



# Lecture 5

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

# Agenda



Overview of Database Design Process



Example of Database Application (COMPANY)



Entities



Attributes



Relationships



Weak Entity



Final ER Diagram



# Overview of Database Design Process

# Overview of Database Design Process

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

# Overview of Database Design Process

## Purpose of ER Model:

- The ER model allows us to sketch database designs.
  - Kinds of data and how they connect.
  - **Not** how data changes.
- Designs are pictures called *Entity-Relationship diagrams*.
- Later: convert ER designs to relational DB designs.



# Example of Database Application (COMPANY)

# COMPANY Database

We need to create a database schema design based on the following (simplified) **requirements** of the COMPANY Database:

## ➤ Step # 1 : Requirements Collection & Data Analysis

- The COMPANY database keeps track of:
  - ✓ employees and their dependents,
  - ✓ departments, and
  - ✓ projects
- Database designers formulate a “Mini-world” Description

# “Mini-world” Description

- Concerning *the Department*:
  - *company is organized into departments*
  - *a department has a unique name, a unique number, and a specific employee is its manager*
  - *we track the start date for the manager*
  - *a department may be in several locations*
  - *a department controls a number of projects*
- Concerning *the Project*:
  - *a project has a unique name, a unique number, and is in a single location*



# “Mini-world” Description

- Concerning *the Employee*:
  - *each employee has a name, social insurance number, address, salary, gender, and birth date*
  - *an employee is assigned to one department but may work on several projects which are not necessarily controlled by the same department*
  - *we track the number of hours per week that an employee works on each project*
  - *we keep track of the direct supervisor of each employee*
  - *we track the dependents of each employee (for insurance purposes)*

# “Mini-world” Description

- Concerning *the Dependent*
  - *we record each dependent's first name, gender, birth date, and relationship to the employee*
- **Step # 2 : Build the database ER diagram (conceptual schema design):**

# Entities



# Entities

- 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

# Entity Type & Entity Set

- Each **Entity Type** will have a collection of entities stored in the database
  - Called the **Entity Set**
- Same name (EMPLOYEE) 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

# Entities

- In ER diagrams, an entity is displayed in a rectangular box

employee

dependent

department

project

# Attributes



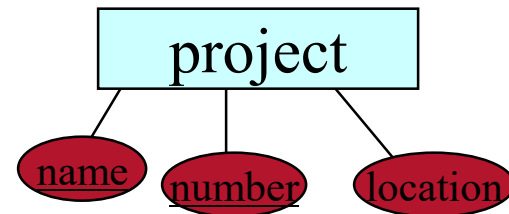
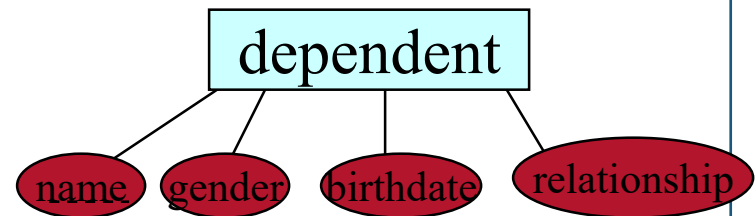
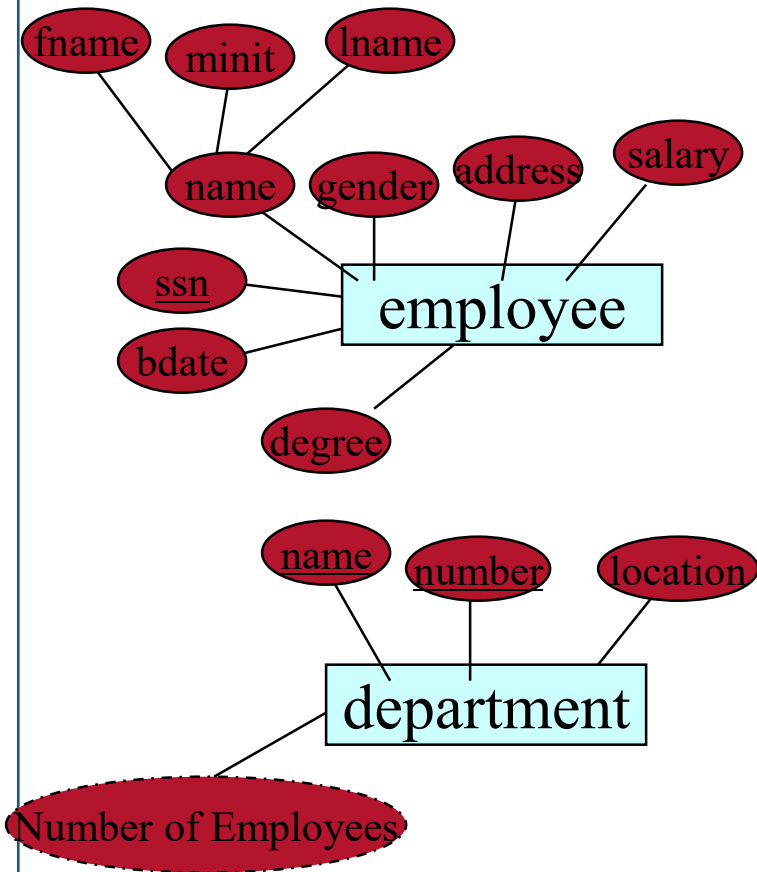


# Attributes

- 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 data type) associated with it – e.g. integer, string, subrange, ...

# Attributes

➤ In ER diagrams, an attribute is displayed in an oval



# Types of Attributes

## ➤ 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.

## ➤ Multi-valued

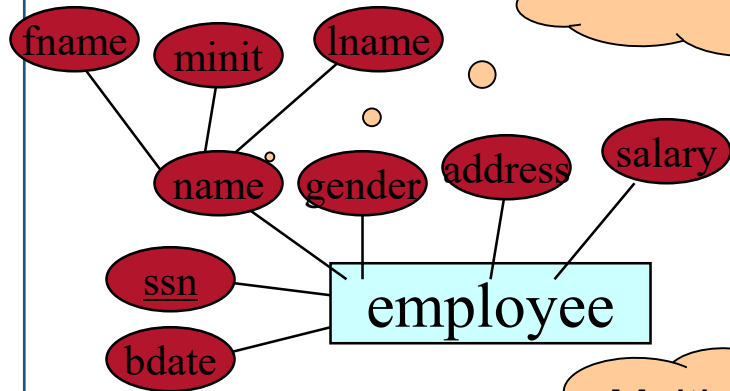
- An entity may have multiple values for that attribute. For example, Previous Degrees of a STUDENT or Multiple Locations of a DEPARTMENT

## ➤ Derived

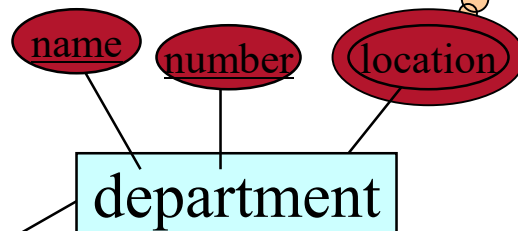
- Attributes do not exist in the physical database, but their values are derived from other attributes present in the database. For example, “Number of Employees”, should not be saved directly in the database, instead it can be derived.

