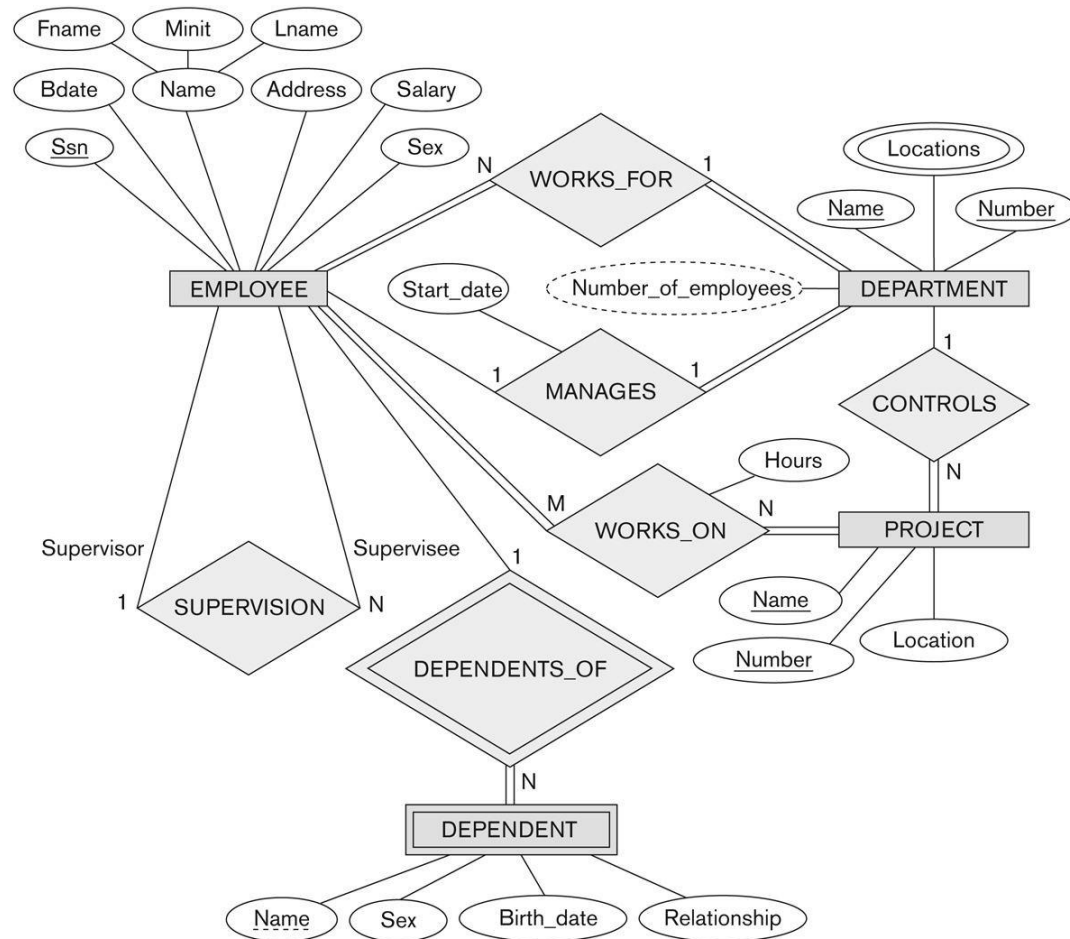


Figure 3.2

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

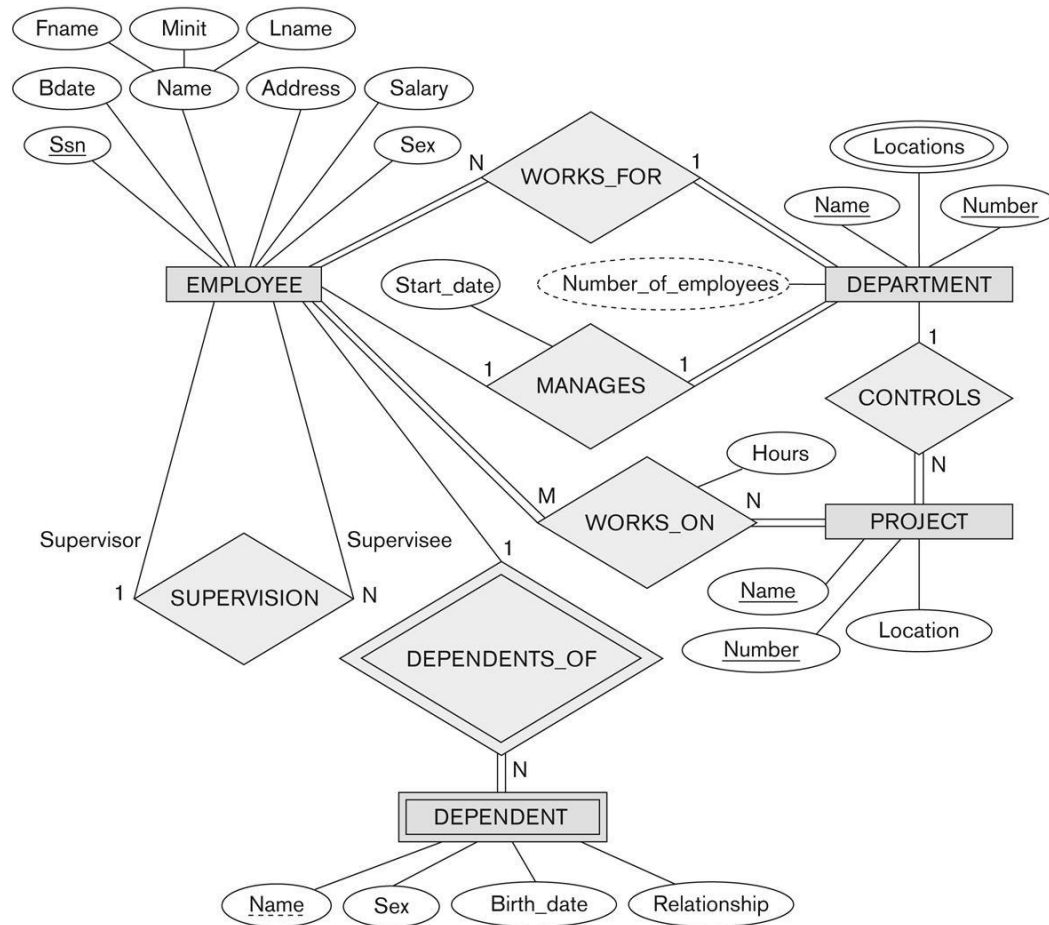


Figure 3.2

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

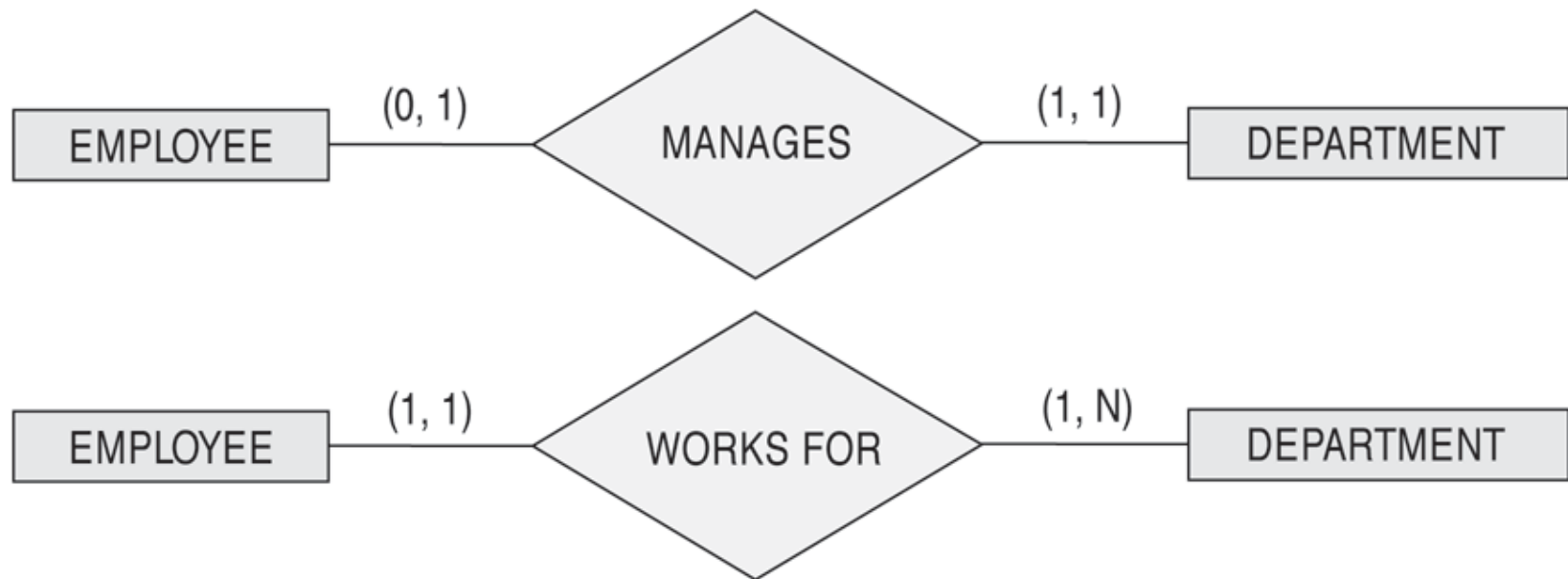
Notation for Constraints on Relationships

