CS 405G: Introduction to Database Systems

Entity – Relationship Model

Jinze Liu

Review

- A database is
  - a large collection of integrated data
- A miniworld is
  - some aspect of the real word, described by facts (data)

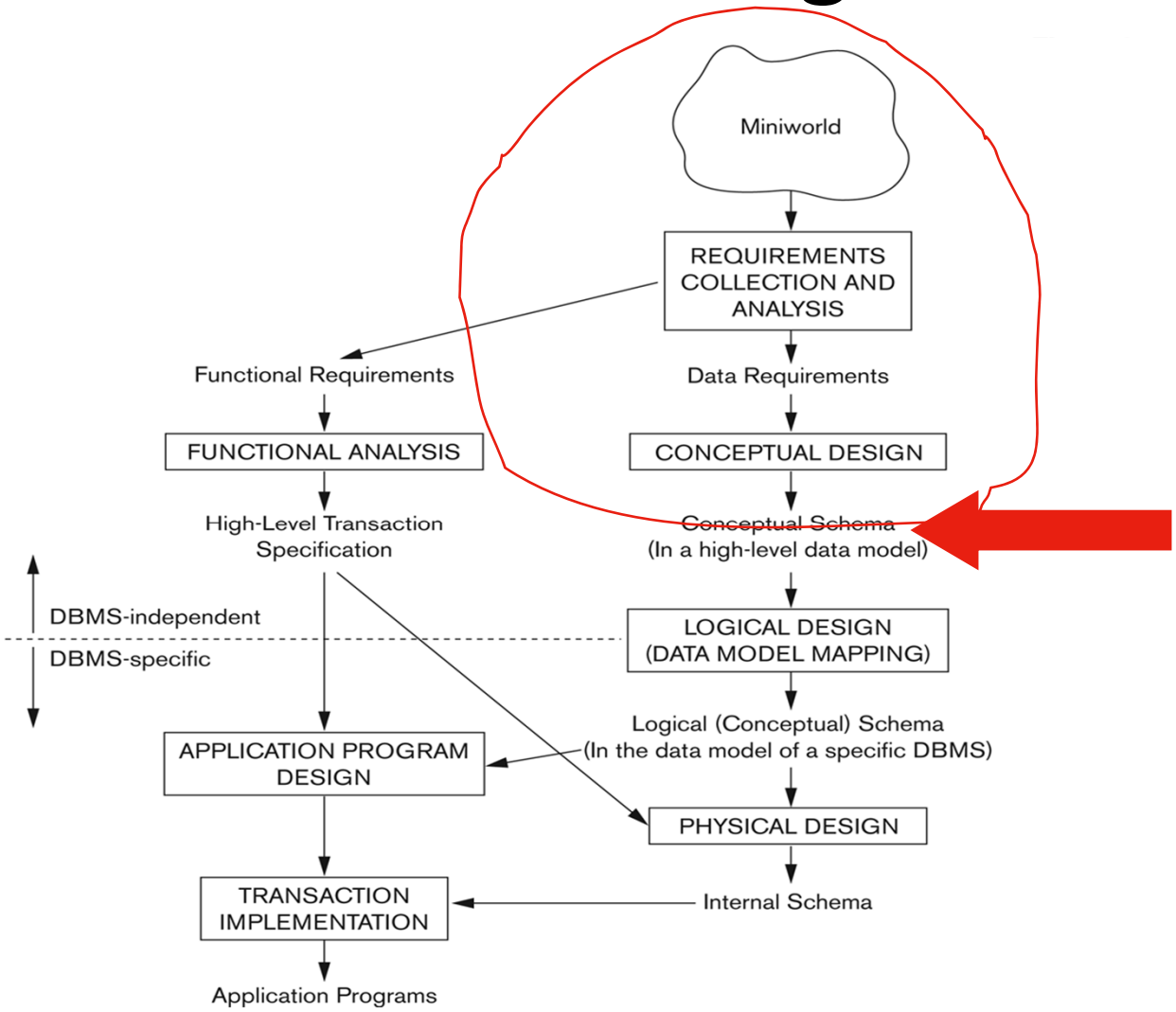
Topics

- Database design
- ER Model
  - 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

