4CCS1DBS – Database Systems

Data Modelling Using the Entity-Relationship Model

Recap: Data Models

- **Data Model:**
  - A set of concepts to describe the ***structure*** of a database, the ***operations*** for manipulating these structures, and certain ***constraints*** that the database should obey.
- **Data Model Structure and Constraints:**
  - Constructs are used to define the database structure
  - Constructs typically include ***elements*** (and their ***data types***) as well as groups of elements (e.g. ***entity, record, table***), and ***relationships*** among such groups
  - Constraints specify some restrictions on valid data; these constraints must be enforced at all times

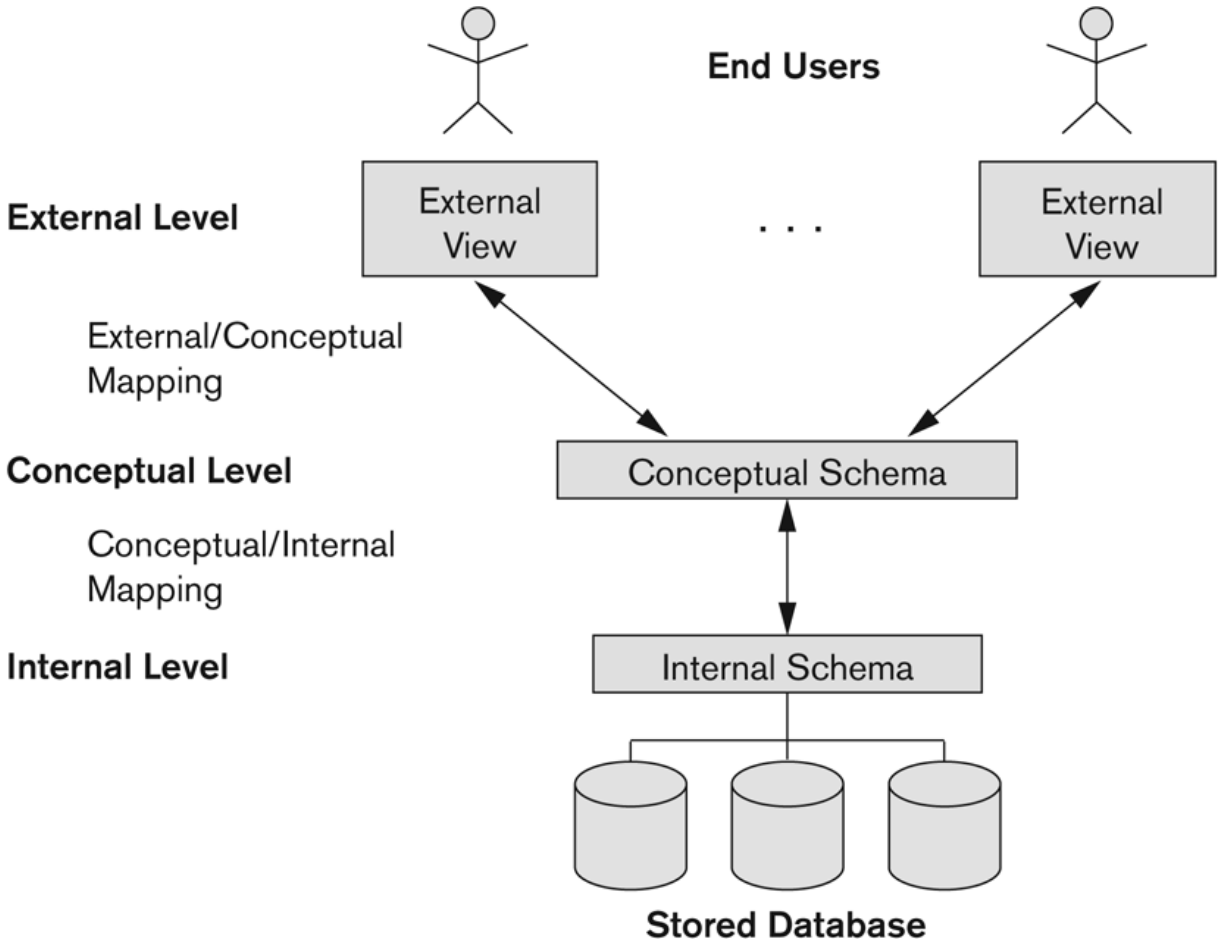
Recap: Data Models (continued)

- **Data Model Operations:**
  - These operations are used for specifying database *retrievals* and *updates* by referring to the constructs of the data model.
  - Operations on the data model may include
    - *basic model operations* (e.g. generic insert, delete, update)
    - *user-defined operations* (e.g. compute_student_gpa, update_inventory)

Recap: Categories of Data Models

- **Conceptual (high-level, semantic) data models:**
  - Provide concepts that are close to the way many users perceive data.
    - (Also called ***entity-based*** or ***object-based*** data models.)
- **Physical (low-level, internal) data models:**
  - Provide concepts that describe details of how data is stored in the computer. These are usually specified in an ad-hoc manner through DBMS design and administration manuals
- **Implementation (representational) data models:**
  - Provide concepts that fall between the above two, used by many commercial DBMS implementations (e.g. relational data models used in many commercial systems).

Recap: The Three-Schema Architecture



Recap: Three-Schema Architecture

- Defines DBMS schemas at ***three*** levels:
  - **Internal schema** at the internal level to describe physical storage structures and access paths (eg. indexes).
    - Typically uses a **physical** data model.
  - **Conceptual schema** at the conceptual level to describe the structure and constraints for the whole database for a community of users.
    - Uses a **conceptual** or an **implementation** data model.
  - **External schemas** at the external level to describe the various user views.
    - Usually uses the same data model as the conceptual schema.

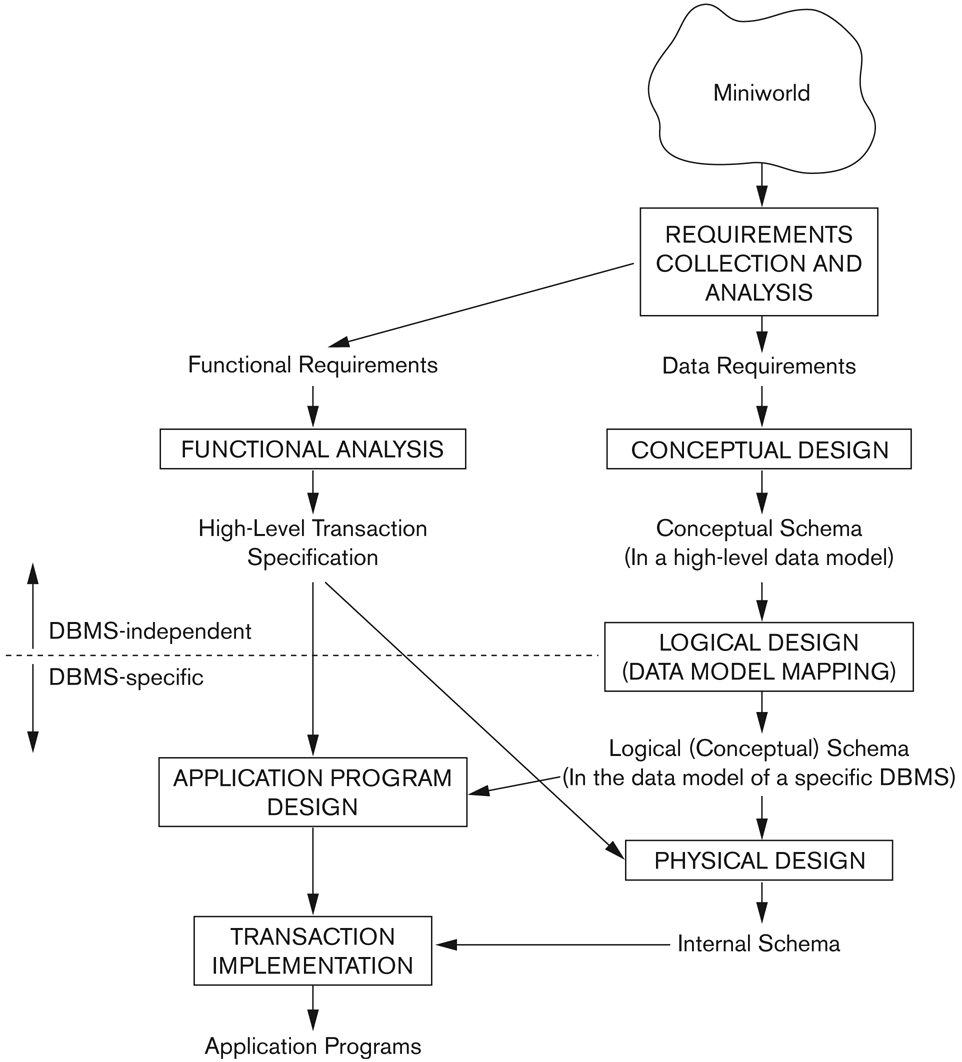
Outline For Today

- 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

Overview of Database Design Process

- Two main activities:
  - Database design
  - Applications design
- Focus on database design here
  - 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