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

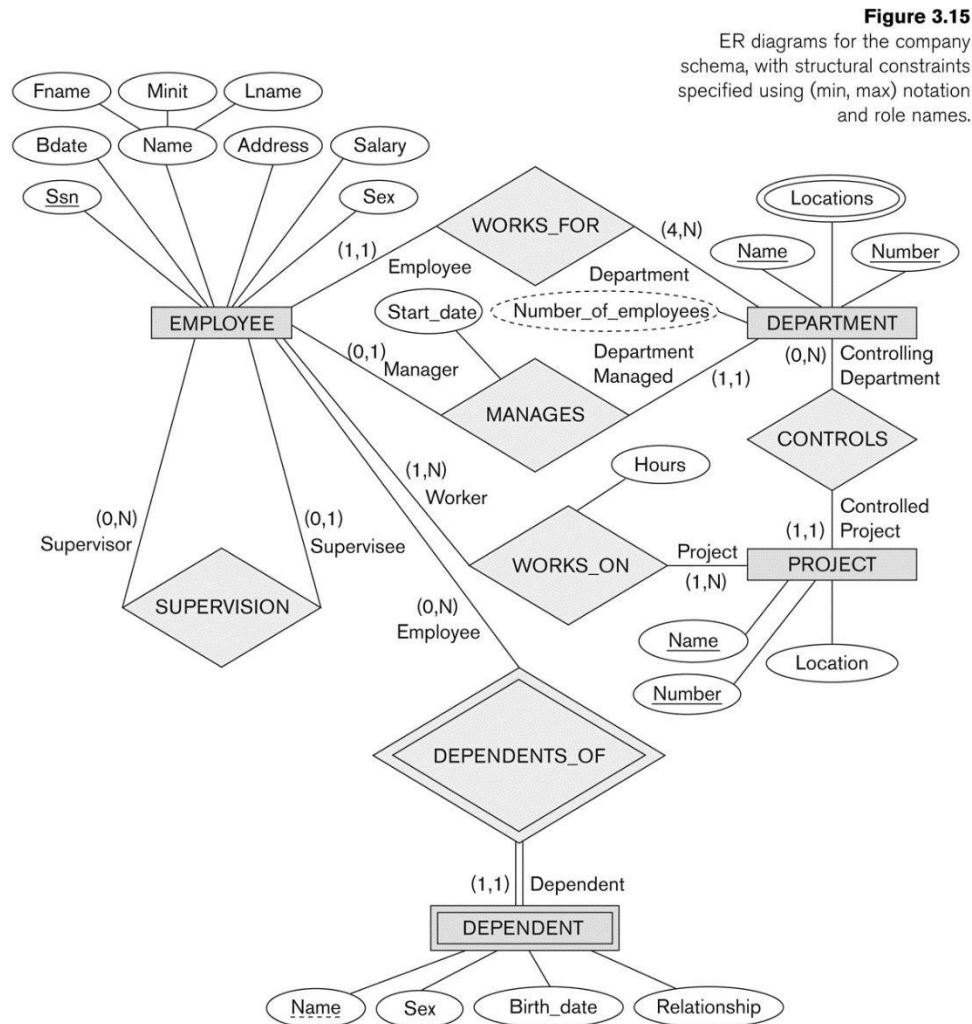
- Specified on each participation of an entity type E in a relationship type R
- Specifies that each entity e in E participates in at least *min* and at most *max* relationship instances in R
- Default(no constraint): min=0, max=n (signifying no limit)
- Must have $\text{min} \leq \text{max}$, $\text{min} \geq 0$, $\text{max} \geq 1$
- 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

COMPANY ER Schema Diagram using (min, max) notation








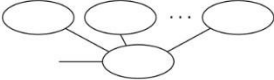

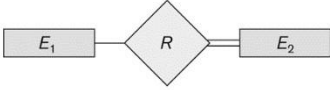

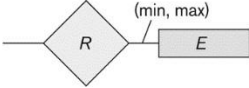


Alternative diagrammatic notation

- 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.

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

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

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 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