Database Design

- Understand the mini-world being modeled
- Specify it using a database design model
  - A few popular ones are:
    - Entity/Relationship (E/R) model
    - UML (Unified Modeling Language)
  - Intuitive and convenient
  - But not necessarily implemented by DBMS
- Translate specification to the data model of DBMS
  - Relational, XML, object-oriented, etc.
- Create DBMS schema

Database Design

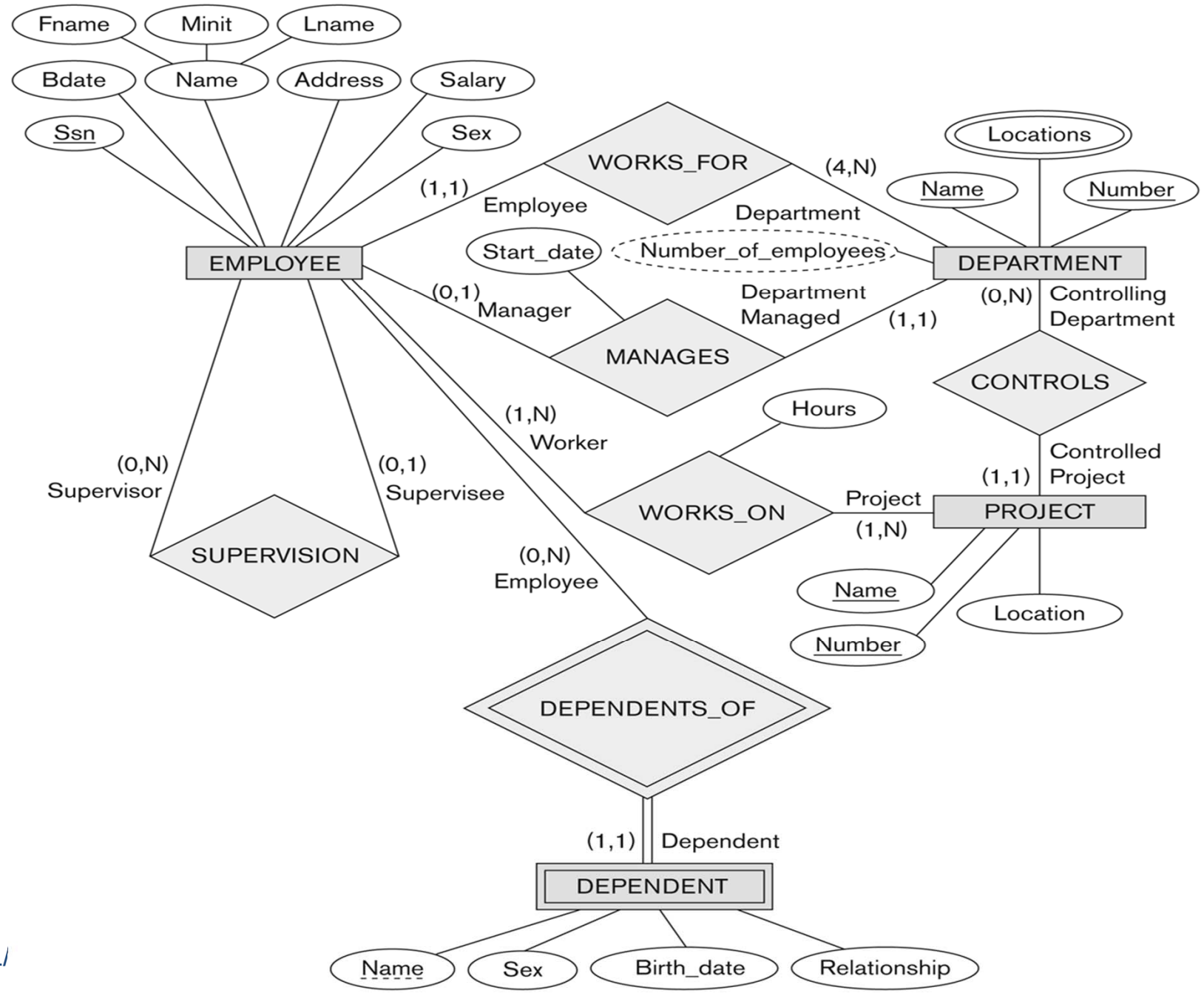


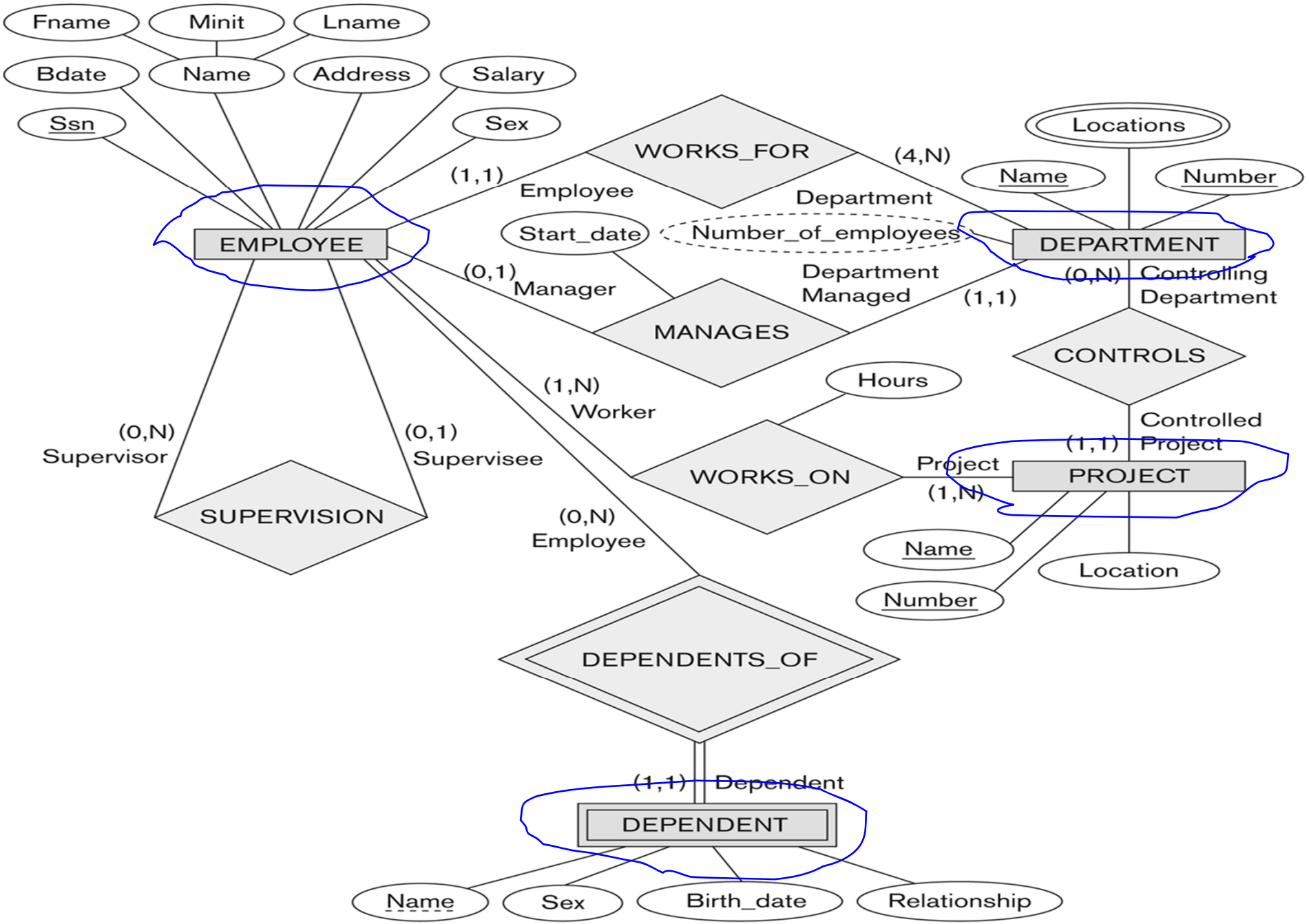
A Database Design Example

- 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.
- Each department controls a number of PROJECTs. Each project has a name, number and is located at a single location.
- We store each EMPLOYEE's social security number, address, salary, sex, 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 employee.

