

Chapter 1

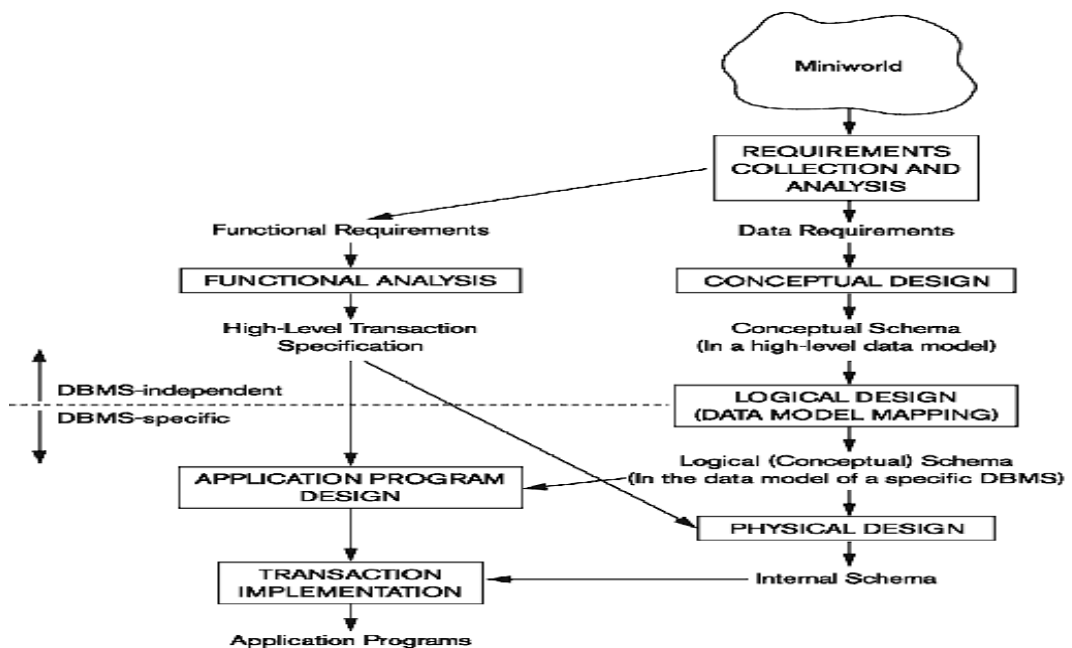
Data Modeling using Entity Relationship (E-R) Diagram

Contents

- Conceptual Data Model for Database Design
- 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

Overview of Database

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



Entity-Relationship (ER) Model Concepts

- A popular high-level conceptual data model
- Entities and Attributes
 - Entities are specific objects or things in the mini-world that are represented in the database.
 - For example the EMPLOYEE John Smith, the Research DEPARTMENT, the ProductX PROJECT
 - Attributes are properties used to describe an entity.
 - For example an EMPLOYEE entity may have the attributes Name, SSN, Address, Gender, 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', Gender='M', BirthDate='09-JAN-55'
 - Each attribute has a *value set* (or datatype) associated with it – e.g. integer, string, subrange, enumerated type, etc

Types of Attributes

- Simple
 - Each entity has a single atomic value for the attribute. For example, SSN or Gender.
- 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}.