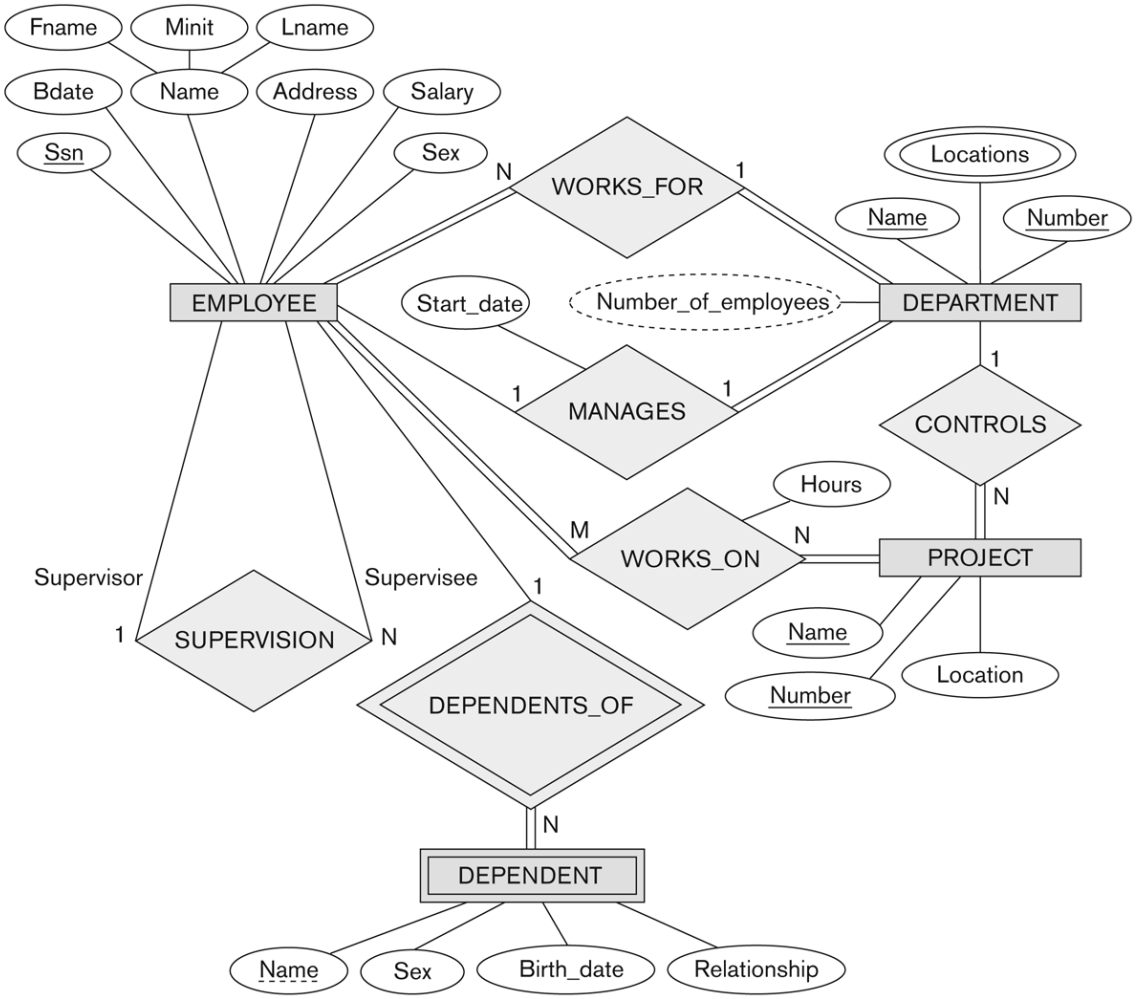
Example COMPANY Database

- We need to create a database schema design based on the following (simplified) **requirements** of the COMPANY Database:
  - The company is organised 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 (Contd.)

- We store each EMPLOYEE's social security number, address, salary, gender and birthdate.
  - Each employee *works for* one department but may *work on* several projects.
  - We keep track of the number of hours per week that an employee currently works on each project.
  - We also keep track of the *direct supervisor* of each employee.
- Each employee may *have* a number of DEPENDENTs.
  - For each dependent, we keep track of their name, sex, birthdate, and relationship to the employee.

An Entity-Relationship Diagram



ER Model Concepts – 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

ER Model Concepts – Entities and Attributes

- Attributes are properties used to describe an entity.
  - For example an EMPLOYEE entity may have the attributes Name, SSN, Address, BirthDate etc.
  - 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, etc.

Types of Attributes (1)

- Simple
  - Each entity has a single atomic value for the attribute. For example: SSN.
- 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.

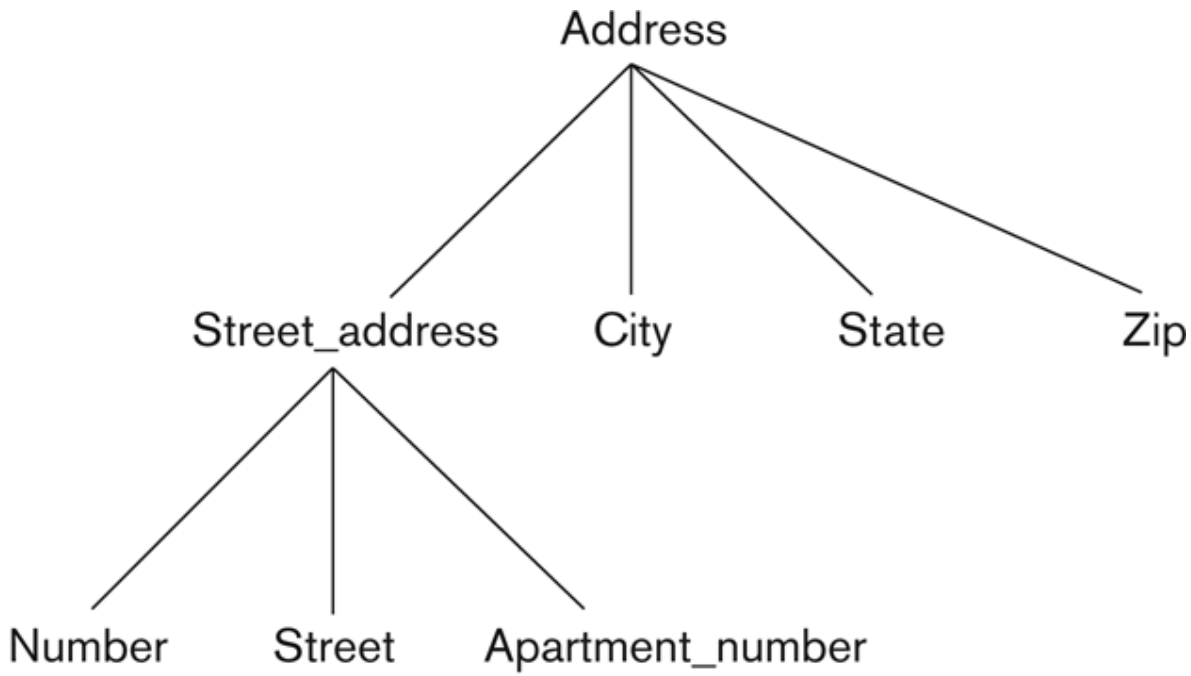
Types of Attributes (2)

- Single-valued
  - e.g. age
- Multi-valued
  - An entity may have multiple values for that attribute. For example, Colour of a CAR or PreviousDegrees of a STUDENT.
    - Denoted as {Colour} or {PreviousDegrees}.

Types of Attributes (3)

- 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



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)

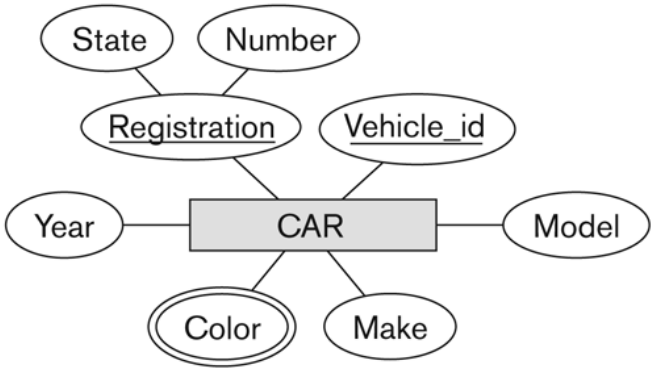
- A key attribute may be *composite*.
  - Example from US car ids: VehicleTagNumber is a key of the CAR entity type with components (Number, State).
- An entity type may have more than one key.
  - The CAR entity type may have two keys:
    - VehicleIdentificationNumber (popularly called VIN)
    - VehicleTagNumber (Number, State) or license plate number.
- Each key is underlined in the ER diagram

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 CAR example on next slide

Entity Type CAR with Two Keys and a Corresponding Entity Set

- Entity type



- Entity set

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})

.
.
.

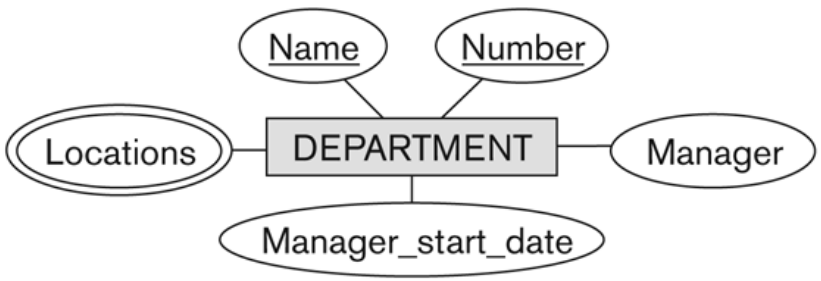
Entity Set

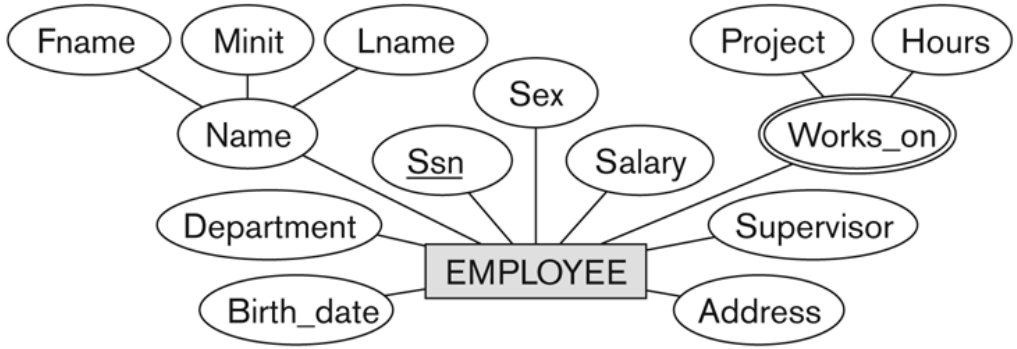
- Each entity type will have a collection of entities stored in the database
  - Called the *entity set*
- Previous slide shows three CAR entity instances in the entity set for CAR
- Same name (CAR) used to refer to both the entity type and the entity set
- Entity set is the current *state* of the entities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