An Database Design Example

- 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.
- Each department *controls* a number of PROJECTs. Each project has a name, number and is located at a single location.
- 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, gender, birthdate, and relationship to employee.


Fname
Minit
Lname
Name
Bdate
Address
Salary
Ssn
Sex
EMPLOYEE
WORKS_FOR
(1,1)
Employee
(4,N)
Department
Locations
Name
Number
DEPARTMENT
Start_date
Number_of_employees
(0,1)
Manager
MANAGES
Department
Managed
(1,1)
(0,N)
Controlling
Department
CONTROLS
Hours
(1,N)
Worker
Controlled
Project
(1,1)
(0,N)
Supervisor
(0,1)
Supervisee
SUPERVISION
WORKS_ON
Project
(1,N)
PROJECT
Name
Location
Number
(0,N)
Employee
DEPENDENTS_OF
(1,1)
Dependent
DEPENDENT
Name
Sex
Birth_date
Relationship


Fname
Minit
Lname
Name
Bdate
Address
Salary
Ssn
Sex
EMPLOYEE
WORKS_FOR
(1,1)
Employee
(4,N)
Department
Locations
Name
Number
DEPARTMENT
Start_date
Number_of_employees
(0,1)
Manager
MANAGES
Department
Managed
(1,1)
(0,N)
Controlling
Department
CONTROLS
Hours
(1,N)
Worker
Controlled
Project
(1,1)
WORKS_ON
Project
(1,N)
PROJECT
(0,N)
Supervisor
(0,1)
Supervisee
SUPERVISION
(0,N)
Employee
Name
Location
Number
DEPENDENTS_OF
(1,1)
Dependent
DEPENDENT
Name
Sex
Birth_date
Relationship

Entity-relationship (E/R) model

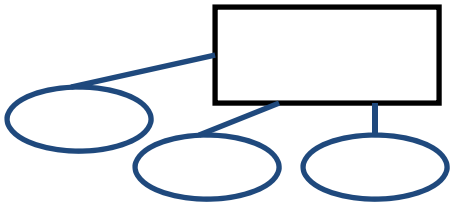
- Historically and still very popular
- Can think of as a “watered-down” object-oriented design model
- Primarily a design model—not directly implemented by DBMS
- Designs represented by E/R diagrams
  - there are other styles
  - Very similar to UML diagrams

Entities and Attributes

- *Entity*: A specific object or "thing" in the mini-world that is represented in the database.

  For example, the EMPLOYEE John Smith,

  the Research DEPARTMENT, the ProductX PROJECT.



- *Attributes*: properties used to describe an entity.

  For example, an EMPLOYEE entity may have a 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'

Types of Attributes

- Simple vs. Composite Attributes
  - *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, Name (FirstName, MiddleName, LastName).

Types of Attributes (cont.)

- Single-valued vs. Multi-valued.
  - *Single-valued*: an entity may have at most one value for the attribute
  - *Multi-valued*: An entity may have multiple values for that attribute. For example, PreviousDegrees of a STUDENT. {PreviousDegrees}.
- *NULL* values
  - What if the student does not hold a previous degree?
  - What if the student has a previous degree but the information is not provided?
  - Apartment number in an address

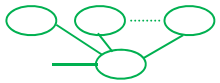
Types of Attributes (cont.)

- Stored vs. derived
  - Number of credit hours a student took in a semester
  - GPA of a student in a semester

Key Attributes

- Entities with the same basic attributes are grouped or typed into an entity type.
  - For example, the EMPLOYEE entity type or the PROJECT entity type.
- 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.
  - An entity type may have more than one key.