# Attributes

Composite



Multi-valued



Derived

Number of Employees

dependent



project



Simple

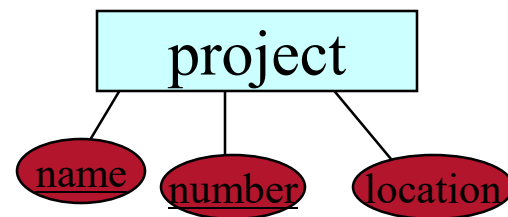
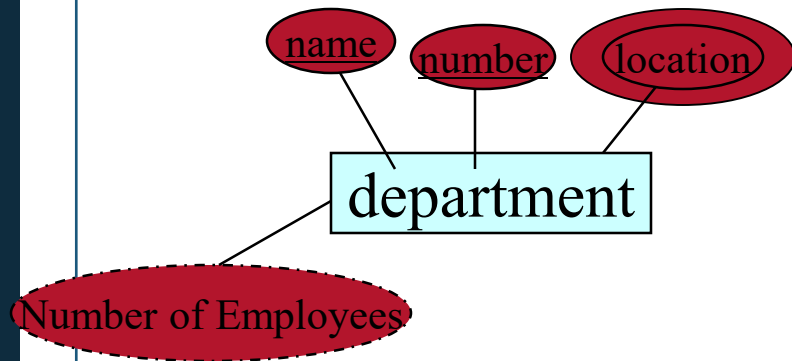
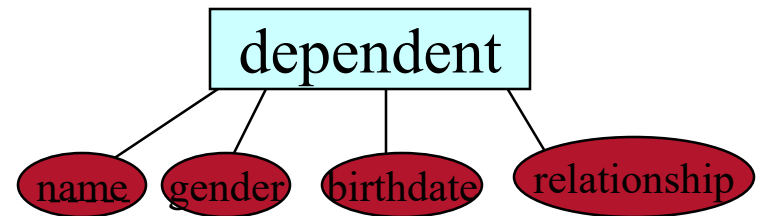
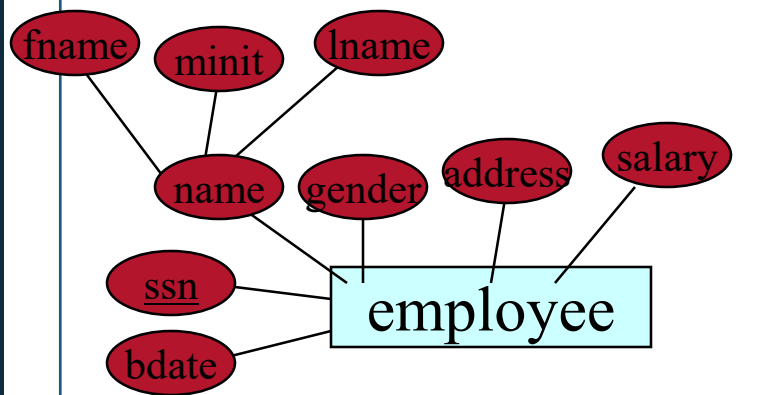
# Complex Attributes

- 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

# Key Attributes

- 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.
- Each key is underlined

# Attributes



# Relationships





## Refining the initial design by introducing relationships

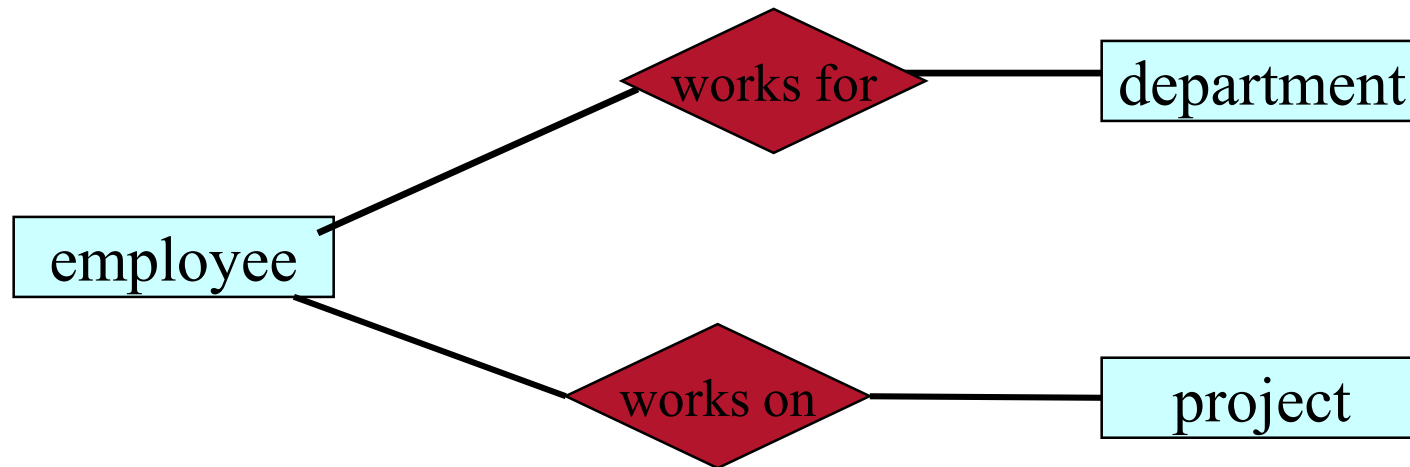
- The initial design is typically not complete
- Some aspects in the requirements will be represented as **relationships**
- ER model has three main concepts:
  - **Entities** (and their entity types and entity sets)
  - **Attributes** (simple, composite, multivalued)
  - **Relationships** (and their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