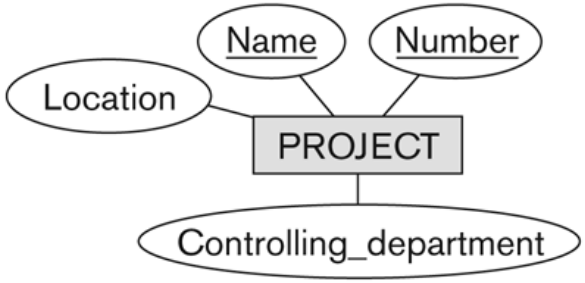
Initial Design of Entity Types for the COMPANY Database Schema

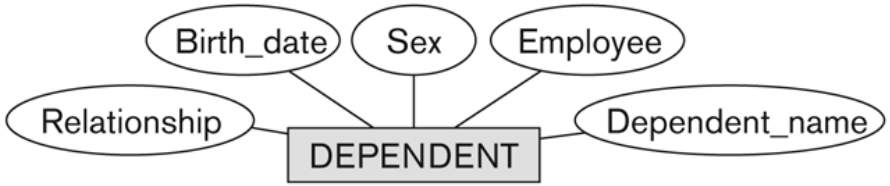
- Based on the requirements, we can identify four initial entity types in the COMPANY database:
  - DEPARTMENT
  - PROJECT
  - EMPLOYEE
  - DEPENDENT
- Their initial design is shown on the following slide
- The initial attributes shown are derived from the requirements description

Initial Design of Entity Types:
EMPLOYEE, DEPARTMENT, PROJECT, DEPENDENT









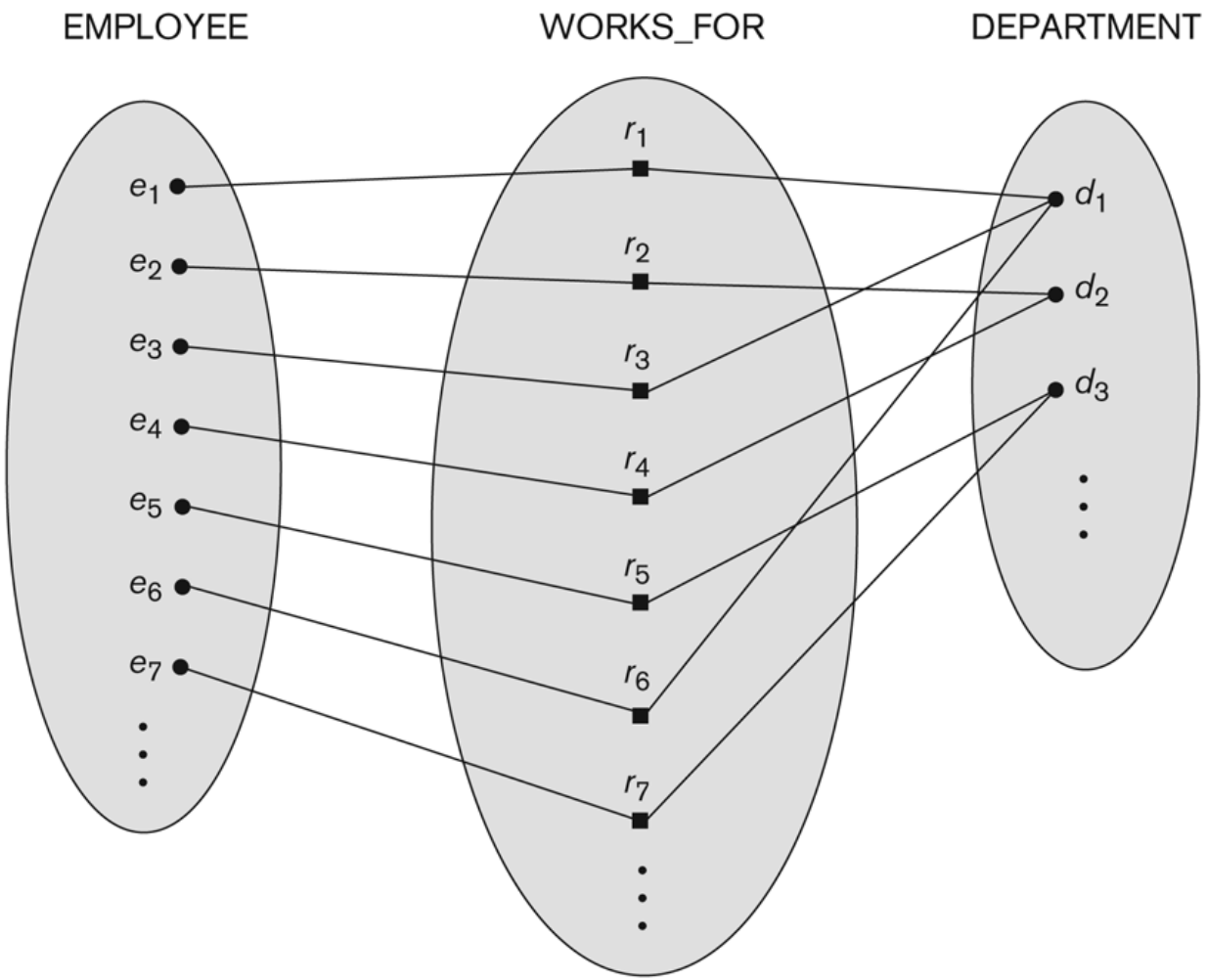
Refining the Initial Design by Introducing Relationships

- 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, multi-valued)
  - Relationships (and their relationship types and relationship sets)
- We introduce relationship concepts next

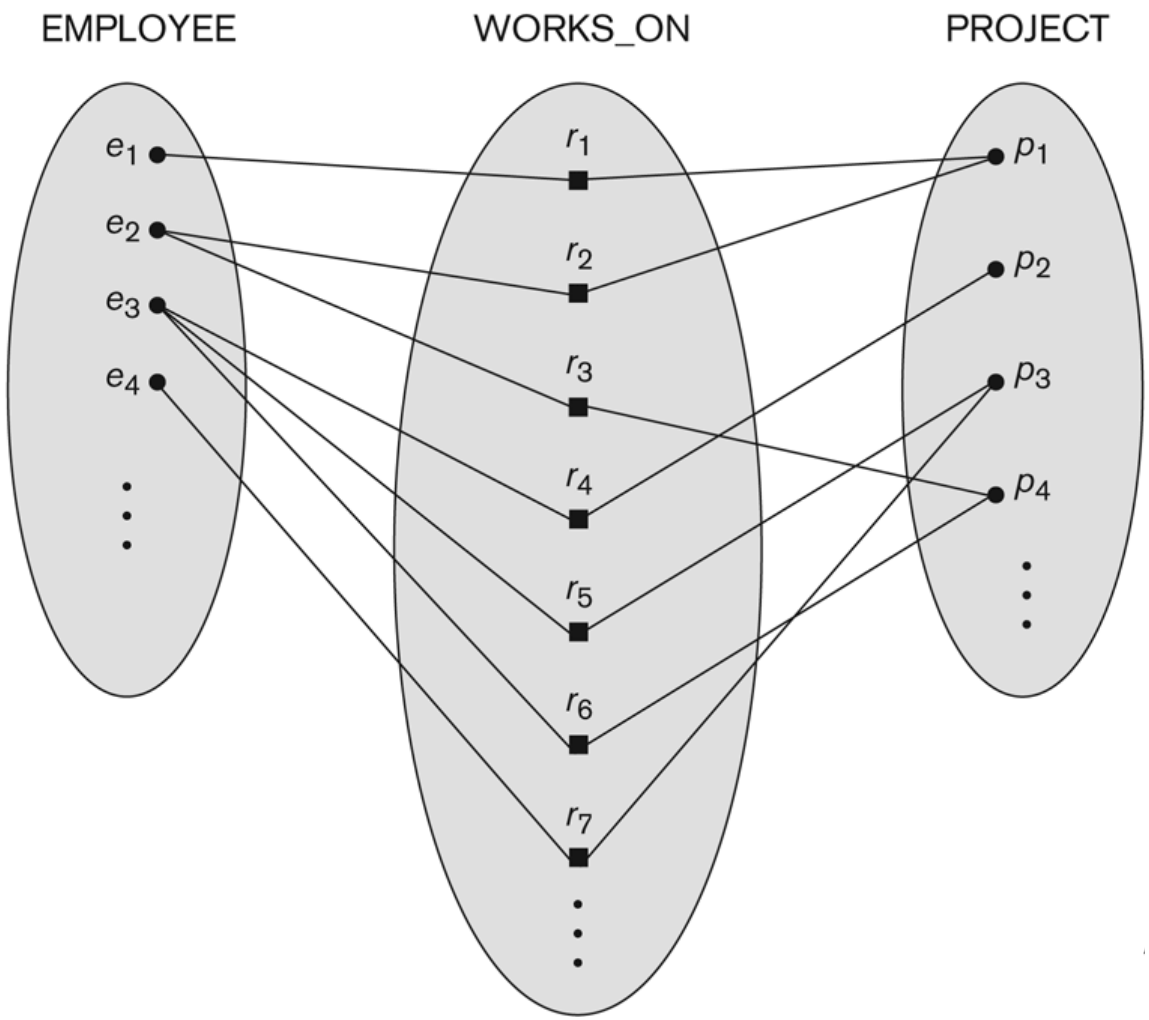
Relationships and Relationship Types (1)

- 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



Relationship Instances of the M:N WORKS_ON Relationship Between EMPLOYEE and PROJECT



Relationship Type vs. Relationship Set

- Relationship Type:
  - A set of associations among entities
  - 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 (contd.)