Example of a Composite Attribute

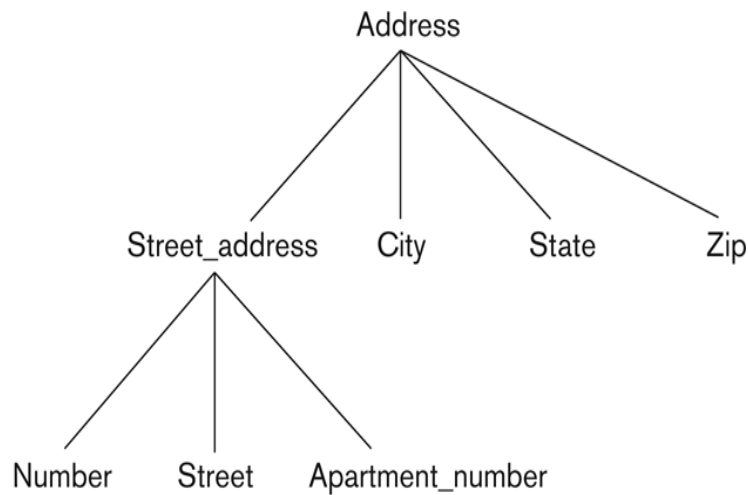



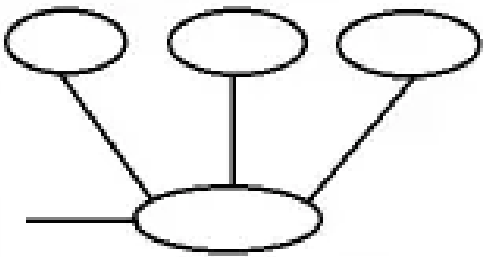



Figure 3.4

A hierarchy of composite attributes.

- 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
- Complex Attributes
 - Nested composite and multivalued attributes
 - Ex. A person has more than one residence and each residence can have a single address and multiple phones

Notations Of Attributes in ER Diagram

	Attribute
	Key Attribute
	Multivalued Attribute
	Compound/Composite Attribute
	Derived Attribute

Stored and Derived Attributes

- An derived attribute is derived from a stored attribute
 - Ex. We can derive a man's age from his birthday.

Null Values -

Its meaning includes

- An attribute value is not applicable
- An attribute value is unknown
 - The value exists but is missing
 - The value is unknown whether it exists

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.
- 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

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
 - Derived attributes are denoted by dotted ovals
 - Each key attribute is underlined
 - Multivalued attributes displayed in double ovals
- See CAR example below
- Entity Type CAR with two keys and a corresponding Entity Set

(a)

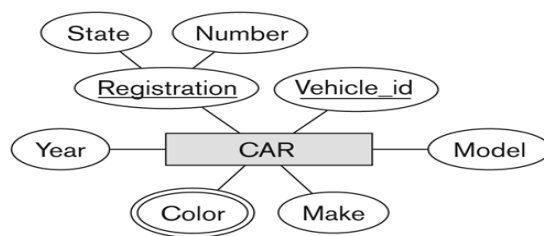
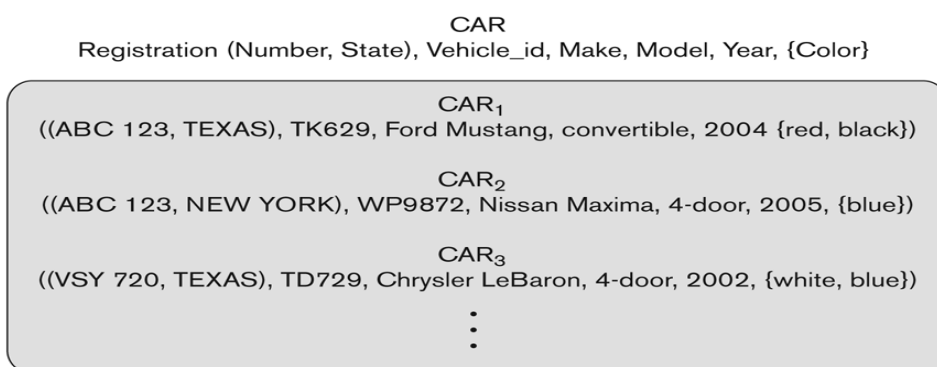


Figure 3.7

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

(b)