# Relationships and Relationship Types

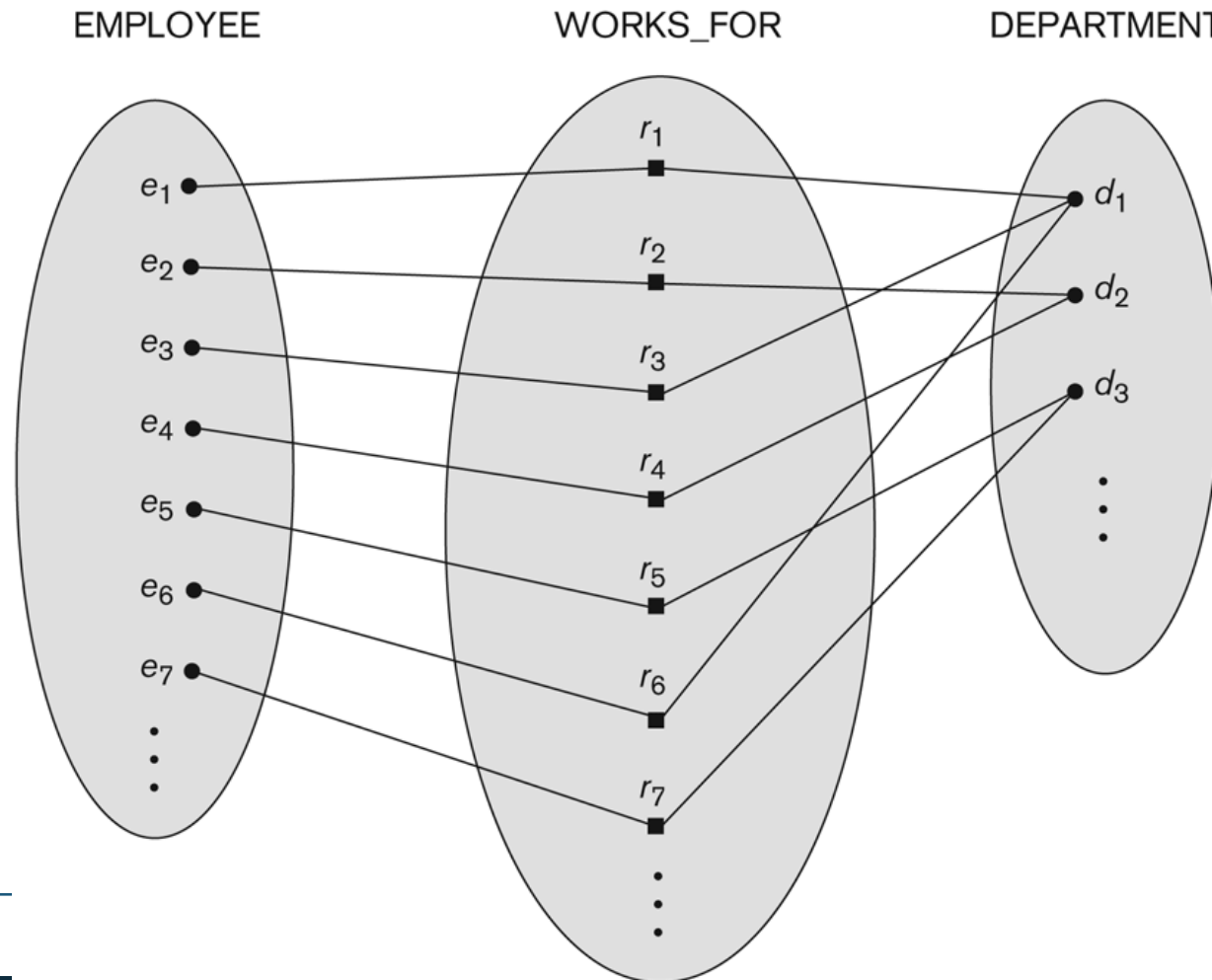
- A **relationship** relates two or more distinct entities with a specific meaning. For example:
  - EMPLOYEE John Smith *works on* ProductX PROJECT,
  - EMPLOYEE Franklin Wong *manages* Research DEPARTMENT.
- Relationships of the same type are grouped or typed into a **relationship type**. For example:
  - WORKS\_ON relationship type in which EMPLOYEEs and PROJECTs participate, or
  - MANAGES relationship type in which EMPLOYEEs and DEPARTMENTs participate.

# Relationships

- In ER diagrams, we represent the *relationship type* as: **Diamond-shaped box connected to the participating entity types via straight lines**



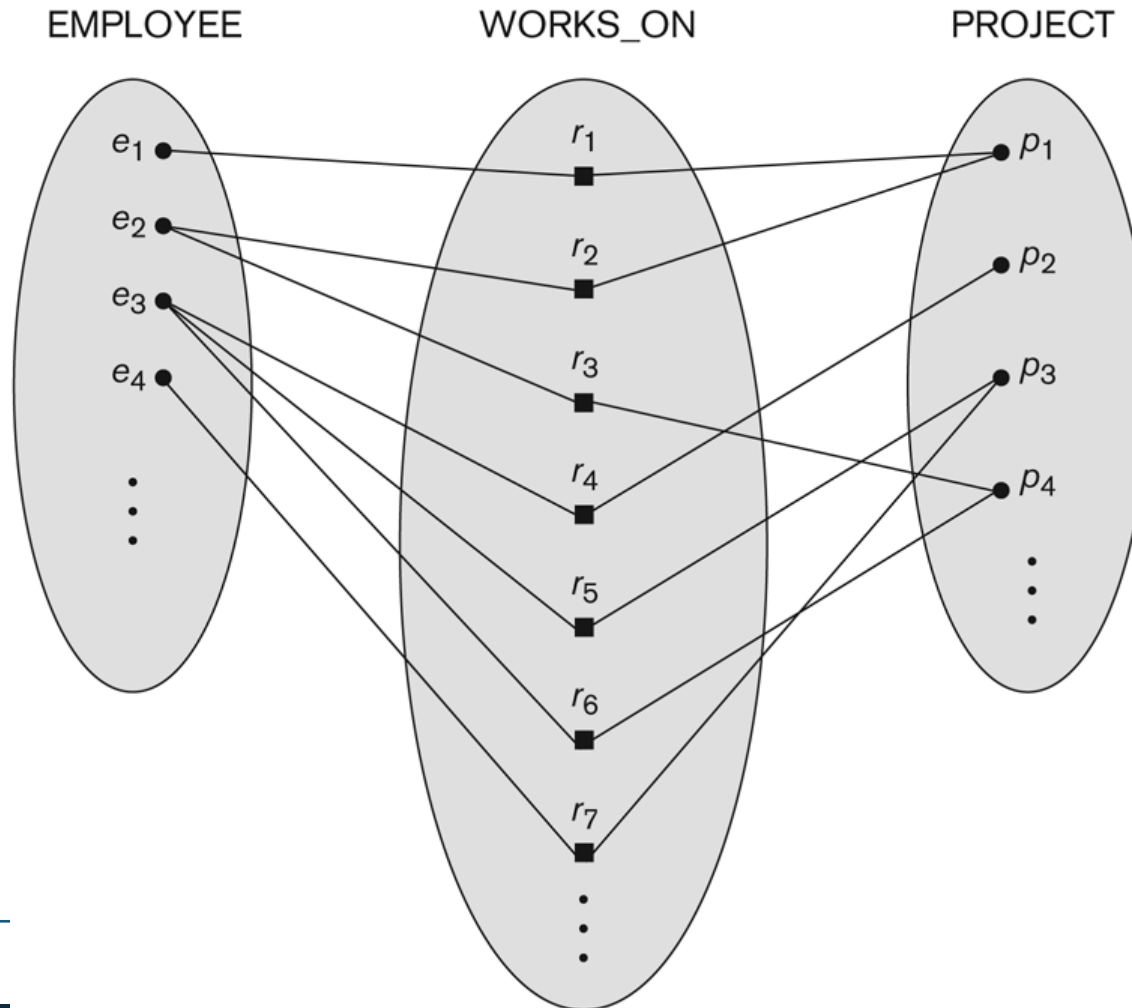
## Relationship instances of the WORKS\_FOR N:1 relationship between EMPLOYEE and DEPARTMENT