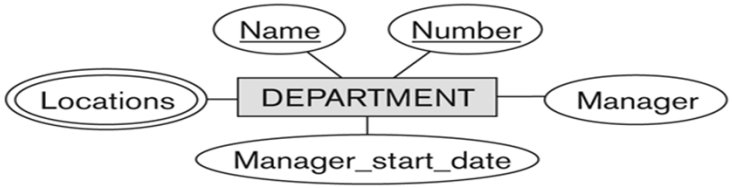
SUMMARY OF ER-DIAGRAM NOTATION

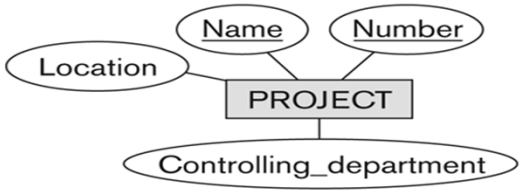
Symbol	Meaning
	ENTITY TYPE
	ATTRIBUTE
	KEY ATTRIBUTE
	MULTIVALUED ATTRIBUTE
	COMPOSITE ATTRIBUTE
	DERIVED ATTRIBUTE

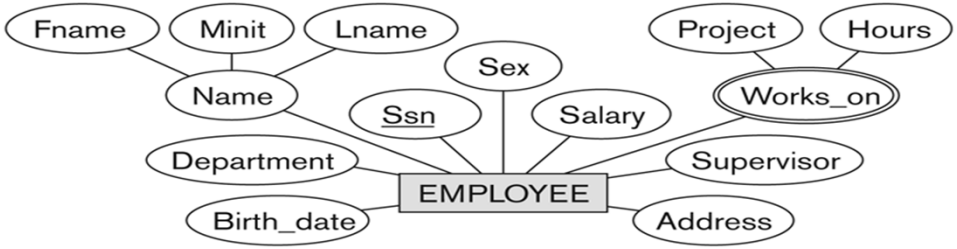
A Database Design Example

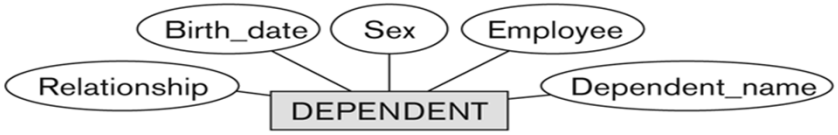
- The company is organized into DEPARTMENTs. Each department has a name, number and an employee who *manages* the department. We keep track of the start date of the department manager.
- Each department *controls* a number of PROJECTs. Each project has a name, number and is located at a single location.
- We store each EMPLOYEE's social security number, address, salary, sex, 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 employee.

Summary (cont.)