Entity Set Value Sets (Domains) of Attributes

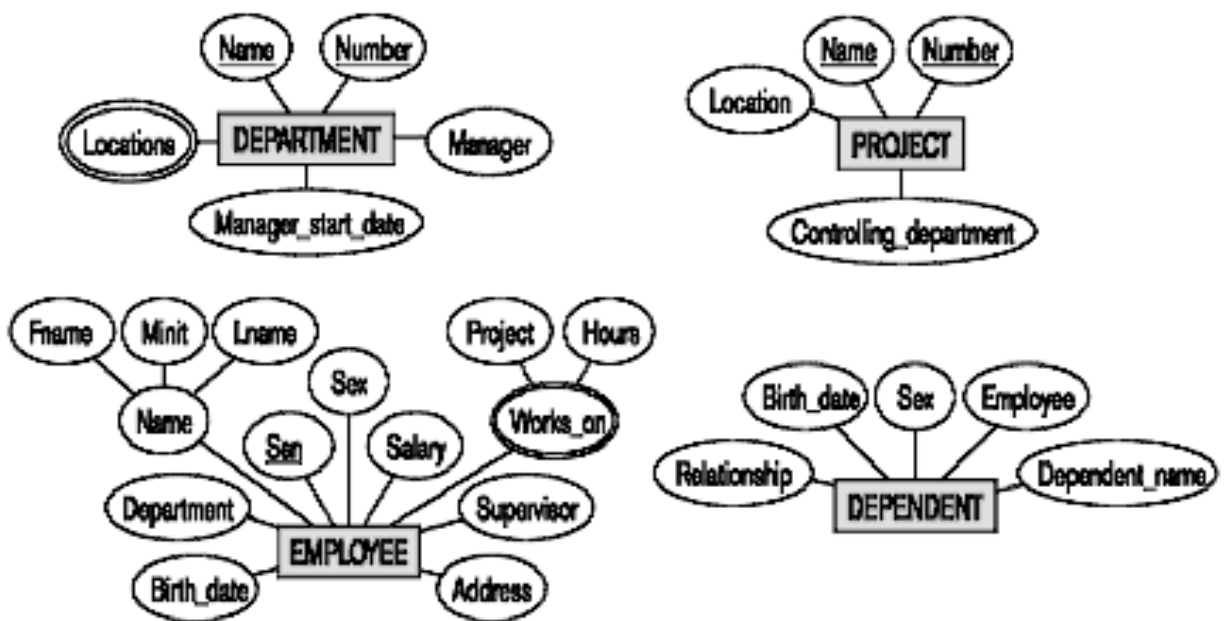
- Each entity type will have a collection of entities stored in the database
 - Called the **entity set** (also called the **extension** of the entity type)
 - An entity type describes the **schema** or **intension** for a set of entities
- Previous slide shows three CAR entity instances in the entity set for CAR
- Same name (CAR) used to refer to both the entity type and the entity set
- Entity set is the current *state* of the entities of that type that are stored in the database
- Each simple attribute is associated with a **value set** (or **domain** of values)
 - Ex. The **Age** attribute of **EMPLOYEE** to be the set of integer numbers between 16 to 70

Initial 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 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



Refining the Initial Design by Introducing Relationships

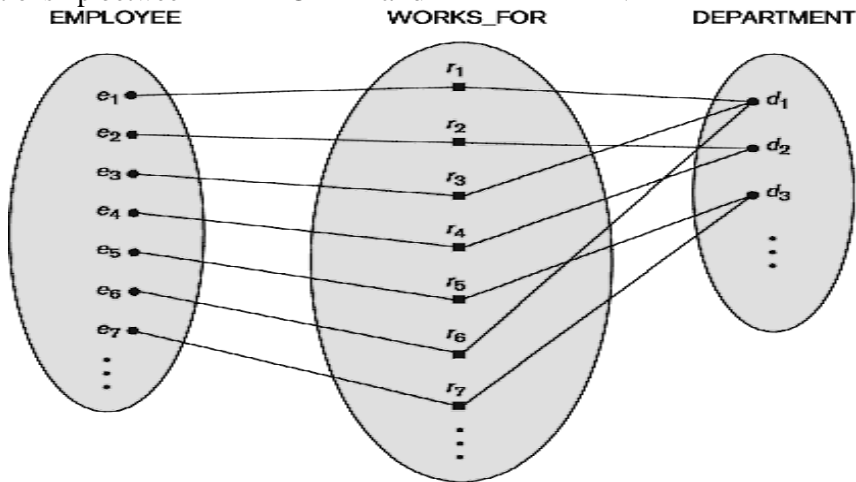
- The initial design is typically not complete
- Some aspects in the requirements will be represented as **relationships**
- ER model has three main concepts:
 - Entities (and their entity types and entity sets)
 - Attributes (simple, composite, multivalued)
 - Relationships (and their relationship types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

Relationships and Relationship Types

- 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.