**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

## ➤ 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

# Degree of a relationship type

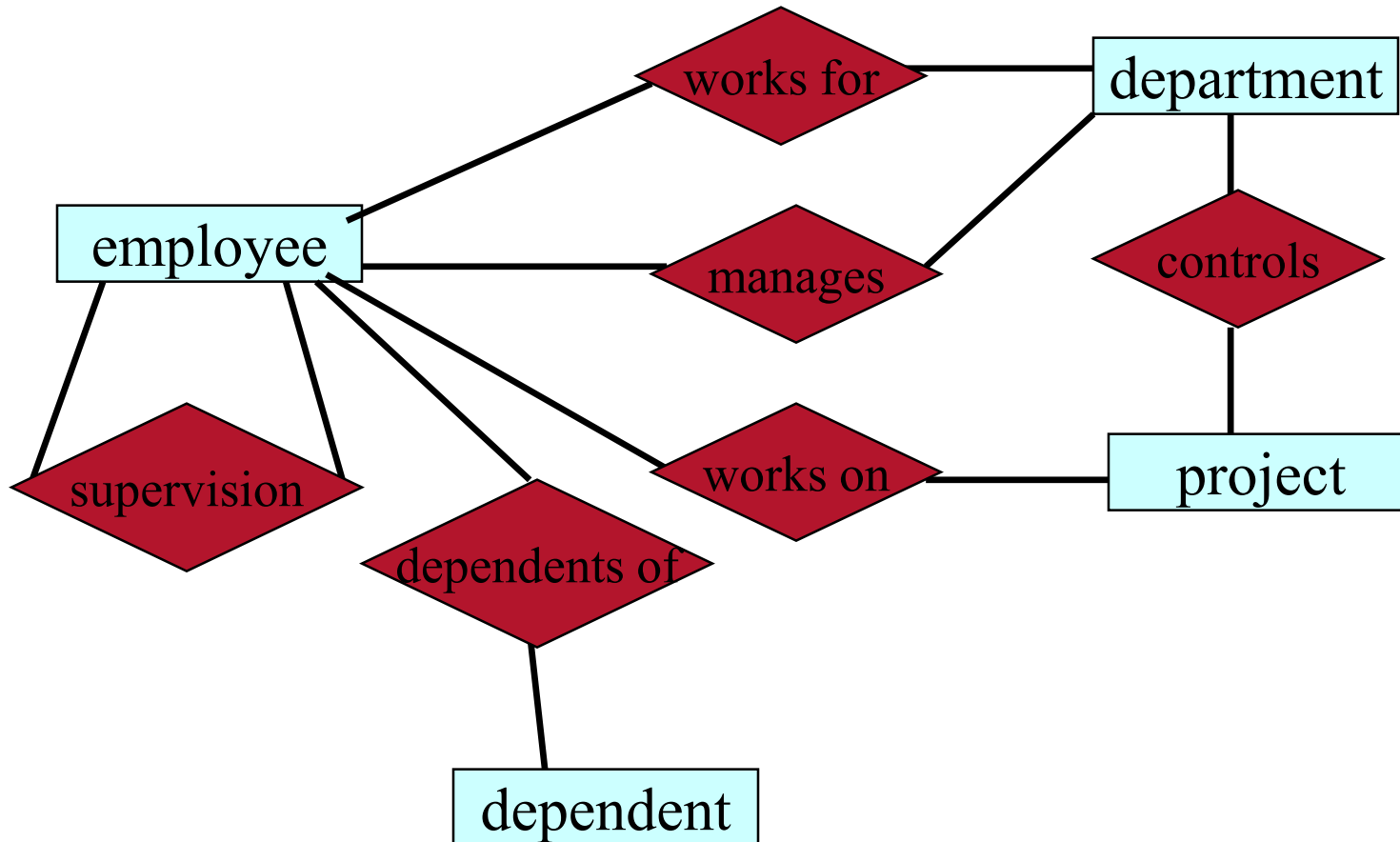
- The degree of a relationship type is the number of participating entity types.
  - Both MANAGES and WORKS\_ON are *binary* relationships.
- The “works\_for” relationship is of degree two → *binary*
- A relationship in which for example a supplier supplies parts to a project is of degree three → *ternary*

## Refining the COMPANY database schema by introducing relationships

- By examining the requirements, six relationship types are identified
- All are *binary* relationships
- 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 supervisee), EMPLOYEE (as supervisor))
  - DEPENDENTS\_OF (between EMPLOYEE, DEPENDENT)



# Relationships



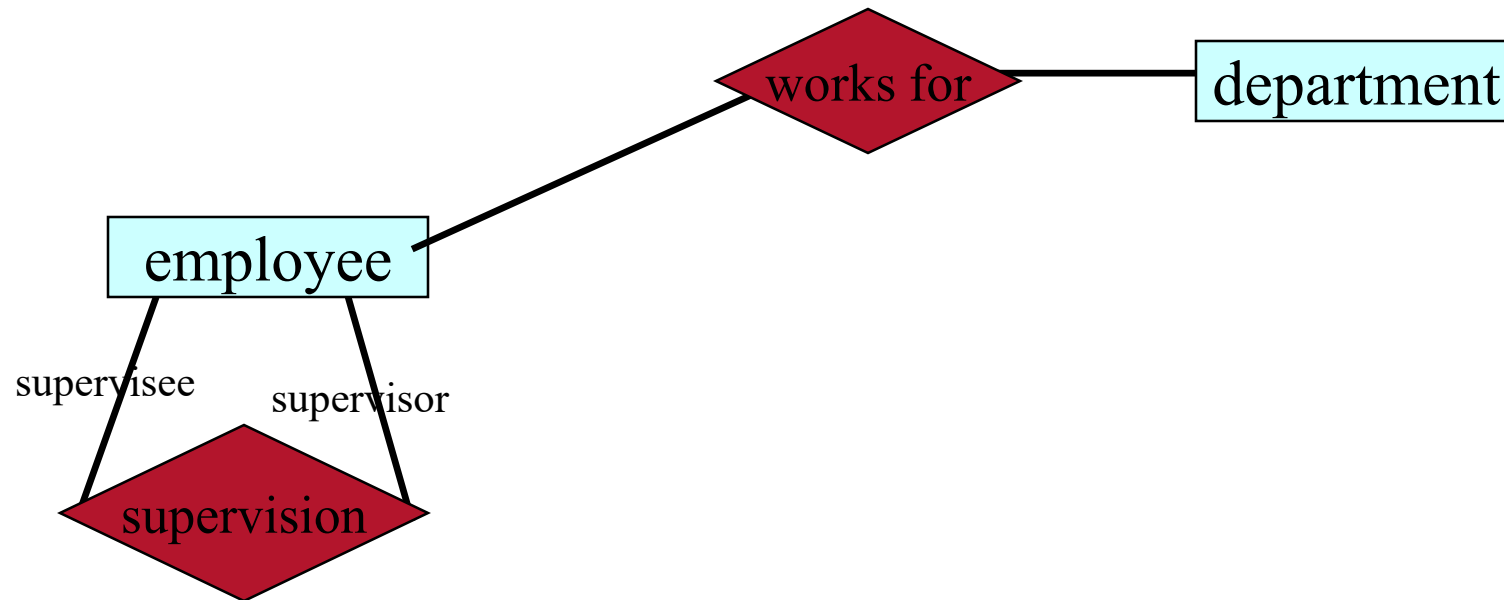
# Discussion on Relationship Types

- 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 with the same participating entity type acting 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