- 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

COMPANY Database Requirements

- The company is organised 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.
- Each employee works for one department but may work on several projects. We keep track of the number of hours per week that an employee currently works on each project, and the direct supervisor of each employee.
- Each employee may have a number of DEPENDENTs. For each dependent, we keep track of their name, sex, birthdate, and relationship to the employee.

COMPANY Database Requirements

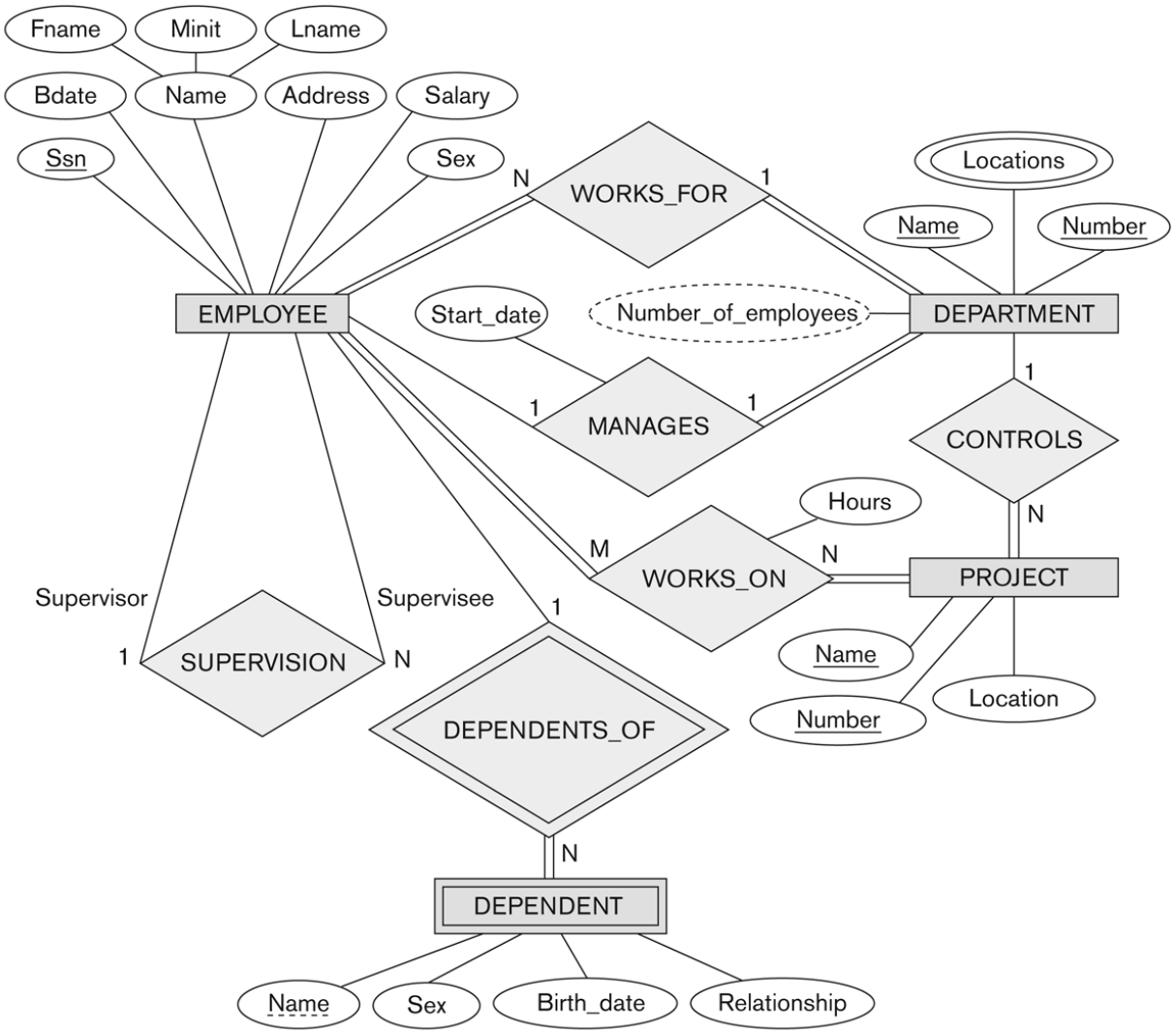
- The company is organised into DEPARTMENTs. Each department has a name, number and an employee who ***manages*** the department. We keep track of the start date of the department manager. A department may have several locations.
- Each department ***controls*** a number of PROJECTs. Each project has a unique name, unique number and is located at a single location.
- Each employee ***works for*** one department but may *work on* several projects. We keep track of the number of hours per week that an employee currently works on each project, and the *direct* ***supervisor*** of each employee.
- Each employee may ***have*** a number of DEPENDENTs. For each dependent, we keep track of their name, sex, birthdate, and relationship to the employee.

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



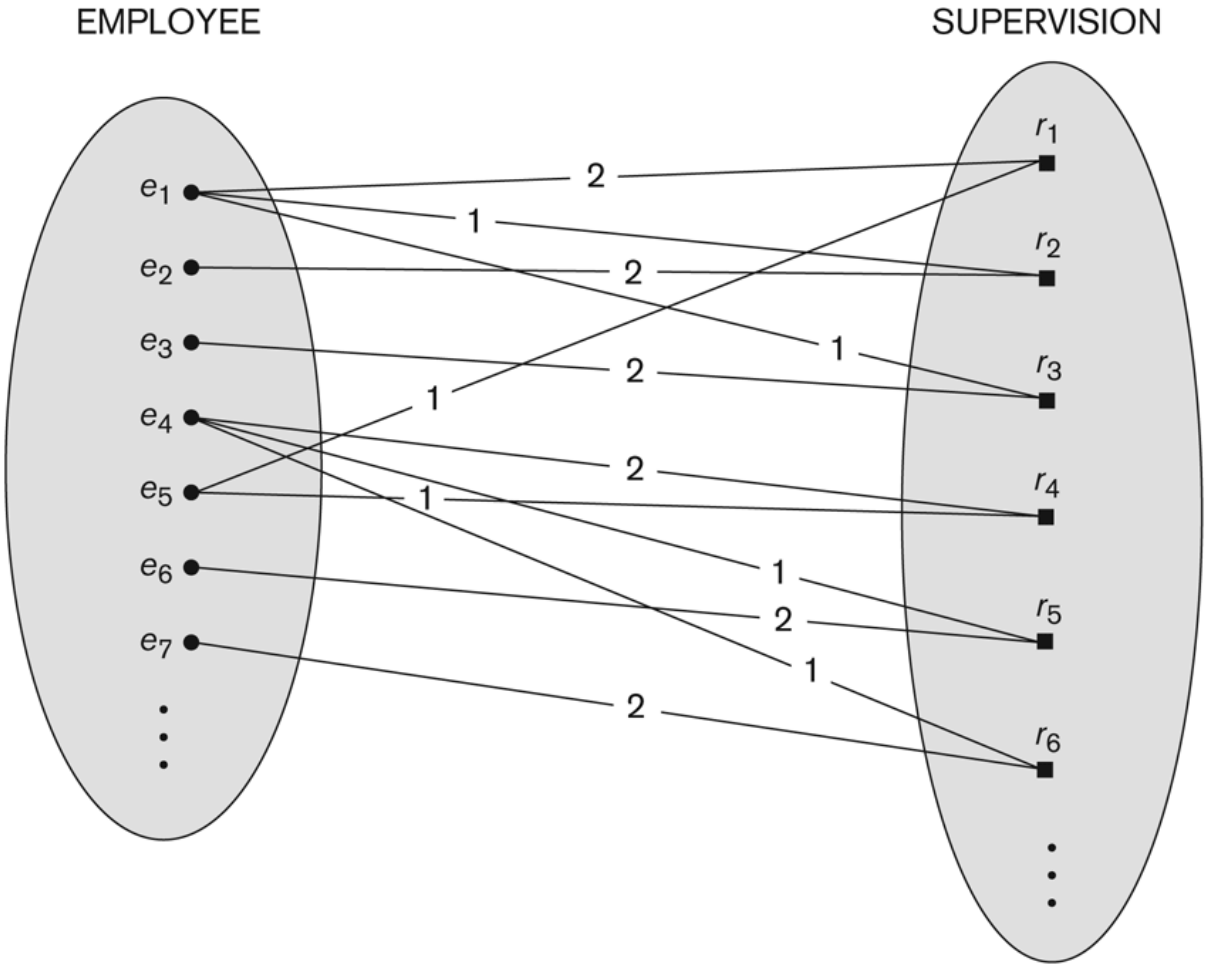
Discussion on Relationship Types

- 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.