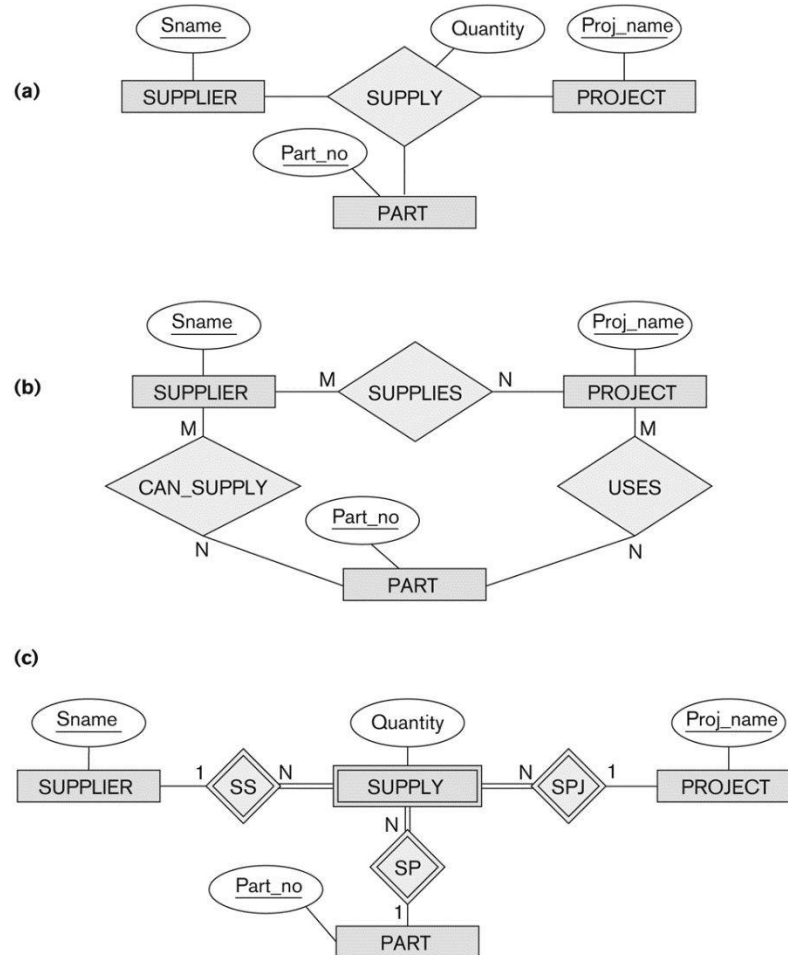


Figure 3.17

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

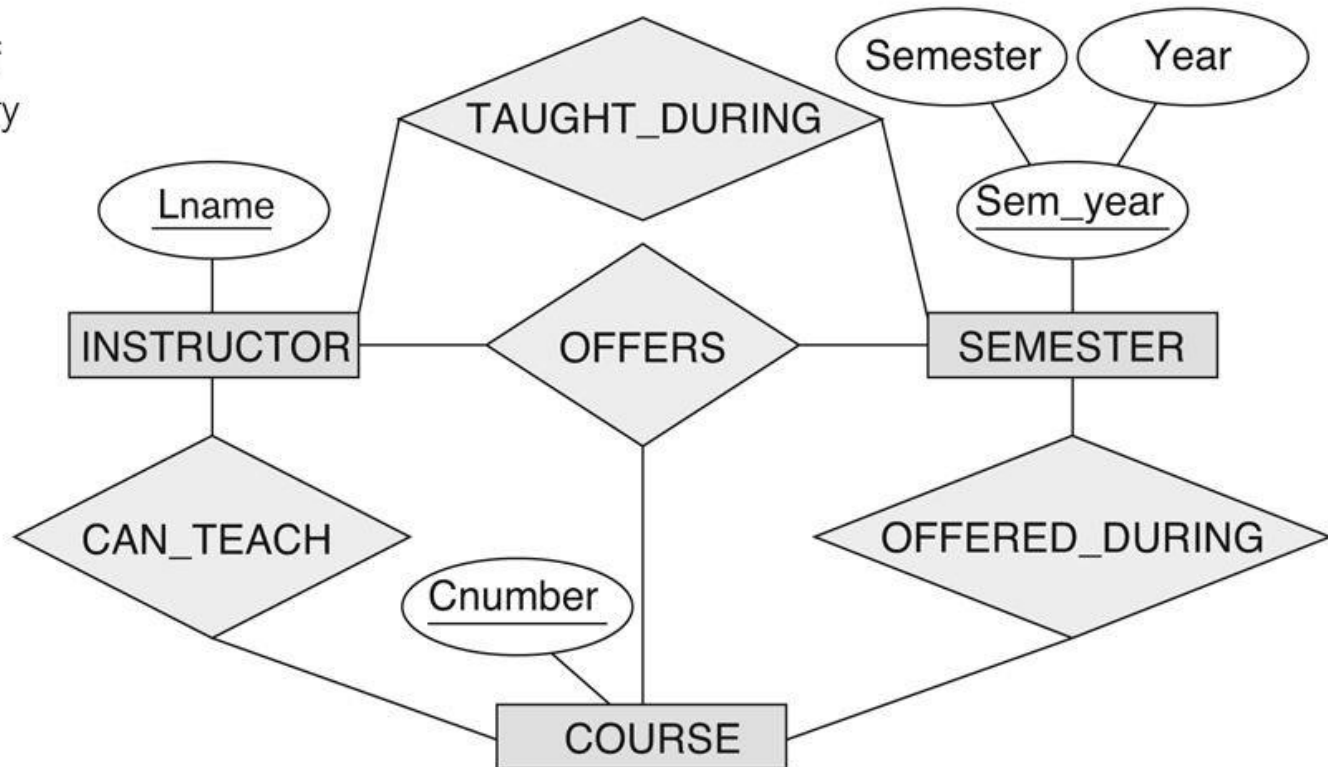
Discussion of n-ary relationships ($n > 2$)

- 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 binary relationship in Figure 3.18 (see next slide) can be derived from the ternary relationship OFFERS (based on the meaning of the relationships)

Another example of a ternary relationship

Figure 3.18

Another example of ternary versus binary relationship types.



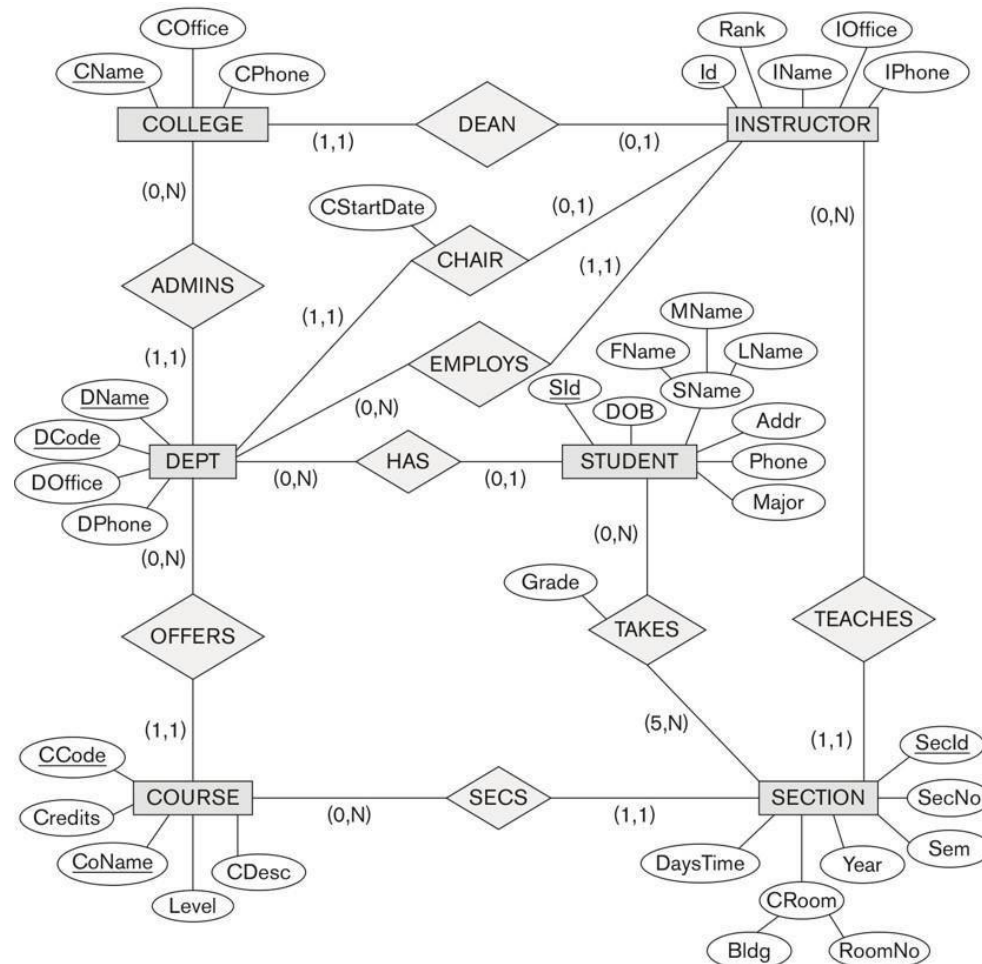
Displaying constraints on higher-degree relationships

- The (min, max) constraints can be displayed on the edges – however, they do not fully describe the constraints
- Displaying a 1, M, or N indicates additional constraints
 - An M or N indicates no constraint
 - A 1 indicates that an entity can participate in at most one relationship instance *that has a particular combination of the other participating entities*
- In general, both (min, max) and 1, M, or N are needed to describe fully the constraints
- Overall, the constraint specification is difficult and possibly ambiguous when we consider relationships of a degree higher than two.

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



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.

CHAPTER 5

The Relational Data Model and
Relational Database Constraints

Chapter Outline

- Relational Model Concepts
- Relational Model Constraints and Relational Database Schemas
- Update Operations and Dealing with Constraint Violations

Relational Model Concepts

- The relational Model of Data is based on the concept of a *Relation*
 - The strength of the relational approach to data management comes from the formal foundation provided by the theory of relations
- We review the essentials of the *formal relational model* in this chapter
- In *practice*, there is a *standard model* based on SQL – this is described in Chapters 6 and 7 as a language
- Note: There are several important differences between the *formal* model and the *practical* model, as we shall see

Relational Model Concepts

- A Relation is a mathematical concept based on the ideas of sets
- The model was first proposed by Dr. E.F. Codd of IBM Research in 1970 in the following paper:
 - "A Relational Model for Large Shared Data Banks," Communications of the ACM, June 1970
- The above paper caused a major revolution in the field of database management and earned Dr. Codd the coveted ACM Turing Award

Informal Definitions

- Informally, a **relation** looks like a **table** of values.
- A relation typically contains a **set of rows**.
- The data elements in each **row** represent certain facts that correspond to a real-world **entity** or **relationship**
 - In the formal model, rows are called **tuples**
- Each **column** has a column header that gives an indication of the meaning of the data items in that column
 - In the formal model, the column header is called an **attribute name** (or just **attribute**)

Example of a Relation

Relation Name

STUDENT

Attributes

Tuples

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25

Figure 5.1

The attributes and tuples of a relation STUDENT.