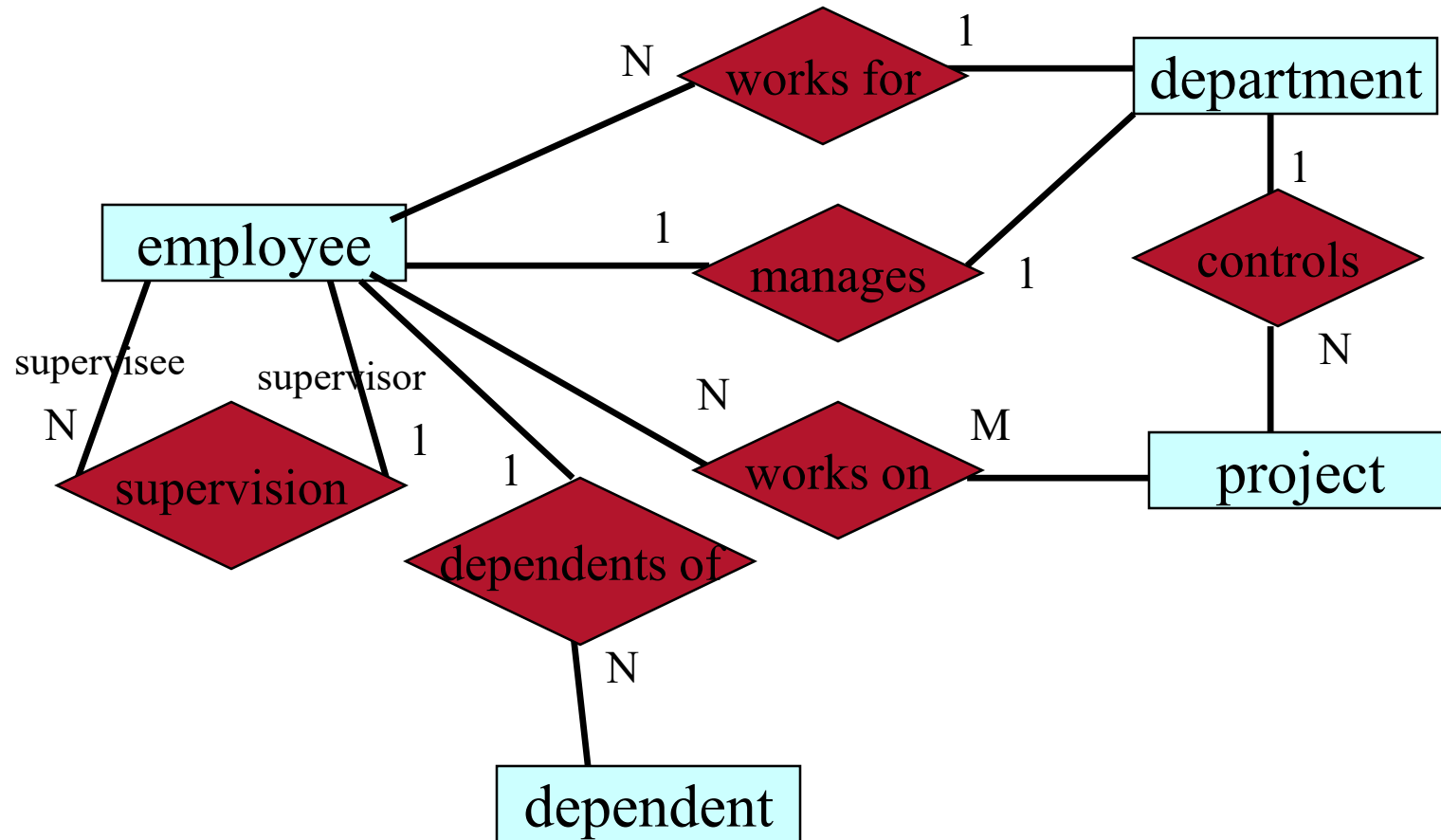
# Role Names & Recursive Relationships



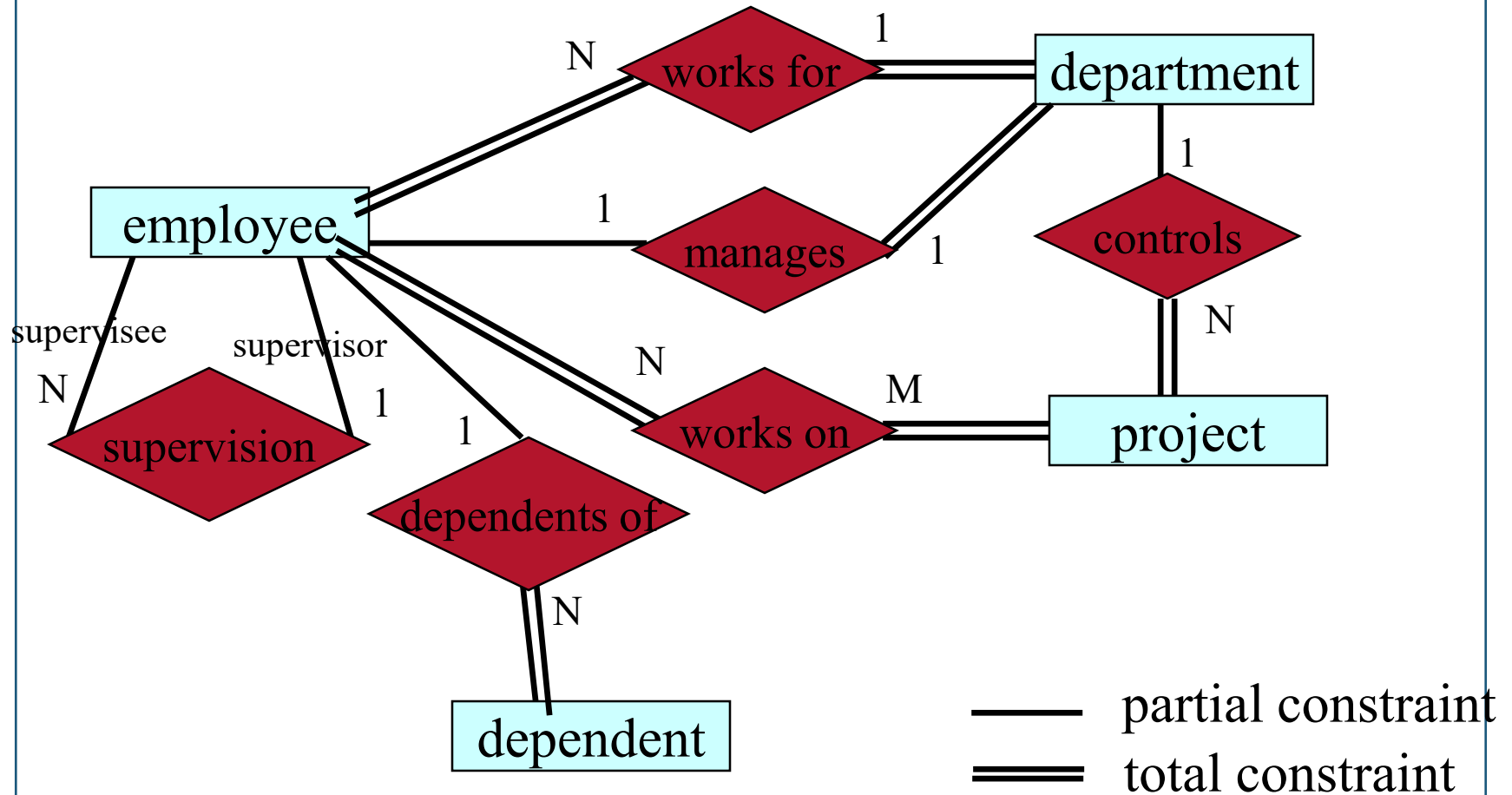
# Constraints on Relationships

- **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)
  
- **Participation Constraint** (Existence Dependency Constraint)
  - specifies *minimum* participation
  - zero (optional participation, not existence-dependent)
  - one or more (mandatory participation, existence-dependent)