Recursive Relationship Type

- A relationship type whose with the same participating entity type in distinct roles
- 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

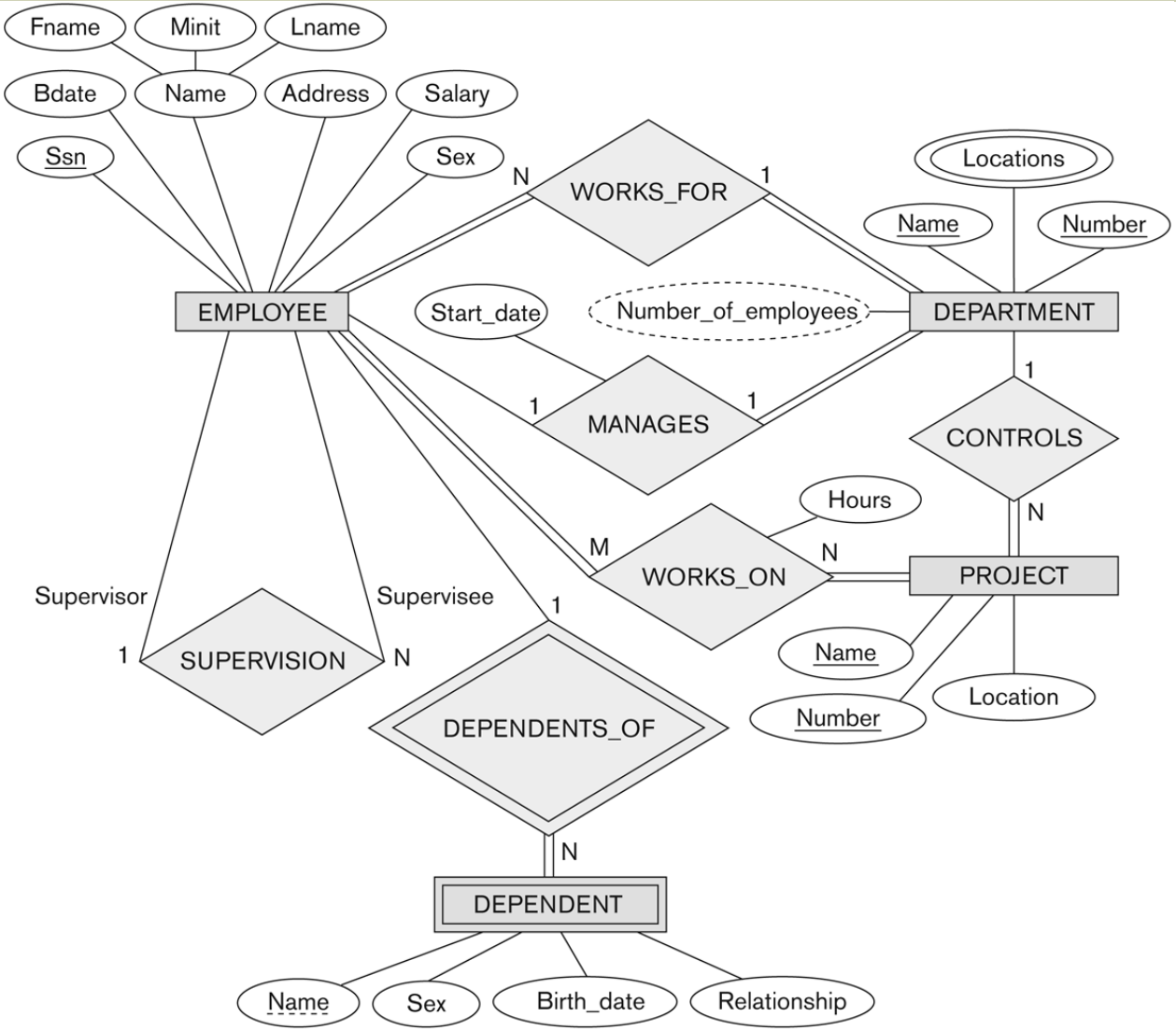
A Recursive Relationship Supervision



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 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.

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



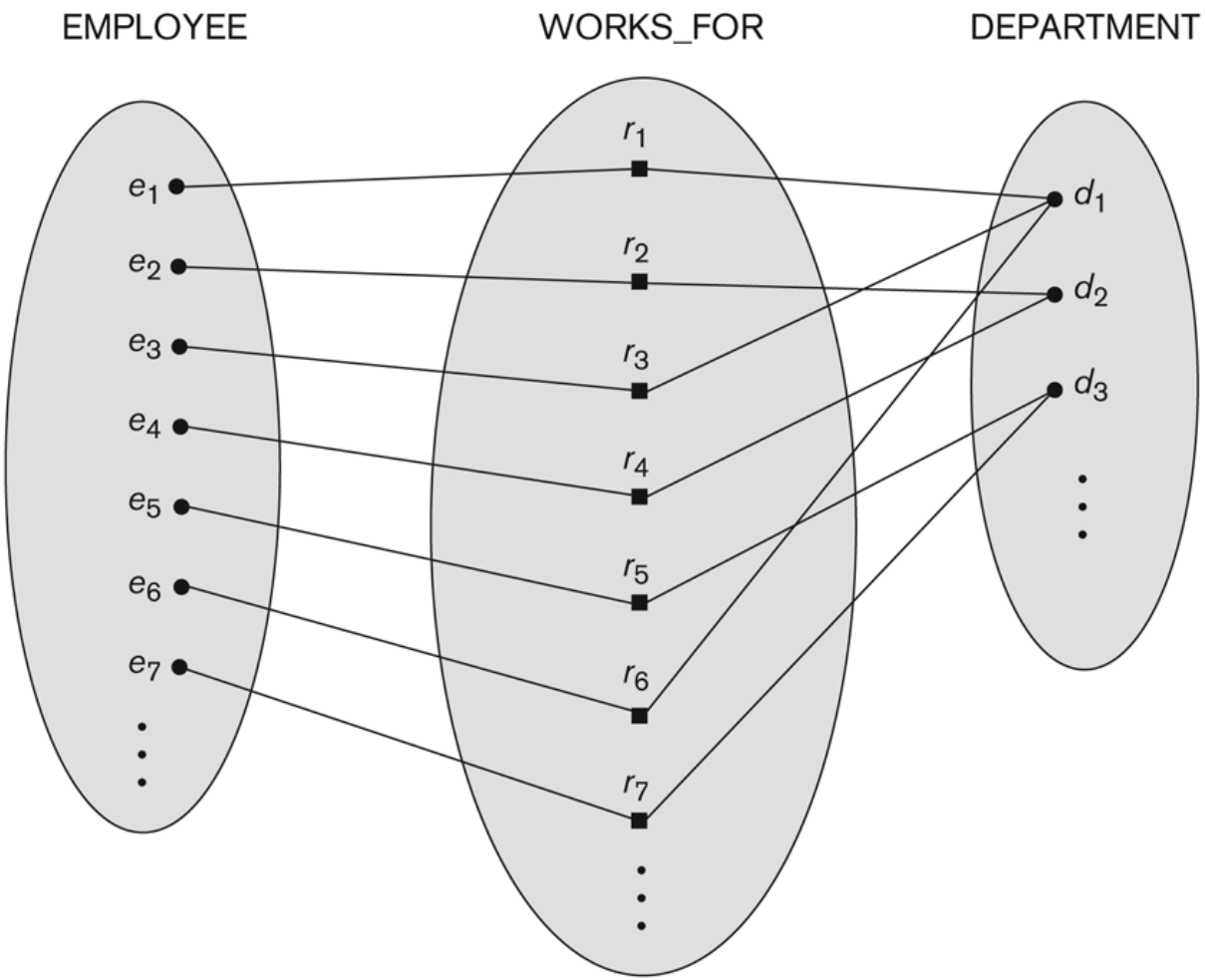
Weak Entity Types

- 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

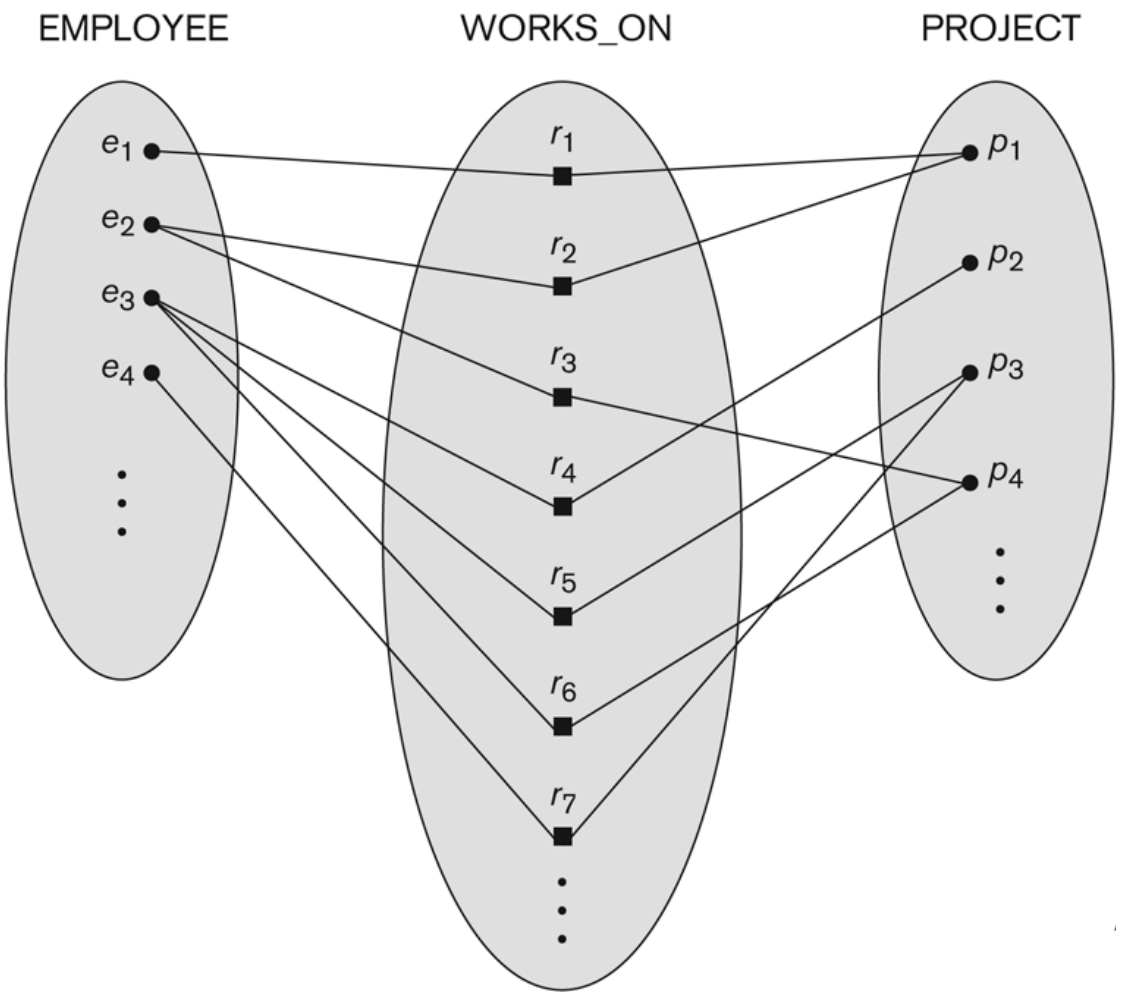
Constraints on Relationships

- Structural Constraints on Relationship Types
  - Also known as *ratio constraints.*
  - *Cardinality Ratio*: specifies *maximum* number of relationship instances that an entity can participate in.
    - 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, i.e. if existence of an entity depends on its being related to another entity via relationship type (also called *participation constraint*)
    - zero (optional participation, not existence-dependent)
    - one or more (mandatory participation, existence-dependent)