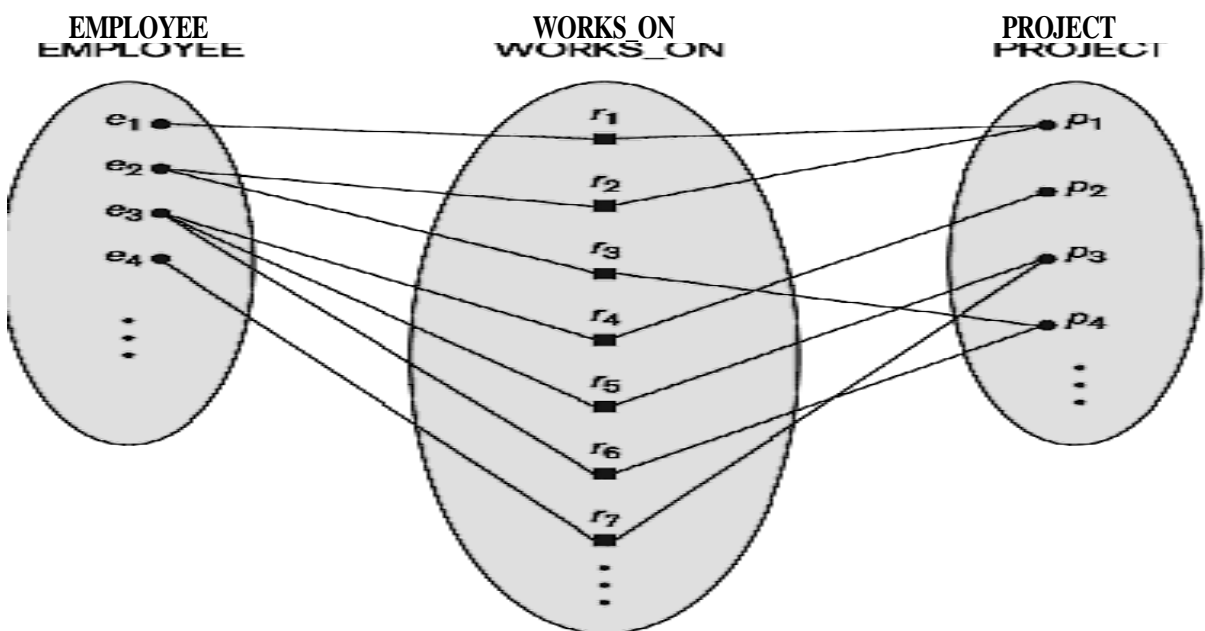
Relationship Instances of the WORKS_FOR

Relationship between EMPLOYEE and DEPARTMENT



Relationship Instances of the M:N WORKS_ON

Relationship between EMPLOYEE and PROJECT



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
- 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

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)

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.
- Each entity type that participates in a relationship type plays a particular **role** in the relationship

Recursive Relationship Type

- A relationship type where the same entity type participates more than once in the relationship in **distinct roles is called recursive relationship**
- 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