Informal Definitions

- Key of a Relation:
 - Each row has a value of a data item (or set of items) that uniquely identifies that row in the table
 - Called the *key*
 - In the STUDENT table, SSN is the key
- Sometimes row-ids or sequential numbers are assigned as keys to identify the rows in a table
 - Called *artificial key* or *surrogate key*

Formal Definitions - Schema

- The **Schema** (or description) of a Relation:
 - Denoted by $R(A_1, A_2, \dots, A_n)$
 - R is the **name** of the relation
 - The **attributes** of the relation are A_1, A_2, \dots, A_n
- Example:
CUSTOMER (Cust-id, Cust-name, Address, Phone#)
 - CUSTOMER is the relation name
 - Defined over the four attributes: Cust-id, Cust-name, Address, Phone#
- Each attribute has a **domain** or a set of valid values.
 - For example, the domain of Cust-id is 6 digit numbers.

Formal Definitions - Tuple

- A **tuple** is an ordered set of values (enclosed in angled brackets '< ... >')
- Each value is derived from an appropriate *domain*.
- A row in the CUSTOMER relation is a 4-tuple and would consist of four values, for example:
 - <632895, "John Smith", "101 Main St. Atlanta, GA 30332", "(404) 894-2000">
 - This is called a 4-tuple as it has 4 values
 - A tuple (row) in the CUSTOMER relation.
- A relation is a **set** of such tuples (rows)

Formal Definitions - Domain

- A **domain** has a logical definition:
 - Example: “USA_phone_numbers” are the set of 10 digit phone numbers valid in the U.S.
- A domain also has a data-type or a format defined for it.
 - The USA_phone_numbers may have a format: (ddd)ddd-dddd where each d is a decimal digit.
 - Dates have various formats such as year, month, date formatted as yyyy-mm-dd, or as dd mm,yyyy etc.
- The attribute name designates the role played by a domain in a relation:
 - Used to interpret the meaning of the data elements corresponding to that attribute
 - Example: The domain Date may be used to define two attributes named “Invoice-date” and “Payment-date” with different meanings

Formal Definitions - State

- The **relation state** is a subset of the Cartesian product of the domains of its attributes
 - each domain contains the set of all possible values the attribute can take.
- Example: attribute Cust-name is defined over the domain of character strings of maximum length 25
 - $\text{dom}(\text{Cust-name})$ is `varchar(25)`
- The role these strings play in the CUSTOMER relation is that of the *name of a customer*.

Formal Definitions - Summary

- Formally,
 - Given $R(A_1, A_2, \dots, A_n)$
 - $r(R) \subset \text{dom}(A_1) \times \text{dom}(A_2) \times \dots \times \text{dom}(A_n)$
- $R(A_1, A_2, \dots, A_n)$ is the **schema** of the relation
- R is the **name** of the relation
- A_1, A_2, \dots, A_n are the **attributes** of the relation
- $r(R)$: a specific **state** (or "value" or "population") of relation R – this is a *set of tuples* (rows)
 - $r(R) = \{t_1, t_2, \dots, t_n\}$ where each t_i is an n -tuple
 - $t_i = \langle v_1, v_2, \dots, v_n \rangle$ where each v_j *element-of* $\text{dom}(A_j)$

Formal Definitions - Example

- Let $R(A1, A2)$ be a relation schema:
 - Let $\text{dom}(A1) = \{0,1\}$
 - Let $\text{dom}(A2) = \{a,b,c\}$
- Then: $\text{dom}(A1) \times \text{dom}(A2)$ is all possible combinations:
 $\{ \langle 0,a \rangle, \langle 0,b \rangle, \langle 0,c \rangle, \langle 1,a \rangle, \langle 1,b \rangle, \langle 1,c \rangle \}$
- The relation state $r(R) \subset \text{dom}(A1) \times \text{dom}(A2)$
- For example: $r(R)$ could be $\{ \langle 0,a \rangle, \langle 0,b \rangle, \langle 1,c \rangle \}$
 - this is one possible state (or “population” or “extension”) r of the relation R , defined over $A1$ and $A2$.
 - It has three 2-tuples: $\langle 0,a \rangle, \langle 0,b \rangle, \langle 1,c \rangle$

Definition Summary

<u>Informal Terms</u>		<u>Formal Terms</u>
Table		Relation
Column Header		Attribute
All possible Column Values		Domain
Row		Tuple
Table Definition		Schema of a Relation
Populated Table		State of the Relation

Example – A relation STUDENT

Relation Name						
STUDENT						
Attributes						
Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25

Figure 5.1

The attributes and tuples of a relation STUDENT.

Characteristics Of Relations

- Ordering of tuples in a relation $r(R)$:
 - The tuples are *not considered to be ordered*, even though they appear to be in the tabular form.
- Ordering of attributes in a relation schema R (and of values within each tuple):
 - We will consider the attributes in $R(A_1, A_2, \dots, A_n)$ and the values in $t = \langle v_1, v_2, \dots, v_n \rangle$ to be ordered .
 - (However, a more general alternative definition of relation does not require this ordering. It includes both the name and the value for each of the attributes).
 - Example: $t = \{ \langle \text{name}, \text{"John"} \rangle, \langle \text{SSN}, 123456789 \rangle \}$
 - This representation may be called as “self-describing”.

Same state as previous Figure (but with different order of tuples)

Figure 5.2

The relation STUDENT from Figure 5.1 with a different order of tuples.

STUDENT

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21

Characteristics Of Relations

- Values in a tuple:
 - All values are considered atomic (indivisible).
 - Each value in a tuple must be from the domain of the attribute for that column
 - If tuple $t = \langle v_1, v_2, \dots, v_n \rangle$ is a tuple (row) in the relation state r of $R(A_1, A_2, \dots, A_n)$
 - Then each v_i must be a value from $dom(A_i)$
- A special **null** value is used to represent values that are unknown or not available or inapplicable in certain tuples.

Characteristics Of Relations

- Notation:
 - We refer to **component values** of a tuple t by:
 - $t[A_i]$ or $t.A_i$
 - This is the value v_i of attribute A_i for tuple t
 - Similarly, $t[A_u, A_v, \dots, A_w]$ refers to the subtuple of t containing the values of attributes A_u, A_v, \dots, A_w , respectively in t

CONSTRAINTS

Constraints determine which values are permissible and which are not in the database.

They are of three main types:

1. Inherent or Implicit Constraints: These are based on the data model itself. (E.g., relational model does not allow a list as a value for any attribute)

2. Schema-based or Explicit Constraints: They are expressed in the schema by using the facilities provided by the model. (E.g., max. cardinality ratio constraint in the ER model)

3. Application based or semantic constraints: These are beyond the expressive power of the model and must be specified and enforced by the application programs.