Many-to-one (N:1) Relationship



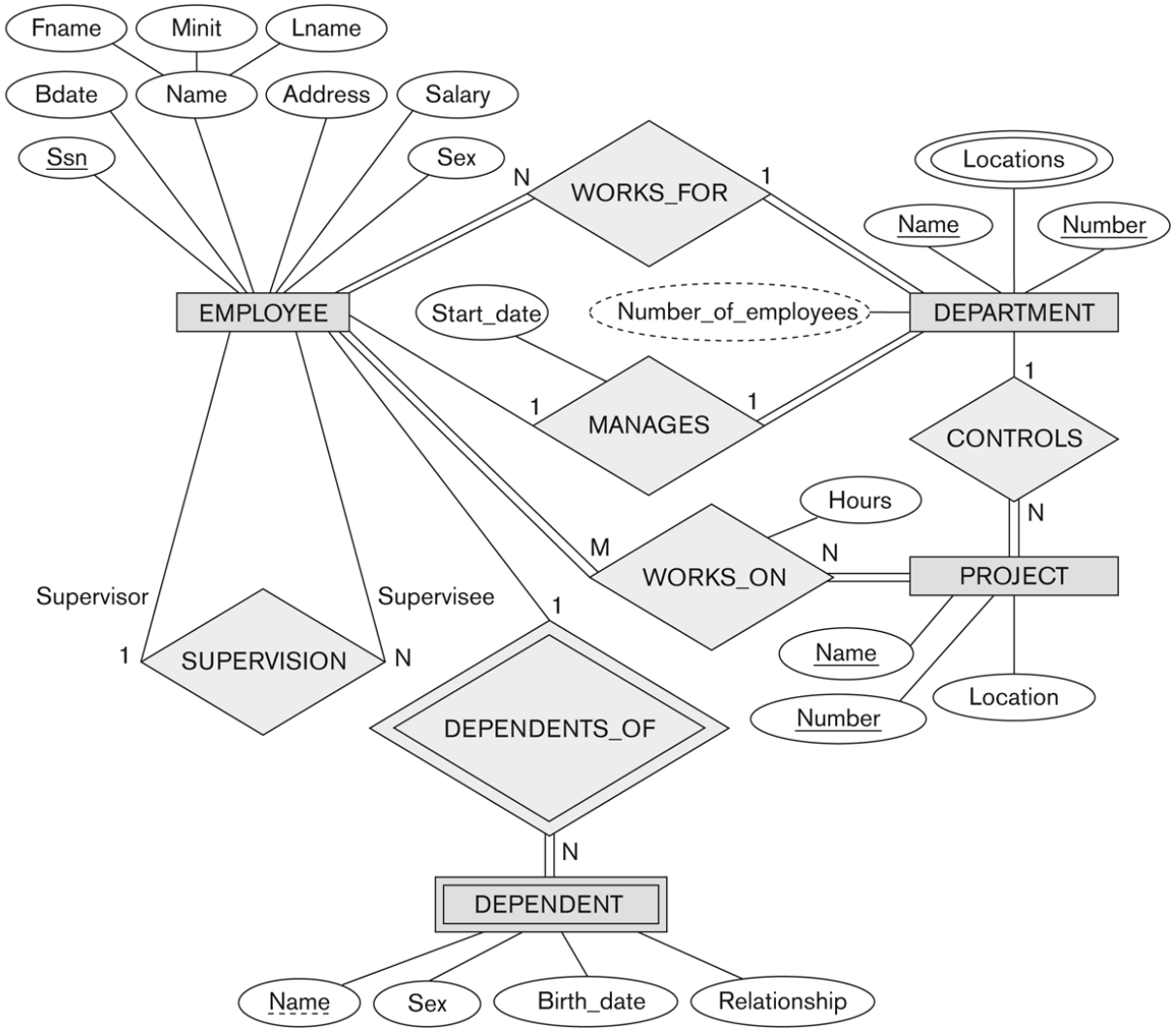
Many-to-many (M:N) Relationship



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



Notation for Constraints on Relationships

- Cardinality ratio (of a binary relationship): 1:1, 1:N, N:1, or M:N
  - 1:N vs. N:1
  - Shown by placing appropriate numbers on the relationship edges.
- Participation constraint (on each participating entity type)
  - total (called *existence dependency*)
    - shown by double line
  - partial.
    - shown by single line

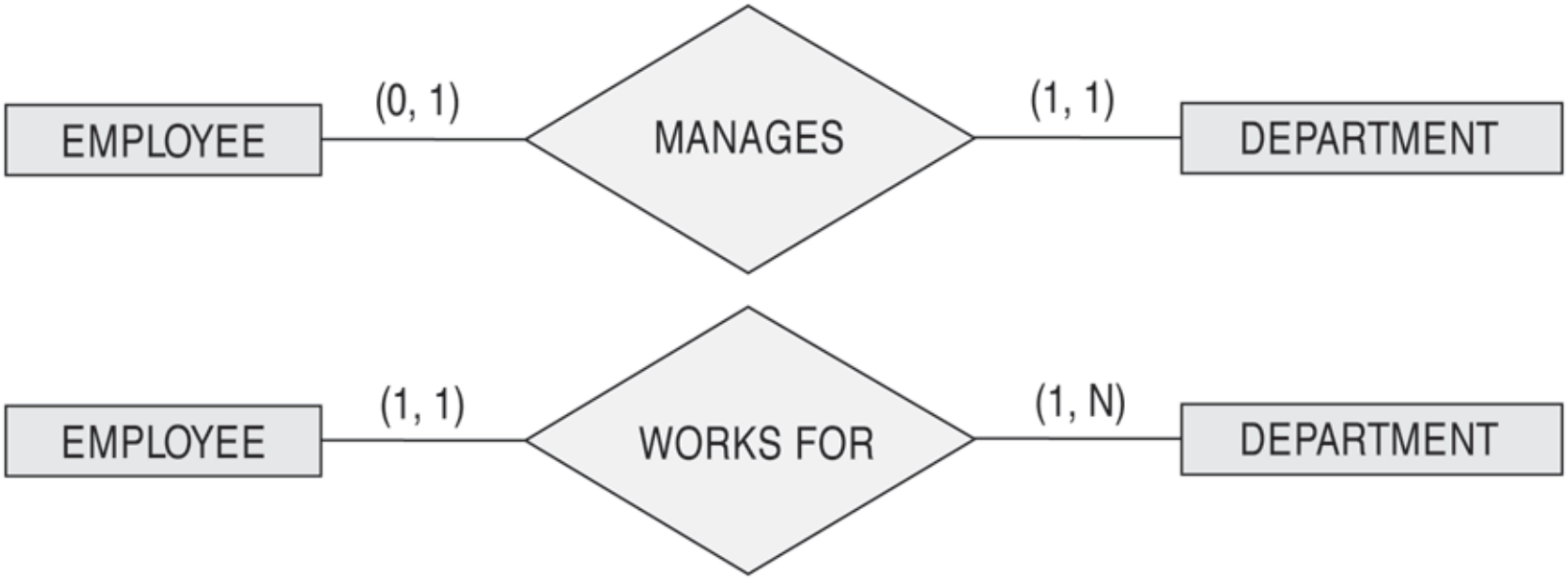
Alternative (min, max) Notation for Relationship (1)

- Structural Constraints:
  - Specified on each participation of an entity type E in a relationship type R
  - Specifies that each entity e in E participates in at least *min* and at most *max* relationship instances in R
  - Default(no constraint): min=0, max=n (signifying no limit)
  - Must have min≤max, min≥0, max ≥1
  - Derived from the knowledge of mini-world constraints