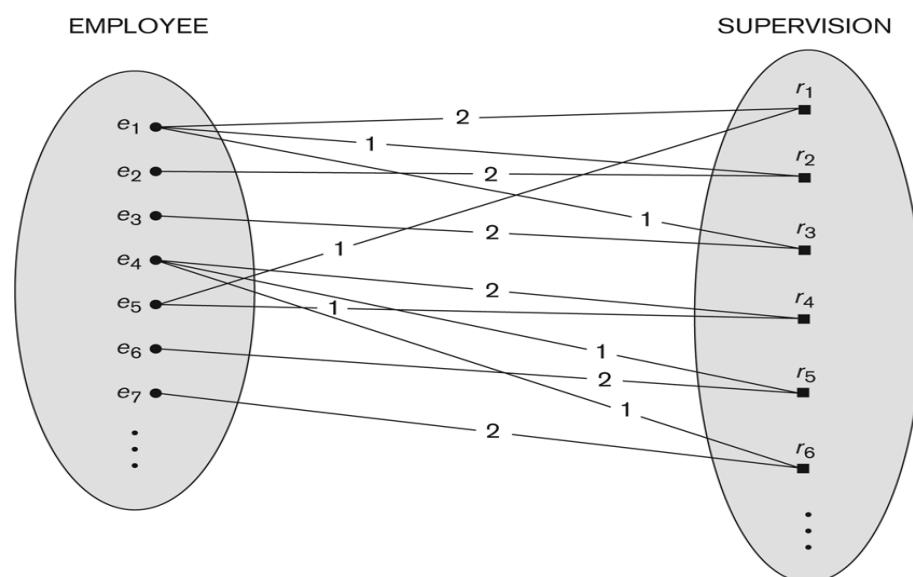
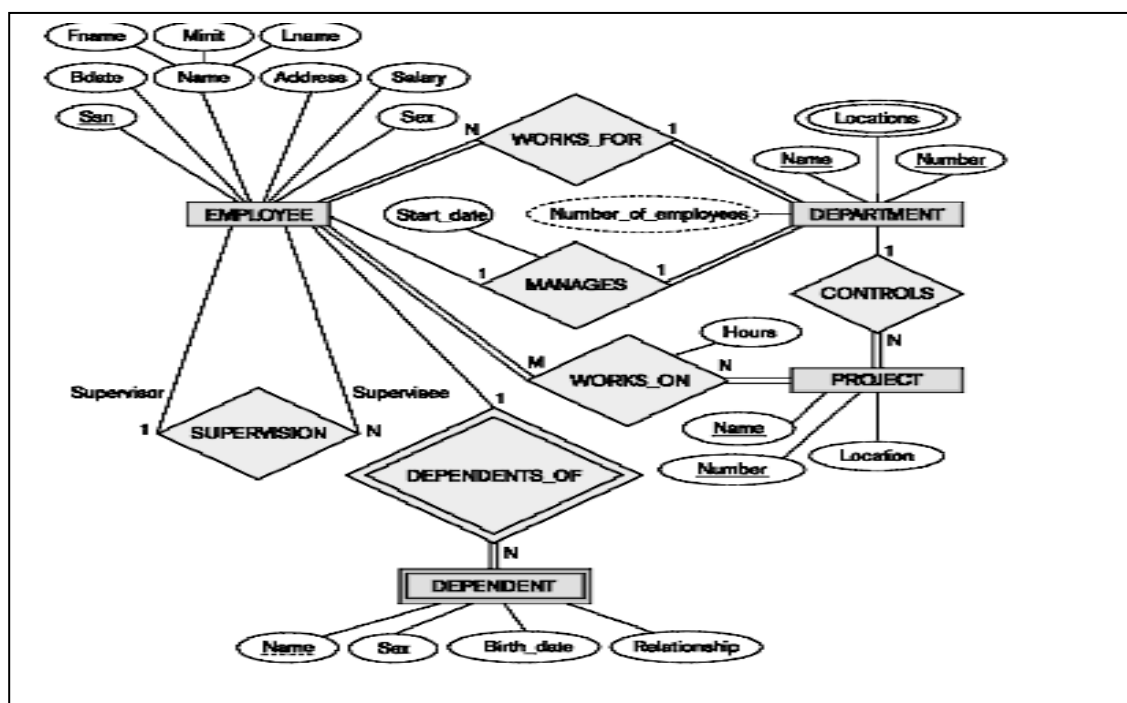


Figure 3.11
A recursive relationship SUPERVISION between EMPLOYEE in the *supervisor* role (1) and EMPLOYEE in the *subordinate* role (2).