Relational Integrity Constraints

- Constraints are **conditions** that must hold on **all** valid relation states.
- There are three *main types* of (explicit schema-based) constraints that can be expressed in the relational model:
 - **Key** constraints
 - **Entity integrity** constraints
 - **Referential integrity** constraints
- Another schema-based constraint is the **domain** constraint
 - Every value in a tuple must be from the *domain of its attribute* (or it could be **null**, if allowed for that attribute)

Key Constraints

- **Superkey of R:**
 - Is a set of attributes SK of R with the following condition:
 - No two tuples in any valid relation state $r(R)$ will have the same value for SK
 - That is, for any distinct tuples $t1$ and $t2$ in $r(R)$, $t1[SK] \neq t2[SK]$
 - This condition must hold in *any valid state* $r(R)$
- **Key of R:**
 - A "minimal" superkey
 - That is, a key is a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey (does not possess the superkey uniqueness property)
- A Key is a Superkey but not vice versa

Key Constraints (continued)

- Example: Consider the CAR relation schema:
 - CAR(State, Reg#, SerialNo, Make, Model, Year)
 - CAR has two keys:
 - Key1 = {State, Reg#}
 - Key2 = {SerialNo}
 - Both are also superkeys of CAR
 - {SerialNo, Make} is a superkey but *not* a key.
- In general:
 - Any *key* is a *superkey* (but not vice versa)
 - Any set of attributes that *includes a key* is a *superkey*
 - A *minimal* superkey is also a key

Key Constraints (continued)

- If a relation has several **candidate keys**, one is chosen arbitrarily to be the **primary key**.
 - The primary key attributes are underlined.
- Example: Consider the CAR relation schema:
 - CAR(State, Reg#, SerialNo, Make, Model, Year)
 - We chose SerialNo as the primary key
- The primary key value is used to *uniquely identify* each tuple in a relation
 - Provides the tuple identity
- Also used to *reference* the tuple from another tuple
 - General rule: Choose as primary key the smallest of the candidate keys (in terms of size)
 - Not always applicable – choice is sometimes subjective

CAR table with two candidate keys – LicenseNumber chosen as Primary Key

CAR

<u>License_number</u>	Engine_serial_number	Make	Model	Year
Texas ABC-739	A69352	Ford	Mustang	02
Florida TVP-347	B43696	Oldsmobile	Cutlass	05
New York MPO-22	X83554	Oldsmobile	Delta	01
California 432-TFY	C43742	Mercedes	190-D	99
California RSK-629	Y82935	Toyota	Camry	04
Texas RSK-629	U028365	Jaguar	XJS	04

Figure 5.4

The CAR relation, with two candidate keys: License_number and Engine_serial_number.

Relational Database Schema

- **Relational Database Schema:**
 - A set S of relation schemas that belong to the same database.
 - S is the name of the whole **database schema**
 - $S = \{R_1, R_2, \dots, R_n\}$ and a set IC of integrity constraints.
 - R_1, R_2, \dots, R_n are the names of the individual **relation schemas** within the database S
- Following slide shows a COMPANY database schema with 6 relation schemas

COMPANY Database Schema

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	------------	-------	---------	-----	--------	-----------	-----

DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
-------	----------------	---------	----------------

DEPT_LOCATIONS

<u>Dnumber</u>	<u>Dlocation</u>
----------------	------------------

PROJECT

Pname	<u>Pnumber</u>	Plocation	Dnum
-------	----------------	-----------	------

WORKS_ON

<u>Essn</u>	<u>Pno</u>	Hours
-------------	------------	-------

DEPENDENT

<u>Essn</u>	<u>Dependent_name</u>	Sex	Bdate	Relationship
-------------	-----------------------	-----	-------	--------------

Figure 5.5
Schema diagram for
the COMPANY
relational database
schema.

Relational Database State

- A **relational database state** DB of S is a set of relation states $DB = \{r_1, r_2, \dots, r_m\}$ such that each r_i is a state of R_i and such that the r_i relation states satisfy the integrity constraints specified in IC.
- A relational database *state* is sometimes called a relational database *snapshot* or *instance*.
- We will not use the term *instance* since it also applies to single tuples.
- A database state that does not meet the constraints is an invalid state

Populated database state

- Each *relation* will have many tuples in its current relation state
- The *relational database state* is a union of all the individual relation states
- Whenever the database is changed, a new state arises
- Basic operations for changing the database:
 - INSERT a new tuple in a relation
 - DELETE an existing tuple from a relation
 - MODIFY an attribute of an existing tuple
- Next slide (Fig. 5.6) shows an example state for the COMPANY database schema shown in Fig. 5.5.

Populated database state for COMPANY

Figure 5.6

One possible database state for the COMPANY relational database schema.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

Entity Integrity

■ Entity Integrity:

- The *primary key attributes* PK of each relation schema R in S cannot have null values in any tuple of $r(R)$.
 - This is because primary key values are used to *identify* the individual tuples.
 - $t[PK] \neq \text{null}$ for any tuple t in $r(R)$
 - If PK has several attributes, null is not allowed in any of these attributes
- Note: Other attributes of R may be constrained to disallow null values, even though they are not members of the primary key.

Referential Integrity

- A constraint involving **two** relations
 - The previous constraints involve a single relation.
- Used to specify a **relationship** among tuples in two relations:
 - The **referencing relation** and the **referenced relation**.

Referential Integrity

- Tuples in the **referencing relation** R1 have attributes FK (called **foreign key** attributes) that reference the primary key attributes PK of the **referenced relation** R2.
 - A tuple t1 in R1 is said to **reference** a tuple t2 in R2 if $t1[FK] = t2[PK]$.
- A referential integrity constraint can be displayed in a relational database schema as a directed arc from R1.FK to R2.

Referential Integrity (or foreign key) Constraint

- Statement of the constraint
 - The value in the foreign key column (or columns) FK of the the **referencing relation** R1 can be **either**:
 - (1) a value of an existing primary key value of a corresponding primary key PK in the **referenced relation** R2, or
 - (2) a **null**.
- In case (2), the FK in R1 should **not** be a part of its own primary key.

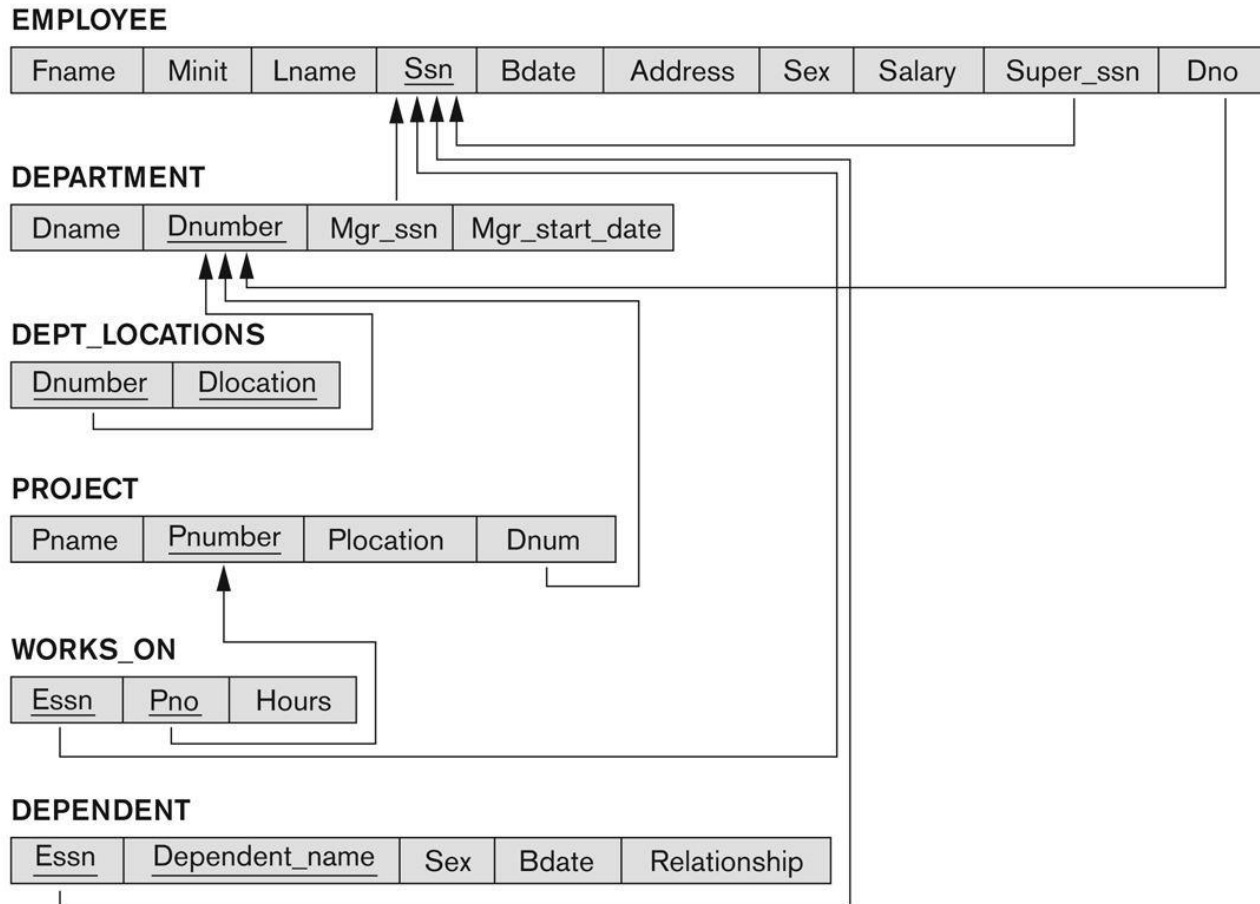
Displaying a relational database schema and its constraints

- Each relation schema can be displayed as a row of attribute names
- The name of the relation is written above the attribute names
- The primary key attribute (or attributes) will be underlined
- A foreign key (referential integrity) constraints is displayed as a directed arc (arrow) from the foreign key attributes to the referenced table
 - Can also point the the primary key of the referenced relation for clarity
- Next slide shows the COMPANY relational schema diagram with referential integrity constraints

Referential Integrity Constraints for COMPANY database

Figure 5.7

Referential integrity constraints displayed on the COMPANY relational database schema.