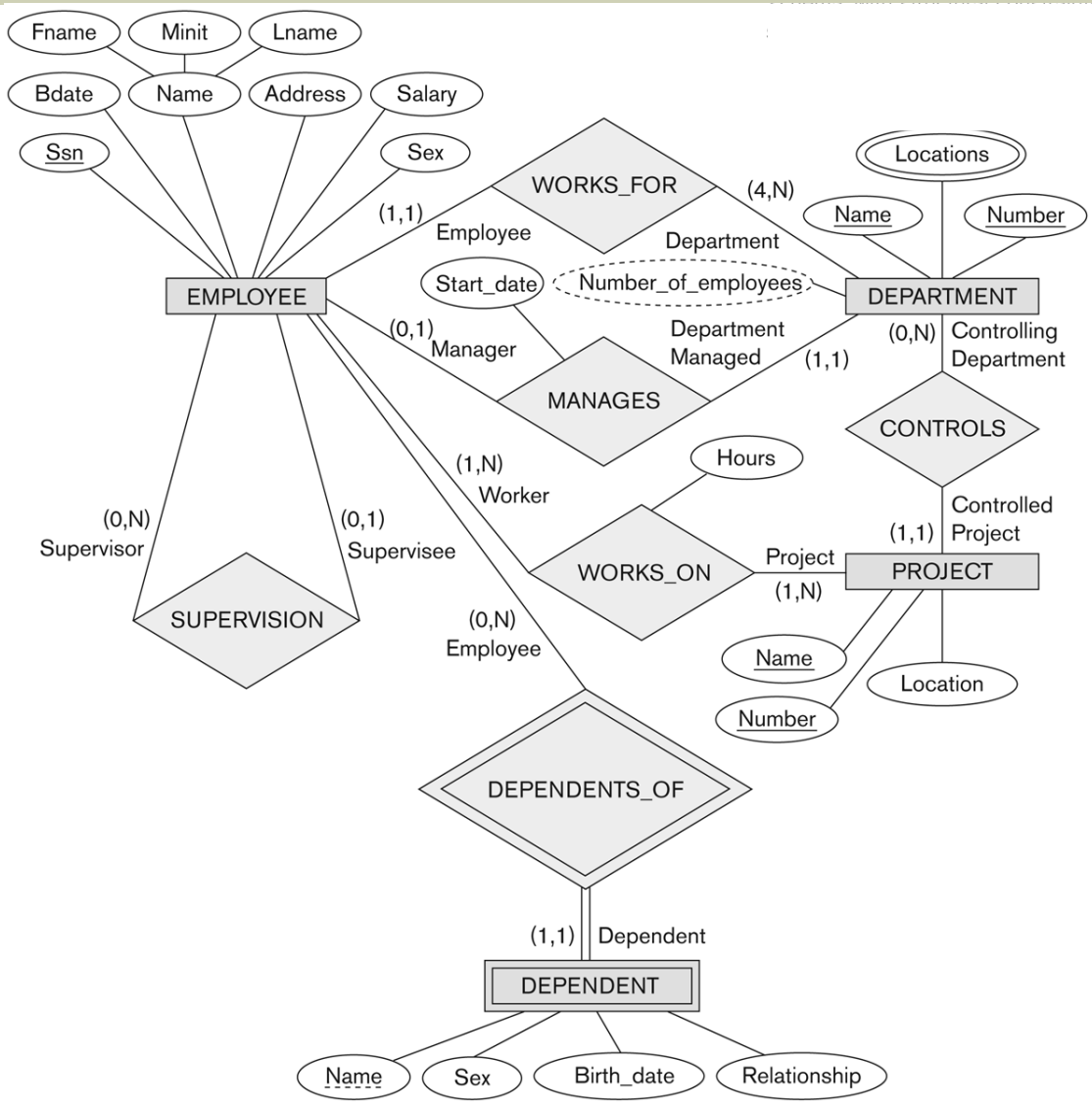
Alternative (min, max) Notation for Relationship (2)

- 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

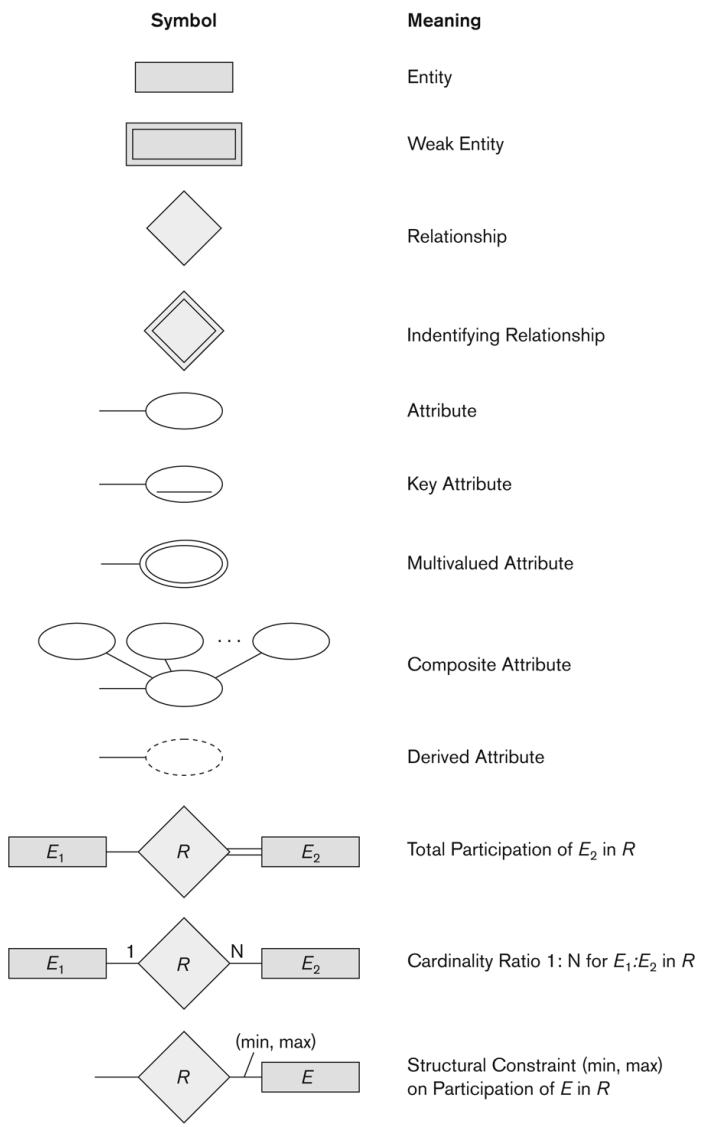
The (min,max) Notation for Relationship Constraints



COMPANY ER Schema Diagram using (min, max) notation



Summary of Notation for ER Diagrams

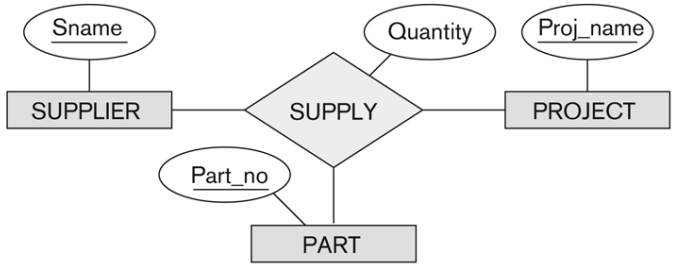


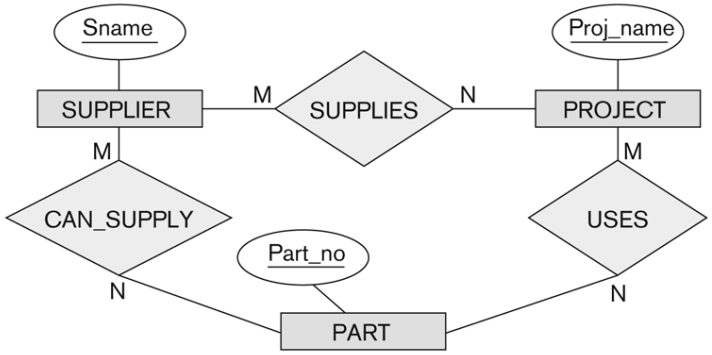
Relationships of Higher Degree

- 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)

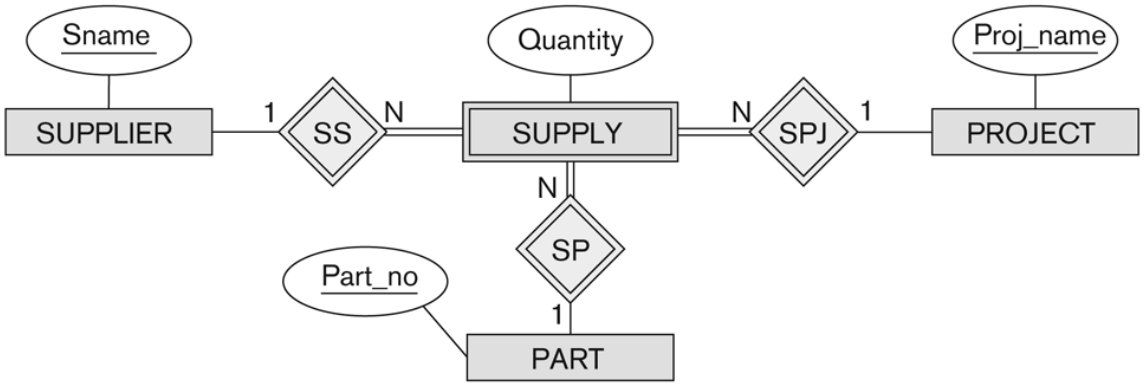
- In general, 3 binary relationships can represent different information than a single ternary relationship
- If needed, the binary and n-ary relationships can all be included in the schema design (where all relationships may convey different meanings)





Discussion of n-ary Relationships (n > 2)