Recursive Relationship type is : SUPERVISION (participation rolenames are shown)



Weak Entity Types

- An entity that does not have a key attribute
- A weak entity must participate in an identifying relationship type with an owner or identifying entity type
- Entities are identified by the combination of:
 - A partial key of the weak entity type
 - The particular entity they are related to in the identifying entity 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
- A weak entity type and its identifying relationship are distinguished by surrounding their boxes and diamonds with double lines
- The partial key attribute is underlined with a dashed or dotted line

Constraints on Relationships

- Constraints on Relationship Types
 - 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)

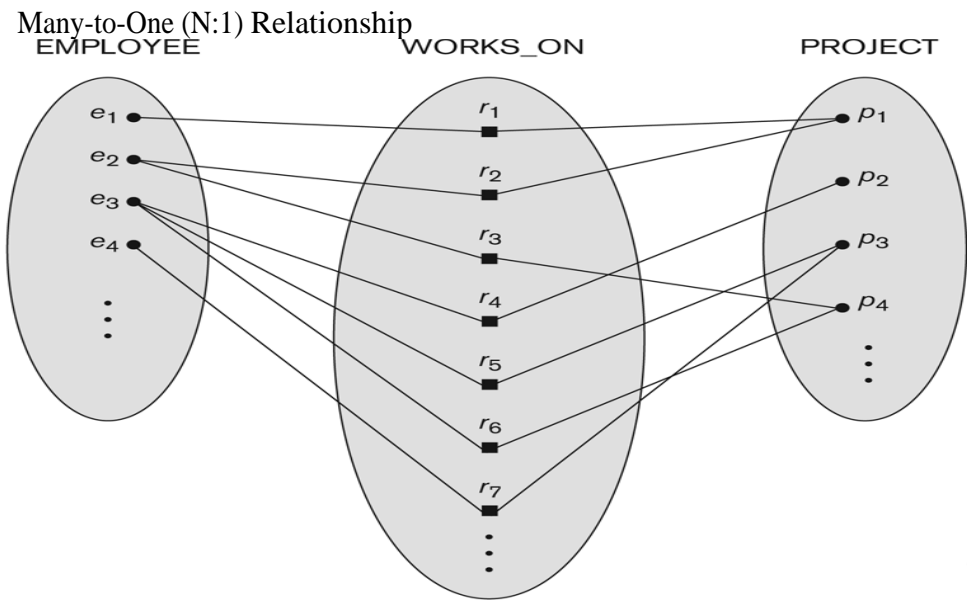


Figure 3.13
An M:N relationship,
WORKS_ON.

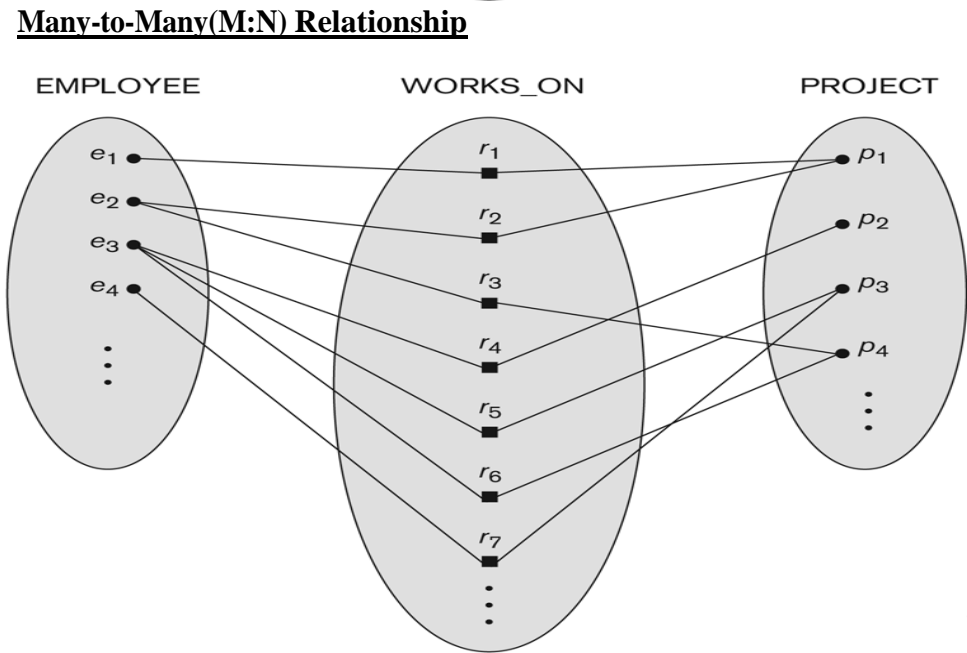


Figure 3.13
An M:N relationship,
WORKS_ON.