# Constraints → Cardinality Ratio



# Constraints → Participation

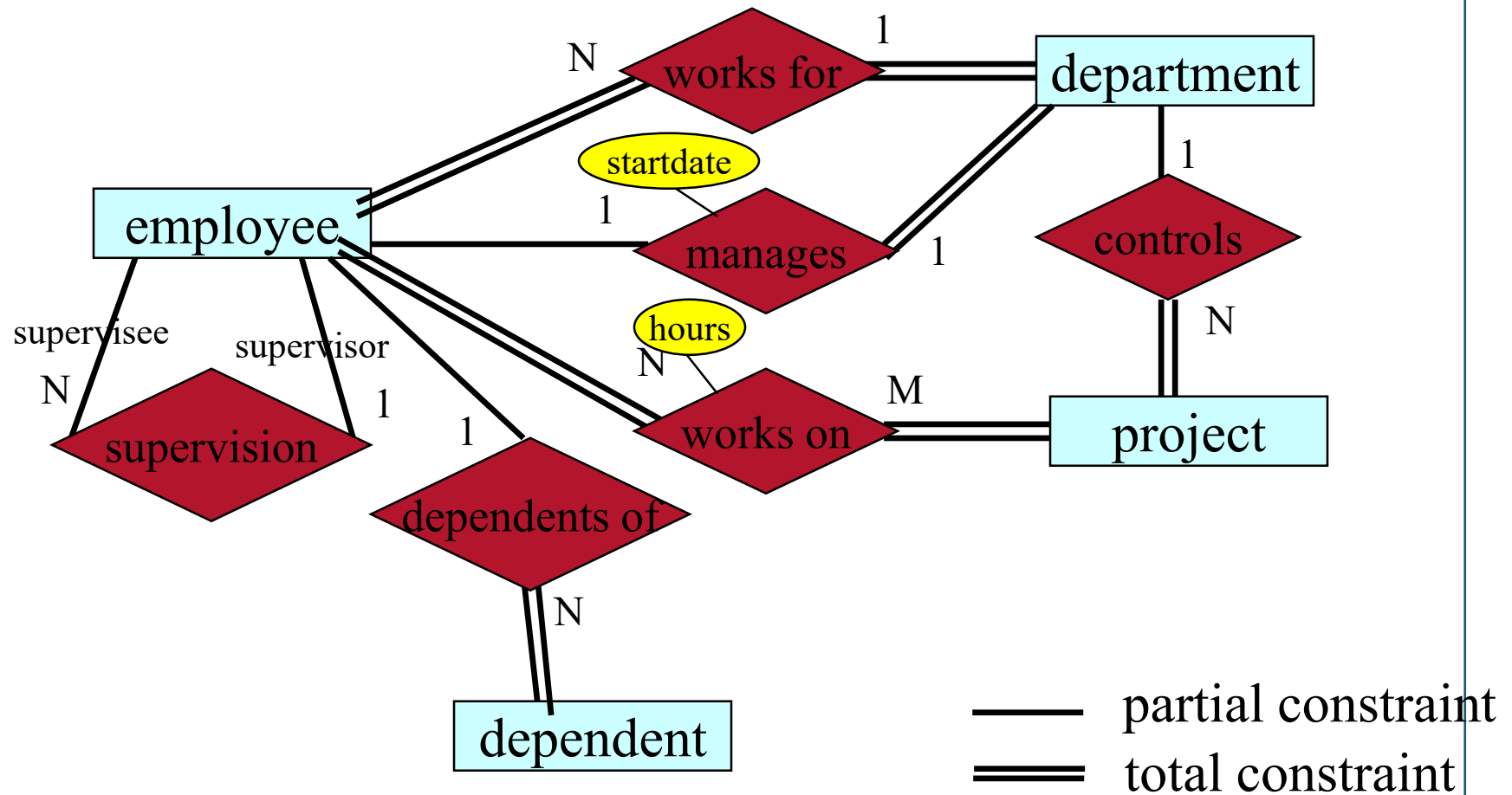


# Attributes of Relationship types

- 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



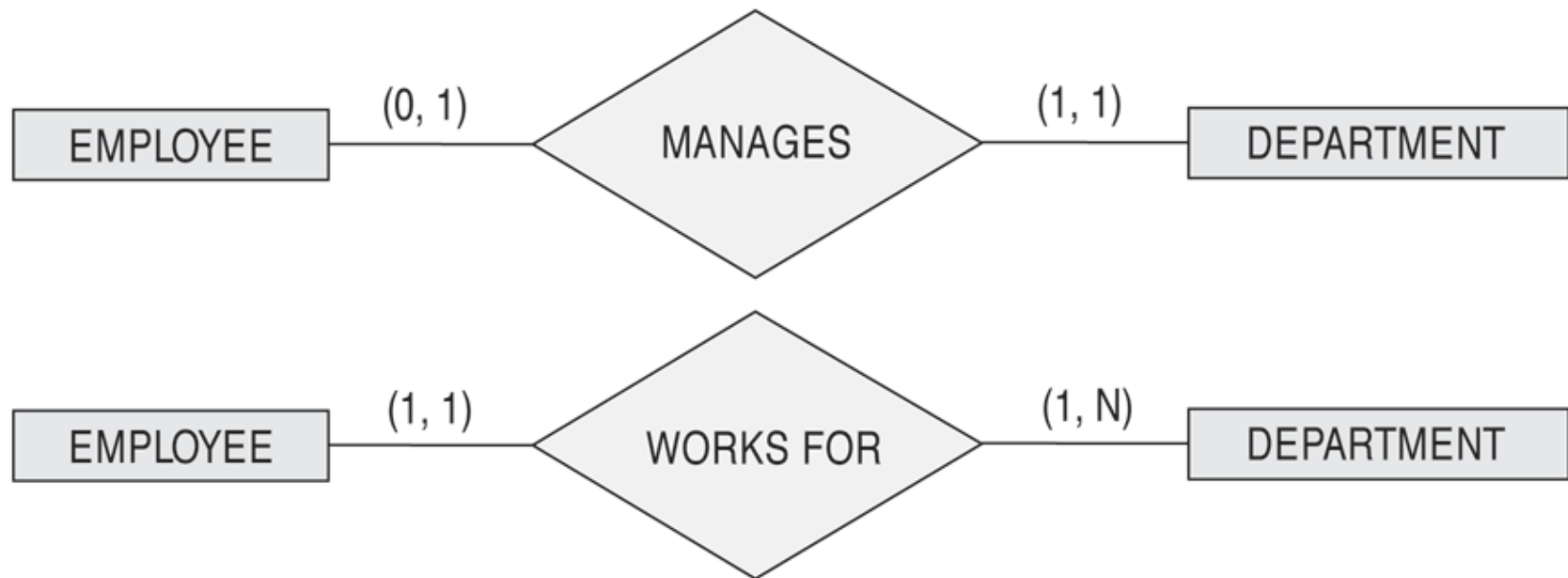
# Attributes of Relationships



# 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):  $\text{min}=0$ ,  $\text{max}=\infty$  (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 (1,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



# Weak Entity

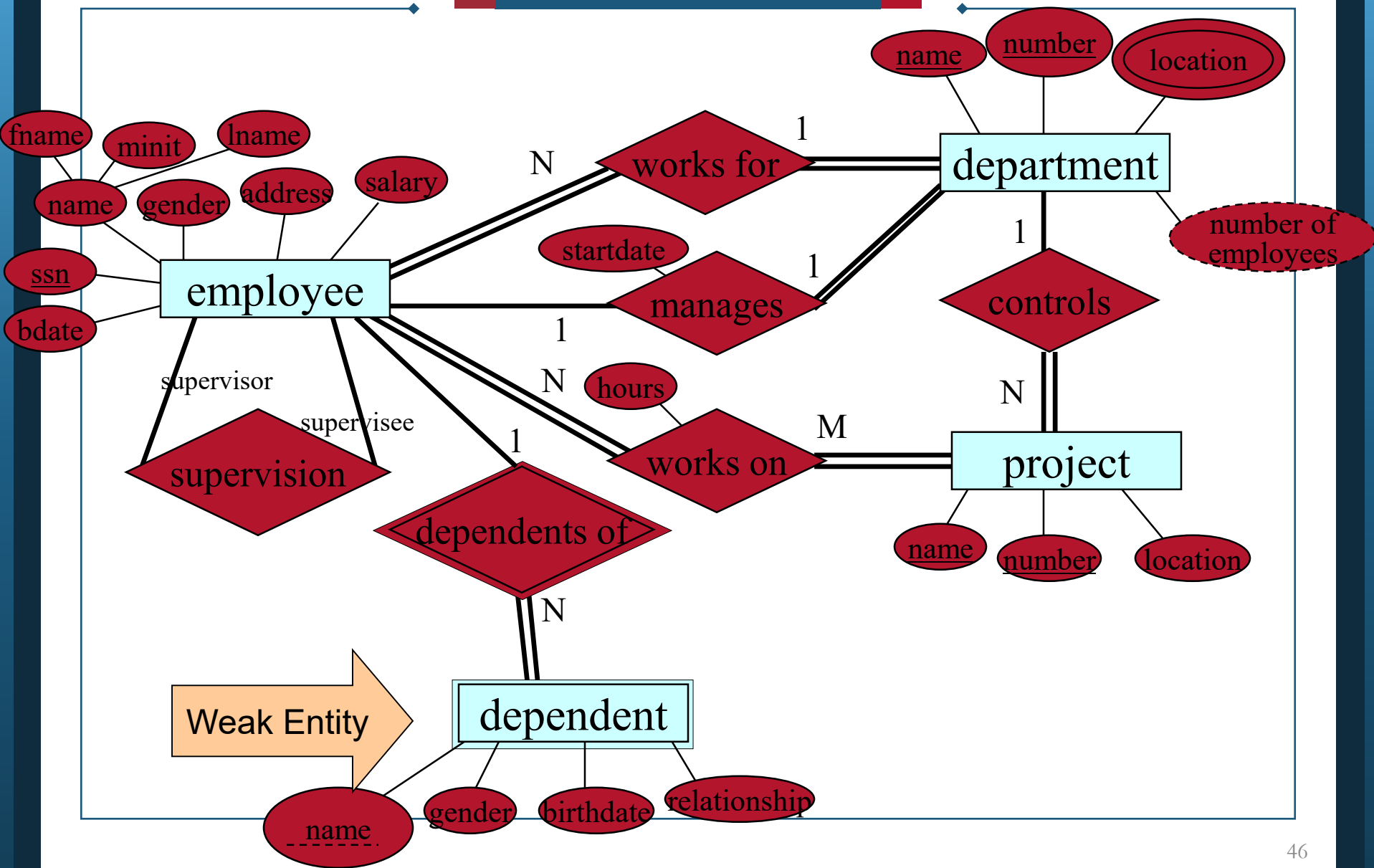
# Weak Entity Types

## ➤ Strong entity

- key attribute
- entity has a key attribute or a combination of attributes which can be used as a key.