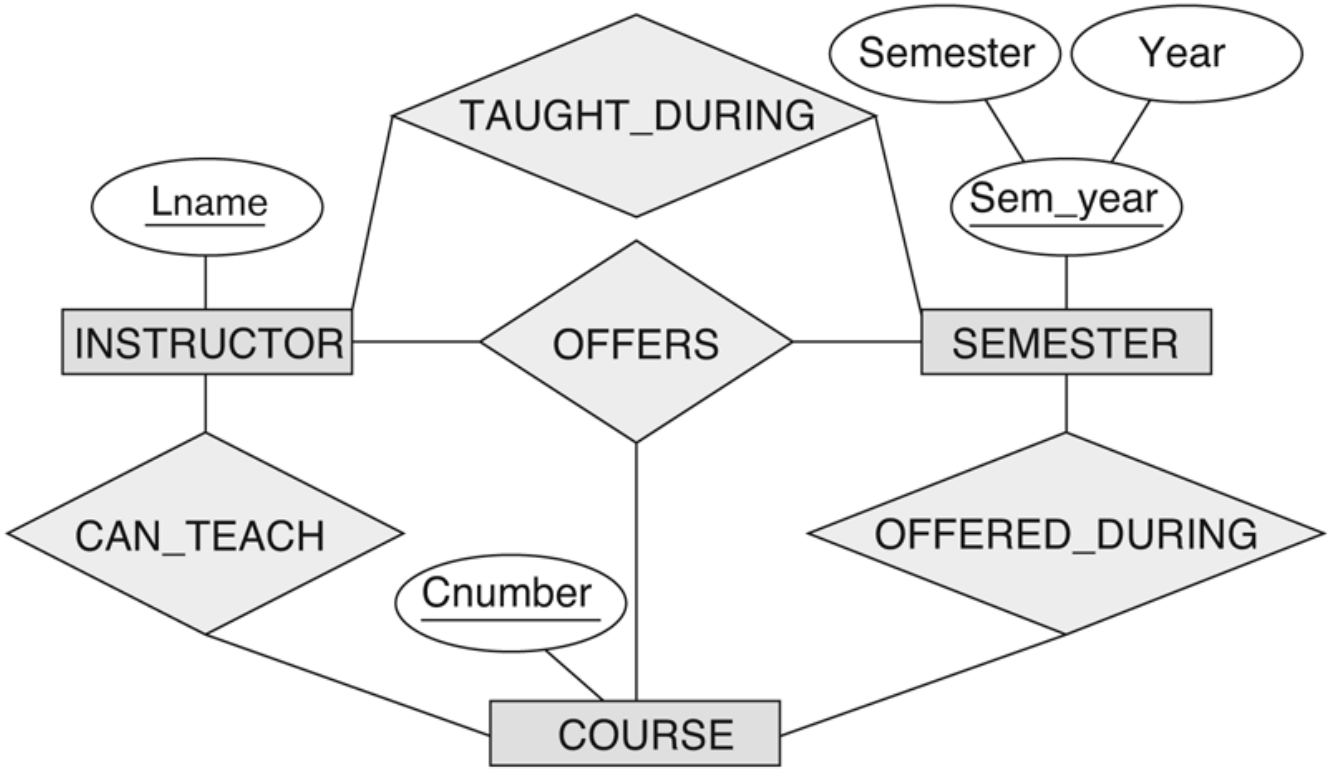
- In some cases, a ternary relationship can be represented as a *weak entity* if the data model allows a weak entity type to have multiple identifying relationships (and hence multiple owner entity types) (below)



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 (next slide) can be derived from the ternary relationship OFFERS (based on the meaning of the relationships).

Another Example of a Ternary Relationship



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*

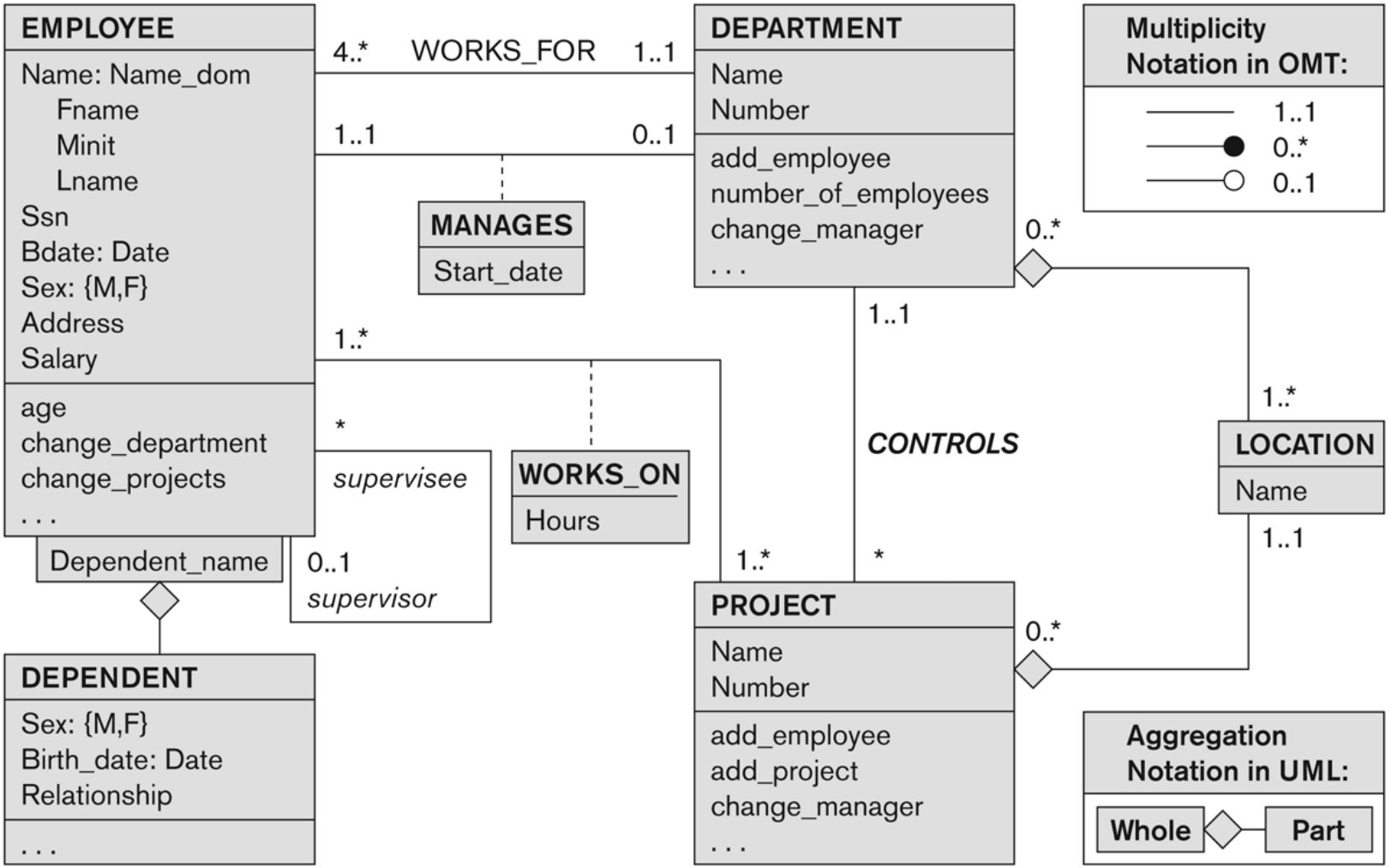
Alternative Diagrammatic Notation

- ER diagrams is a popular means for displaying database schemas.
- Many other notations exist in the literature and in various database design and modeling tools.
- UML class diagrams is representative of another way of displaying ER concepts that is used in several commercial design tools.

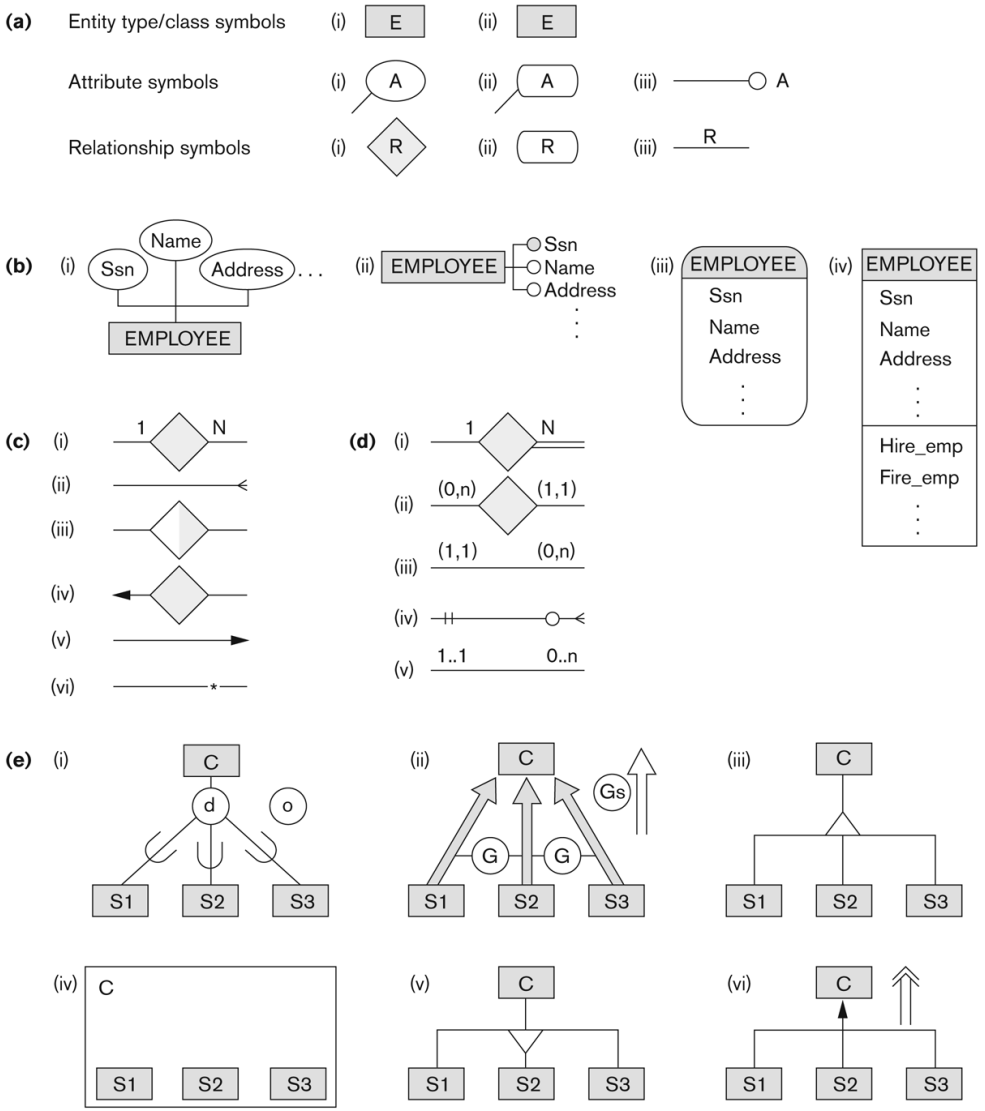
UML Class Diagrams

- Represent *classes* (similar to entity types) as large 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



Other Alternative Diagrammatic Notations



Chapter Summary

- ER Model Concepts: Entities, attributes, relationships
- Constraints in the ER model
- Using ER in step-by-step conceptual schema design for the COMPANY database
- ER Diagrams - Notation
- Alternative Notations – UML class diagrams