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
 - For M:N relationships, some attributes are determined by the combination of participating entities, not by a single entity. Such attributes must be specified as relationship attributes
 - In 1:1 relationships, they can be transferred to one of the participating entities
 - In 1:N relationships, they can be transferred to the entity type on the N-side of the relationship
 - The decision as to where a relationship attribute should be placed is determined subjectively by the schema designers

Example Attribute of a Relationship Type: Hours of WORKS_ON

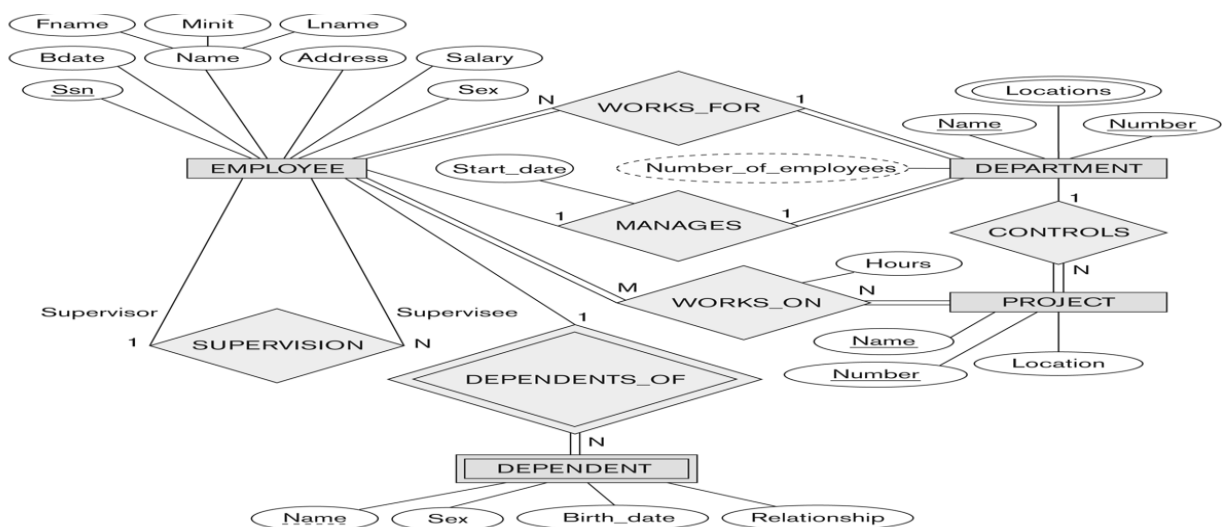


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

Notation for Constraints on Relationships

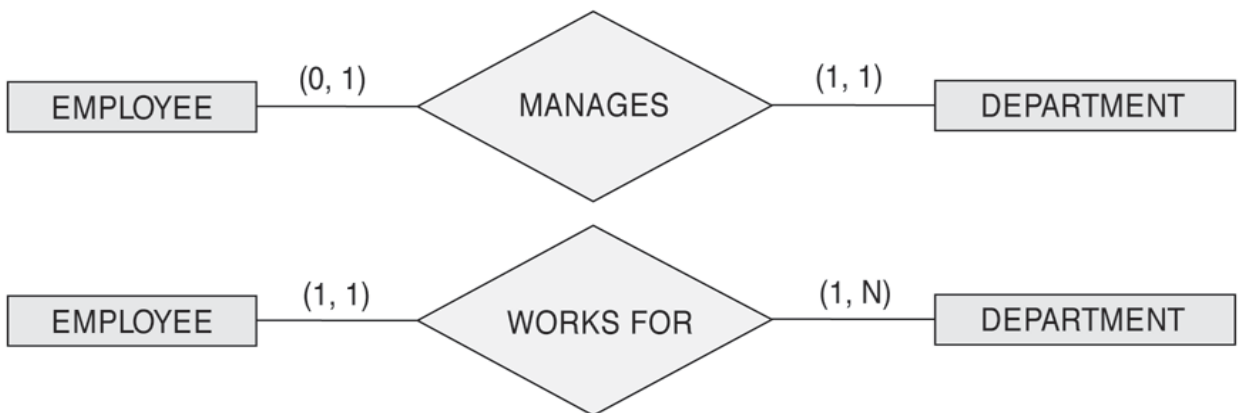
- Cardinality ratio (of a binary relationship): 1:1, 1:N, N:1, or M:N
 - Shown by placing appropriate numbers on the relationship edges.
- Participation constraint (on each participating entity type): **total** (called existence dependency) or **partial**.
 - Total shown by double line, partial by single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