Other Types of Constraints

- Semantic Integrity Constraints:
 - based on application semantics and cannot be expressed by the model per se
 - Example: “the max. no. of hours per employee for all projects he or she works on is 56 hrs per week”
- A **constraint specification** language may have to be used to express these
- SQL-99 allows **CREATE TRIGGER** and **CREATE ASSERTION** to express some of these semantic constraints
- Keys, Permissibility of Null values, Candidate Keys (Unique in SQL), Foreign Keys, Referential Integrity etc. are expressed by the **CREATE TABLE** statement in SQL.

Update Operations on Relations

- INSERT a tuple.
- DELETE a tuple.
- MODIFY a tuple.
- Integrity constraints should not be violated by the update operations.
- Several update operations may have to be grouped together.
- Updates may **propagate** to cause other updates automatically. This may be necessary to maintain integrity constraints.

Update Operations on Relations

- In case of integrity violation, several actions can be taken:
 - Cancel the operation that causes the violation (RESTRICT or REJECT option)
 - Perform the operation but inform the user of the violation
 - Trigger additional updates so the violation is corrected (CASCADE option, SET NULL option)
 - Execute a user-specified error-correction routine

Possible violations for each operation

- INSERT may violate any of the constraints:
 - Domain constraint:
 - if one of the attribute values provided for the new tuple is not of the specified attribute domain
 - Key constraint:
 - if the value of a key attribute in the new tuple already exists in another tuple in the relation
 - Referential integrity:
 - if a foreign key value in the new tuple references a primary key value that does not exist in the referenced relation
 - Entity integrity:
 - if the primary key value is null in the new tuple

Possible violations for each operation

- DELETE may violate only referential integrity:
 - If the primary key value of the tuple being deleted is referenced from other tuples in the database
 - Can be remedied by several actions: RESTRICT, CASCADE, SET NULL (see Chapter 6 for more details)
 - RESTRICT option: reject the deletion
 - CASCADE option: propagate the new primary key value into the foreign keys of the referencing tuples
 - SET NULL option: set the foreign keys of the referencing tuples to NULL
 - One of the above options must be specified during database design for each foreign key constraint

Possible violations for each operation

- UPDATE may violate domain constraint and NOT NULL constraint on an attribute being modified
- Any of the other constraints may also be violated, depending on the attribute being updated:
 - Updating the primary key (PK):
 - Similar to a DELETE followed by an INSERT
 - Need to specify similar options to DELETE
 - Updating a foreign key (FK):
 - May violate referential integrity
 - Updating an ordinary attribute (neither PK nor FK):
 - Can only violate domain constraints

Summary

- Presented Relational Model Concepts
 - Definitions
 - Characteristics of relations
- Discussed Relational Model Constraints and Relational Database Schemas
 - Domain constraints
 - Key constraints
 - Entity integrity
 - Referential integrity
- Described the Relational Update Operations and Dealing with Constraint Violations

CHAPTER 9

Relational Database Design by ER-to-Relational Mapping

Chapter Outline

- **ER-to-Relational Mapping Algorithm**
 - Step 1: Mapping of Regular Entity Types
 - Step 2: Mapping of Weak Entity Types
 - Step 3: Mapping of Binary 1:1 Relation Types
 - Step 4: Mapping of Binary 1:N Relationship Types.
 - Step 5: Mapping of Binary M:N Relationship Types.
 - Step 6: Mapping of Multivalued attributes.
 - Step 7: Mapping of N-ary Relationship Types.

GOALS during Mapping

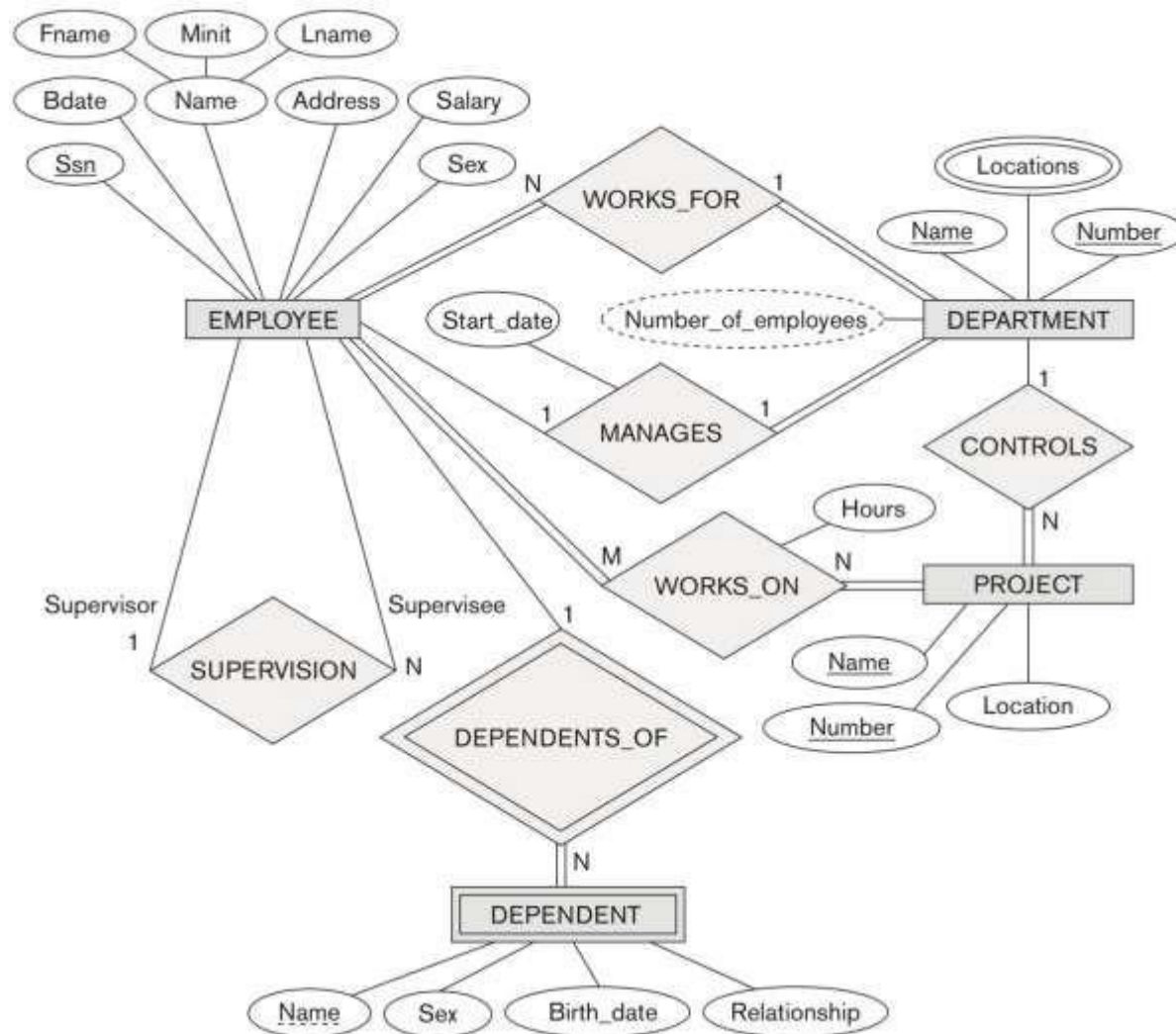
- Preserve all information (that includes all attributes)
- Maintain the constraints to the extent possible (Relational Model cannot preserve all constraints- e.g., max cardinality ratio such as 1:10 in ER; exhaustive classification into subtypes, e.g., STUDENTS are specialized into Domestic and Foreign)
- Minimize null values

The mapping procedure described has been implemented in many commercial tools.