## ➤ Weak entity

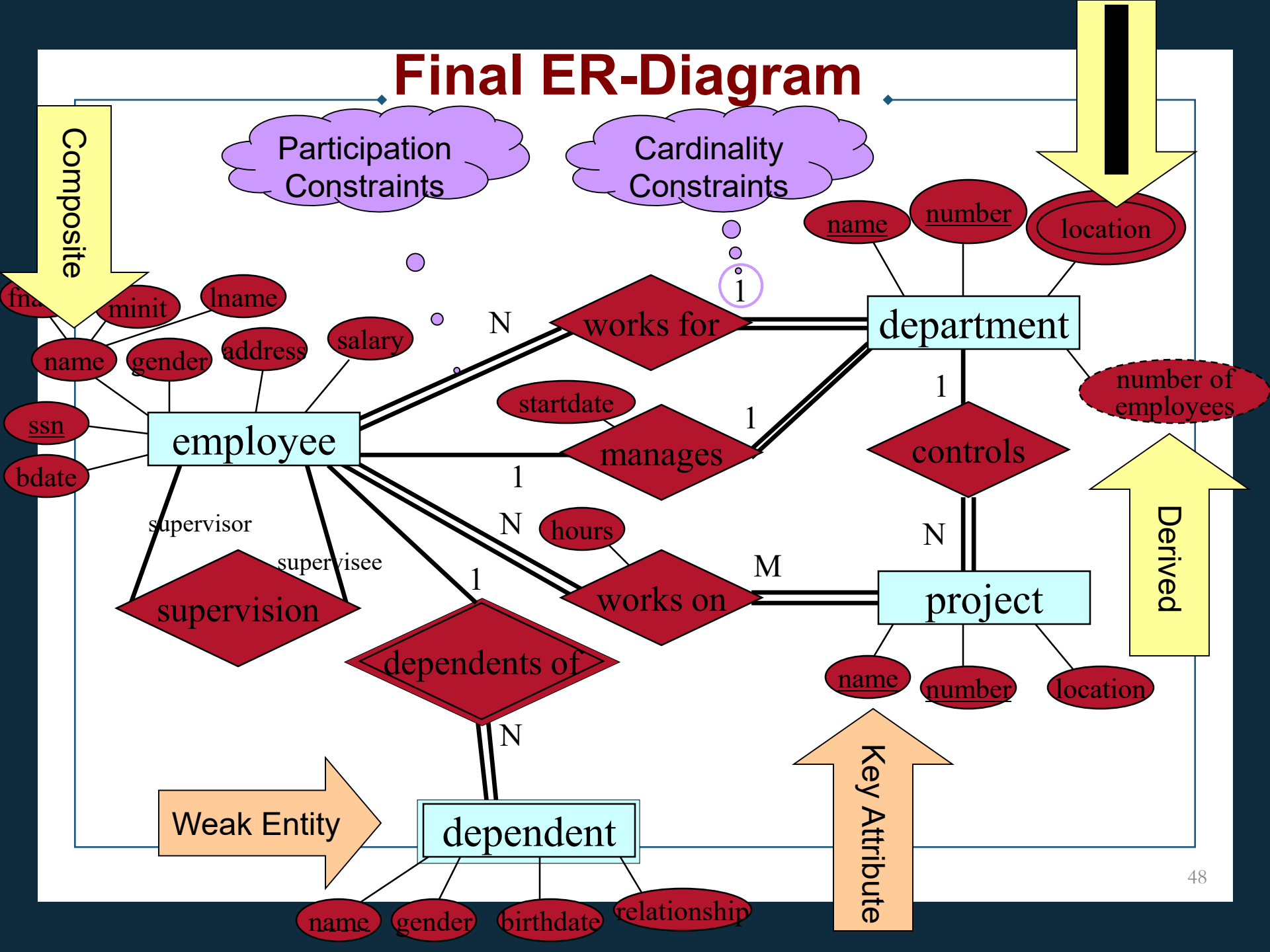
- No key attributes, may have a partial key
- related to specific entities from another entity type in combination with some of their attribute values.
- the identifying relationship will have total participation for the weak entity
- identifying owner





# Final ER Diagram

# Final ER-Diagram

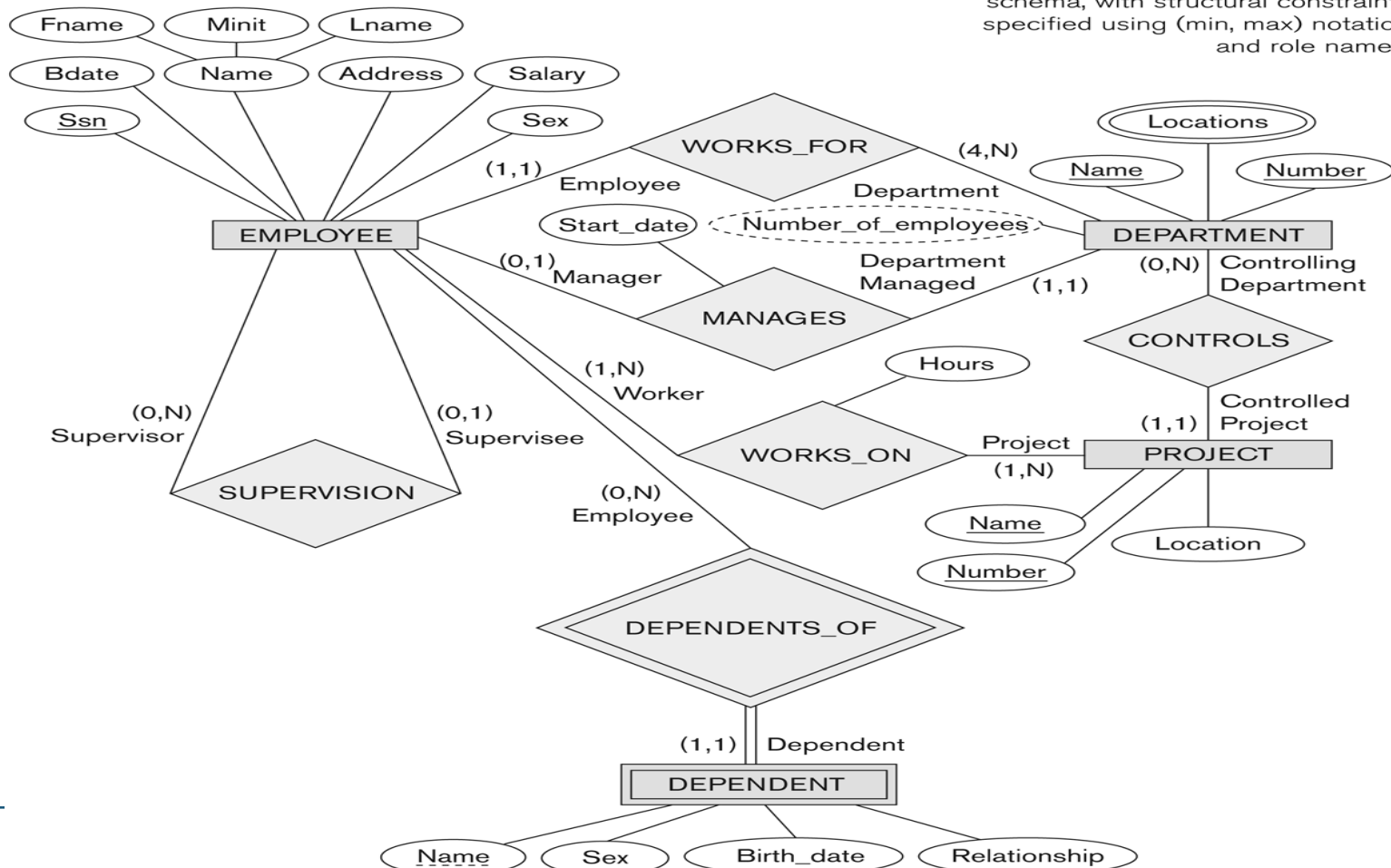




# COMPANY ER Schema Diagram using (min, max) notation


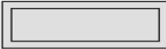
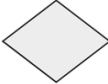
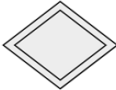



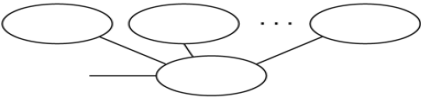

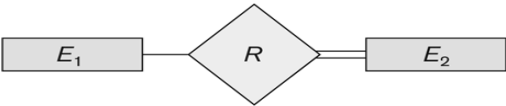

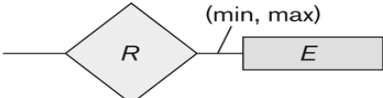
**Figure 3.15**

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



# Summary of notation

**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 $R$



# Thank You