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)

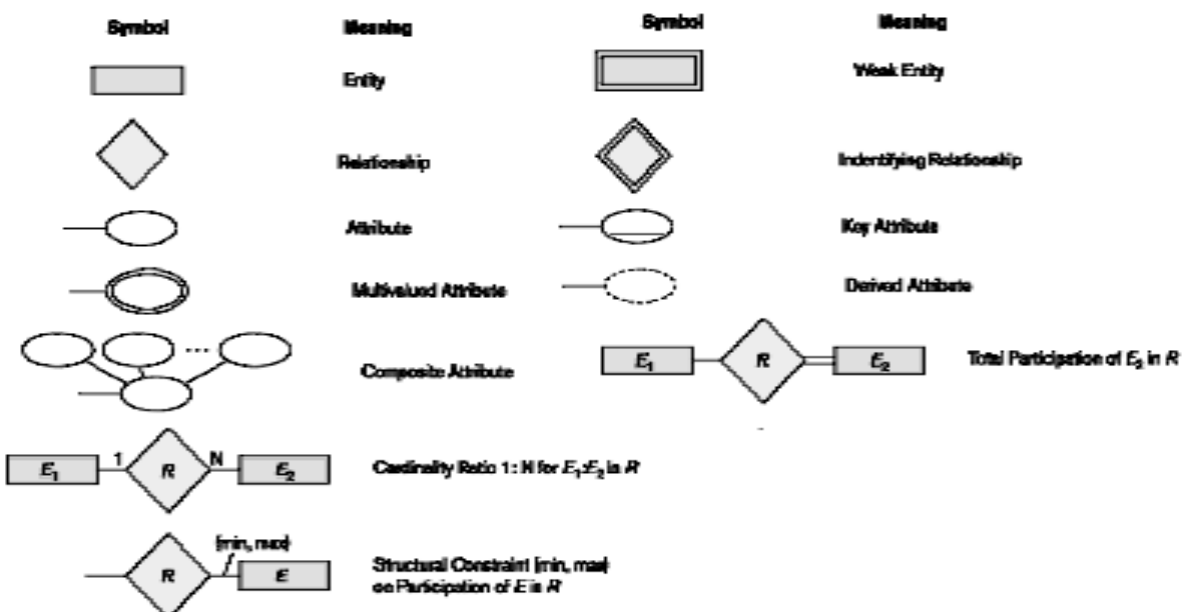
- Must have $\min \leq \max$, $\min \geq 0$, $\max \geq 1$
 - $\min=0$ implies partial participation; $\min>0$ implies total participation
- Derived from the knowledge of mini-world constraints
- Examples:
 - A department has exactly one manager and an employee can manage at most one department.
 - Specify (1,1) for participation of DEPARTMENT in MANAGES
 - Specify (0,1) for participation of EMPLOYEE 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 (1,n) for participation of DEPARTMENT in WORKS_FOR

The(min,max)Notationfor RelationshipConstraints



- Read the min,max numbers next to the entity type and looking away from the entity type

Summaryof Notationfor ER Diagrams



COMPANY ER Schema Diagram Using (min,max) Notation