ER-to-Relational Mapping Algorithm

- Step 1: Mapping of Regular Entity Types.
 - For each regular (strong) entity type E in the ER schema, create a relation R that includes all the simple attributes of E.
 - Choose one of the key attributes of E as the primary key for R.
 - If the chosen key of E is composite, the set of simple attributes that form it will together form the primary key of R.
- Example: We create the relations EMPLOYEE, DEPARTMENT, and PROJECT in the relational schema corresponding to the regular entities in the ER diagram.
 - SSN, DNUMBER, and PNUMBER are the primary keys for the relations EMPLOYEE, DEPARTMENT, and PROJECT as shown.

Figure 9.1 The ER conceptual schema diagram for the COMPANY database.



ER-to-Relational Mapping Algorithm (contd.)

■ Step 2: Mapping of Weak Entity Types

- For each weak entity type *W* in the ER schema with owner entity type *E*, create a relation *R* & include all simple attributes (or simple components of composite attributes) of *W* as attributes of *R*.
- Also, include as foreign key attributes of *R* the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s).
- The primary key of *R* is the *combination* of the primary key(s) of the owner(s) and the partial key of the weak entity type *W*, if any.

■ Example: Create the relation **DEPENDENT** in this step to correspond to the weak entity type **DEPENDENT**.

- Include the primary key SSN of the EMPLOYEE relation as a foreign key attribute of DEPENDENT (renamed to ESSN).
 - The primary key of the DEPENDENT relation is the combination {ESSN, DEPENDENT_NAME} because DEPENDENT_NAME is the partial key of DEPENDENT.

ER-to-Relational Mapping Algorithm (contd.)

- **Step 3: Mapping of Binary 1:1 Relation Types**
 - For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R.
- There are three possible approaches:
 1. **Foreign Key (2 relations) approach:** Choose one of the relations-say S-and include a foreign key in S the primary key of T. It is better to choose an entity type with total participation in R in the role of S.
 - Example: 1:1 relation MANAGES is mapped by choosing the participating entity type DEPARTMENT to serve in the role of S, because its participation in the MANAGES relationship type is total.
 2. **Merged relation (1 relation) option:** An alternate mapping of a 1:1 relationship type is possible by merging the two entity types and the relationship into a single relation. This may be appropriate when both participations are total.
 3. **Cross-reference or relationship relation (3 relations) option:** The third alternative is to set up a third relation R for the purpose of cross-referencing the primary keys of the two relations S and T representing the entity types.

ER-to-Relational Mapping Algorithm (contd.)

- Step 4: Mapping of Binary 1:N Relationship Types.
 - For each regular binary 1:N relationship type R, identify the relation S that represent the participating entity type at the N-side of the relationship type.
 - Include as foreign key in S the primary key of the relation T that represents the other entity type participating in R.
 - Include any simple attributes of the 1:N relation type as attributes of S.
- Example: 1:N relationship types WORKS_FOR, CONTROLS, and SUPERVISION in the figure.
 - For WORKS_FOR we include the primary key DNUMBER of the DEPARTMENT relation as foreign key in the EMPLOYEE relation and call it DNO.
- An alternative approach is to use a Relationship relation (cross referencing relation) – this is rarely done.

ER-to-Relational Mapping Algorithm (contd.)

- **Step 5: Mapping of Binary M:N Relationship Types.**
 - For each regular binary M:N relationship type R, *create a new relation S to represent R. This is a *relationship relation*.*
 - Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types; *their combination will form the primary key of S.*
 - Also include any simple attributes of the M:N relationship type (or simple components of composite attributes) as attributes of S.
- **Example: The M:N relationship type WORKS_ON from the ER diagram is mapped by creating a relation WORKS_ON in the relational database schema.**
 - The primary keys of the PROJECT and EMPLOYEE relations are included as foreign keys in WORKS_ON and renamed PNO and ESSN, respectively.
 - Attribute HOURS in WORKS_ON represents the HOURS attribute of the relation type. The primary key of the WORKS_ON relation is the combination of the foreign key attributes {ESSN, PNO}.

ER-to-Relational Mapping Algorithm (contd.)

■ Step 6: Mapping of Multivalued attributes.

- For each multivalued attribute A, create a new relation R.
- This relation R will include an attribute corresponding to A, plus the primary key attribute K-as a foreign key in R-of the relation that represents the entity type of relationship type that has A as an attribute.
- The primary key of R is the combination of A and K. If the multivalued attribute is composite, we include its simple components.

■ Example: The relation DEPT_LOCATIONS is created.

- The attribute DLOCATION represents the multivalued attribute LOCATIONS of DEPARTMENT, while DNUMBER-as foreign key-represents the primary key of the DEPARTMENT relation.
- The primary key of R is the combination of {DNUMBER, DLOCATION}.

ER-to-Relational Mapping Algorithm (contd.)

- **Step 7: Mapping of N-ary Relationship Types.**
 - For each n-ary relationship type R, where $n > 2$, create a new relationship S to represent R.
 - Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types.
 - Also include any simple attributes of the n-ary relationship type (or simple components of composite attributes) as attributes of S.
- **Example:** The relationship type SUPPY in the ER on the next slide.
 - This can be mapped to the relation SUPPLY shown in the relational schema, whose primary key is the combination of the three foreign keys {SNAME, PARTNO, PROJNAME}

Figure 9.2 Result of mapping the COMPANY ER schema into a relational database schema.

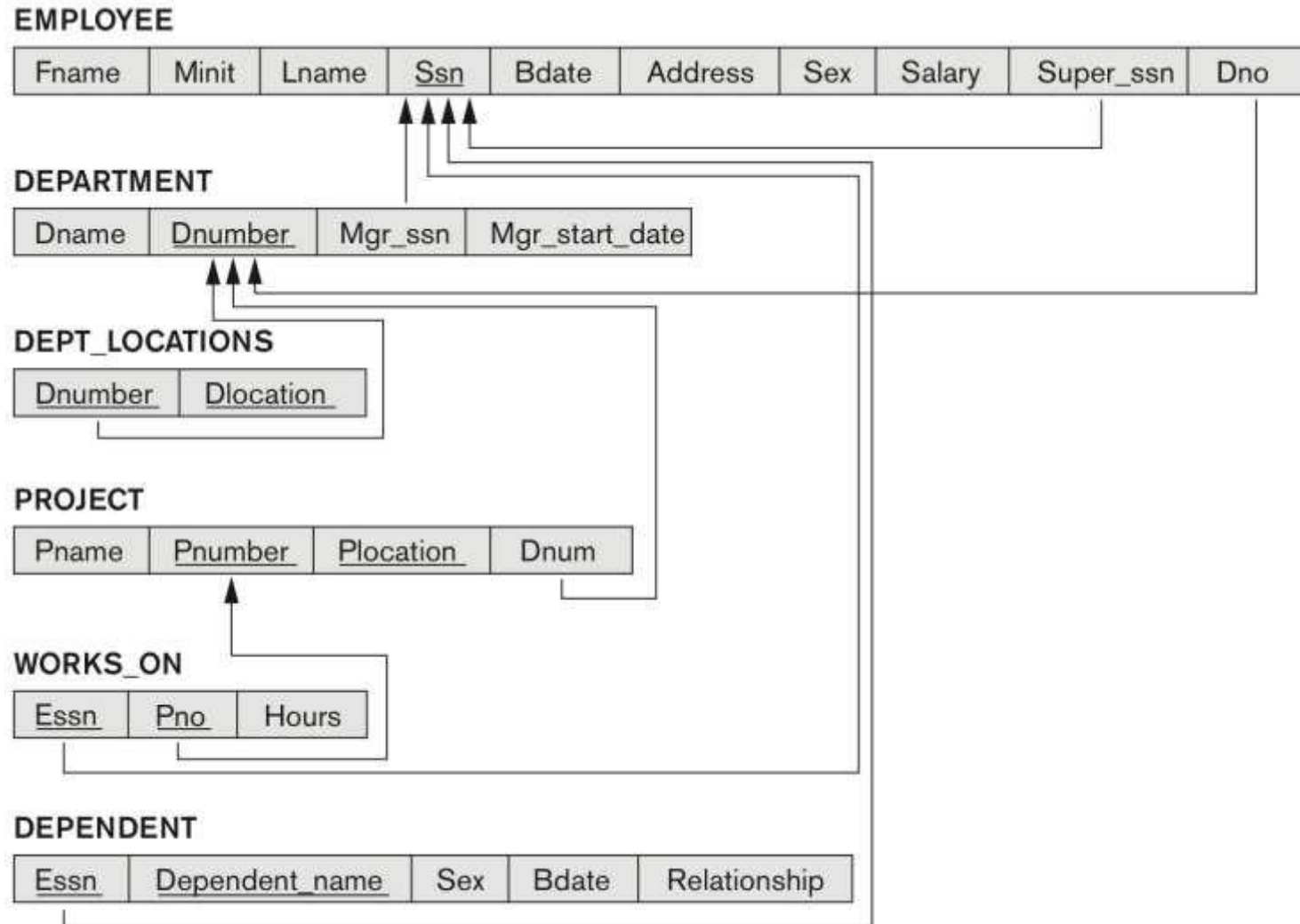
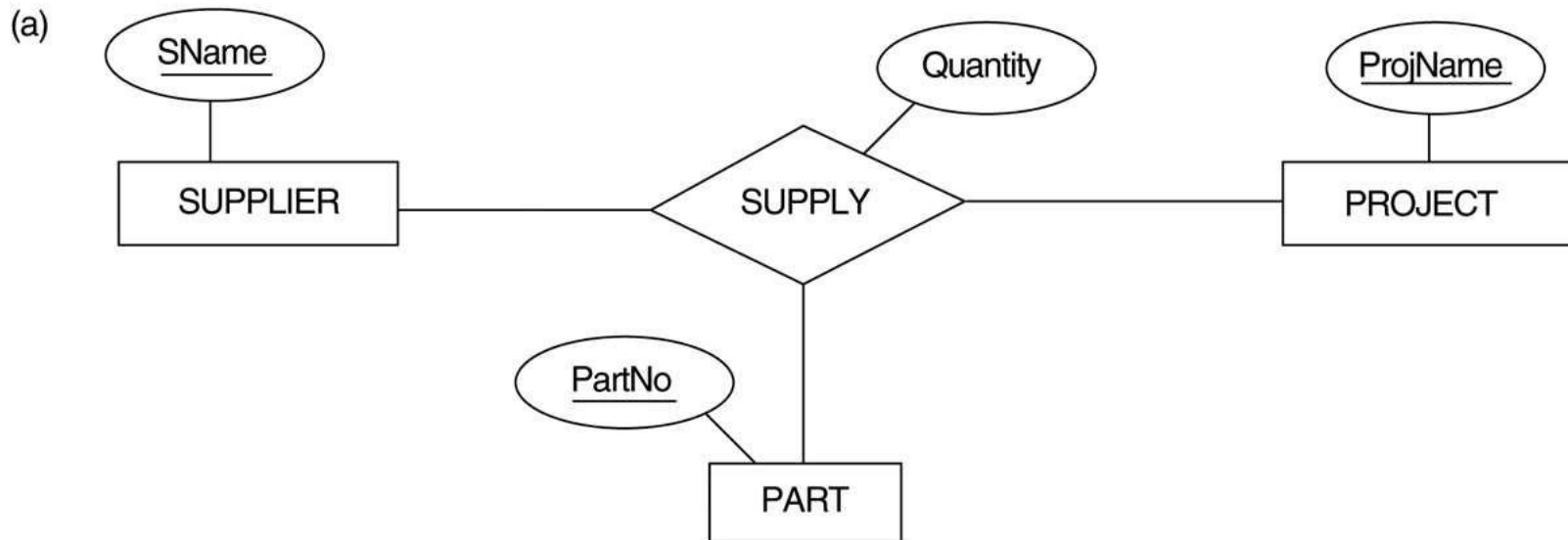
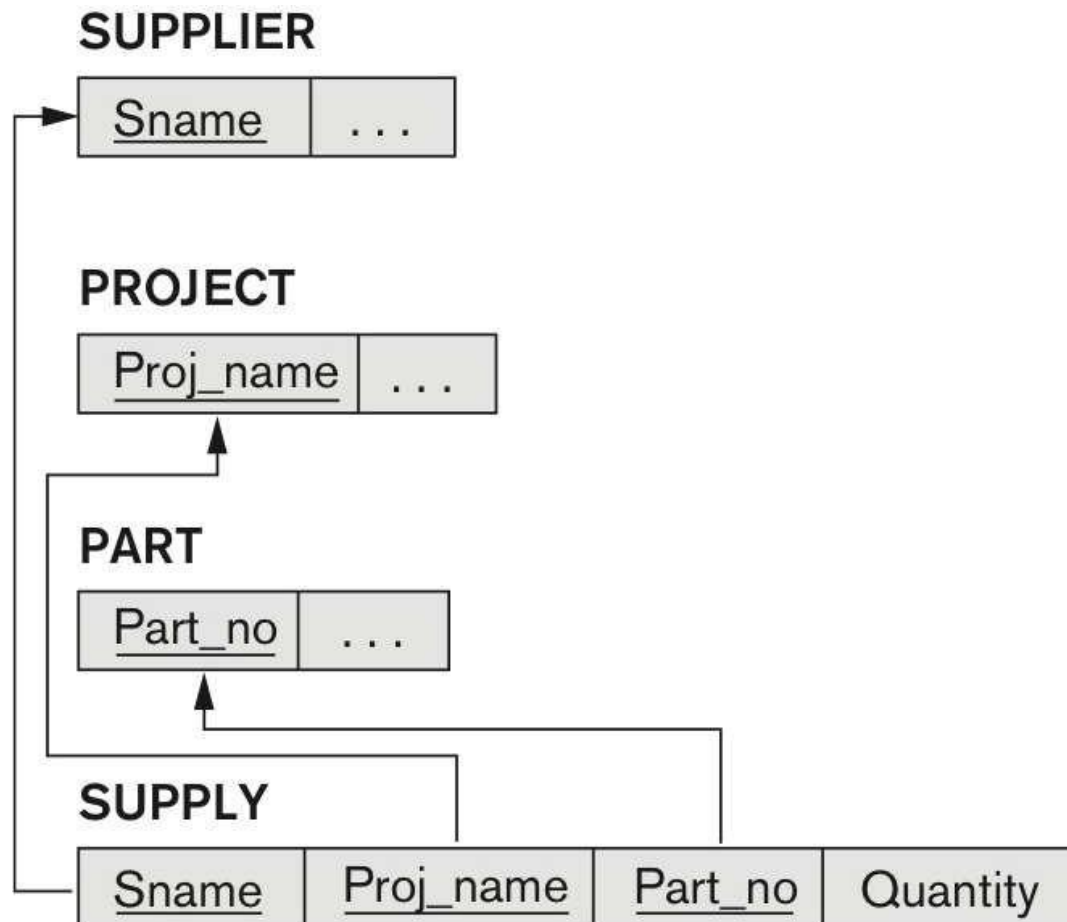


FIGURE 3.17

TERNARY RELATIONSHIP: SUPPLY



Mapping the *n*-ary relationship type SUPPLY



Summary of Mapping constructs and constraints

Table 9.1 Correspondence between ER and Relational Models

ER MODEL	RELATIONAL MODEL
Entity type	<i>Entity</i> relation
1:1 or 1:N relationship type	Foreign key (or <i>relationship</i> relation)
M:N relationship type	<i>Relationship</i> relation and <i>two</i> foreign keys
<i>n</i> -ary relationship type	<i>Relationship</i> relation and <i>n</i> foreign keys
Simple attribute	Attribute
Composite attribute	Set of simple component attributes
Multivalued attribute	Relation and foreign key
Value set	Domain
Key attribute	Primary (or secondary) key