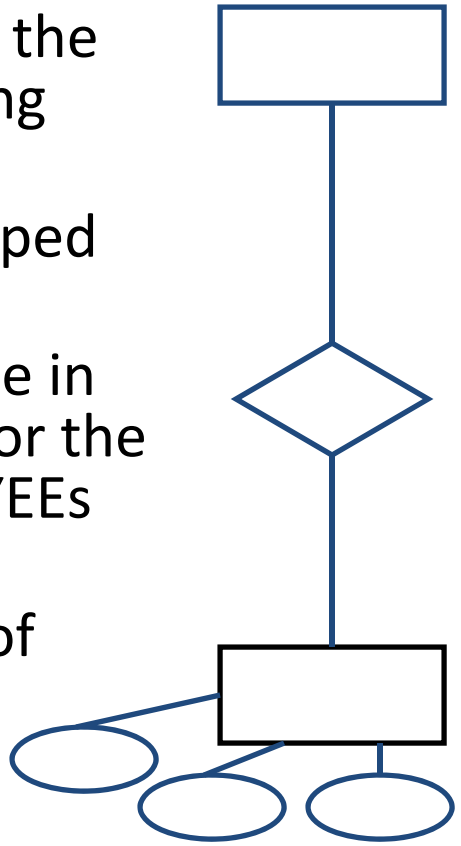




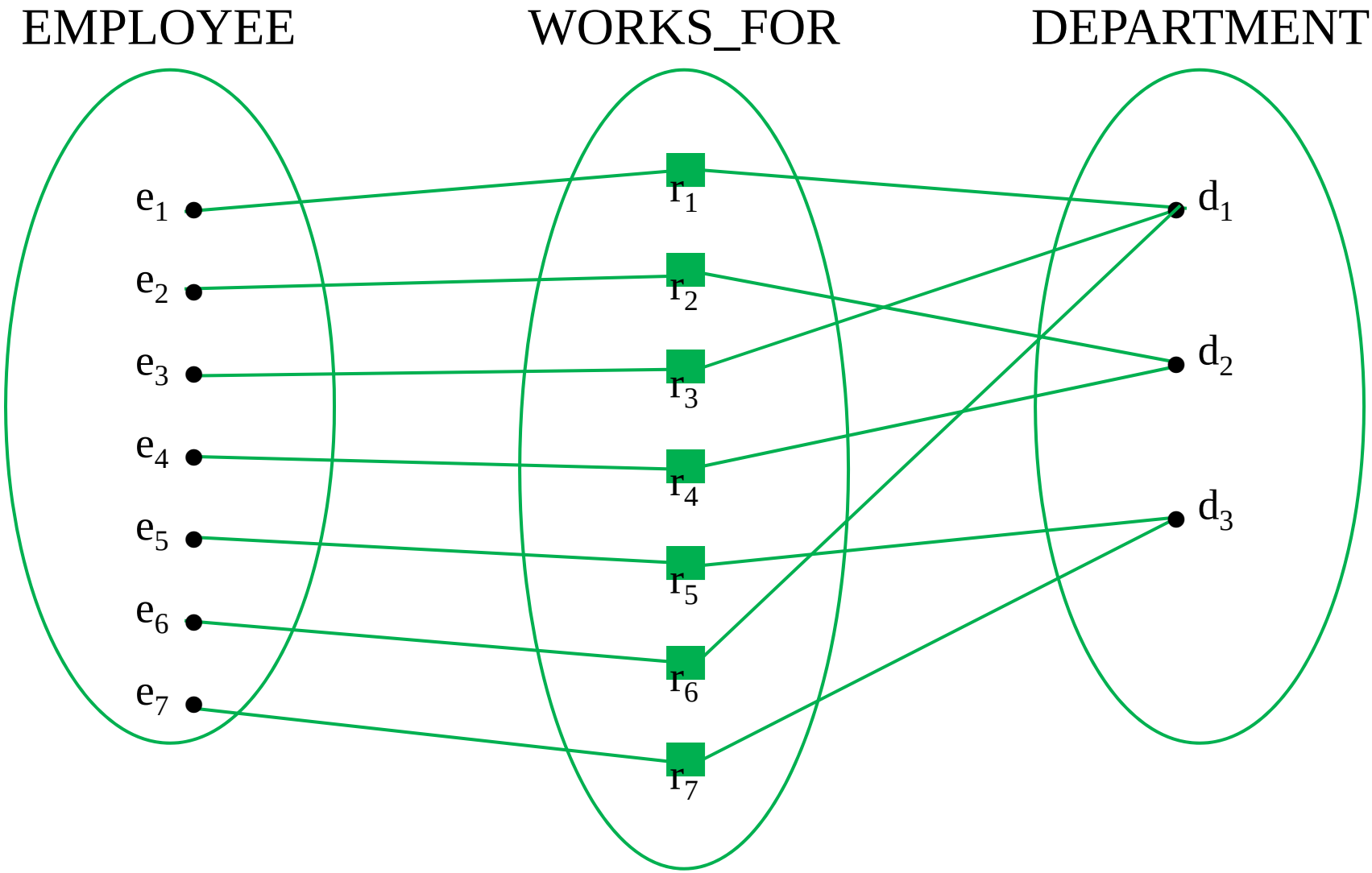


Relationships

- 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.



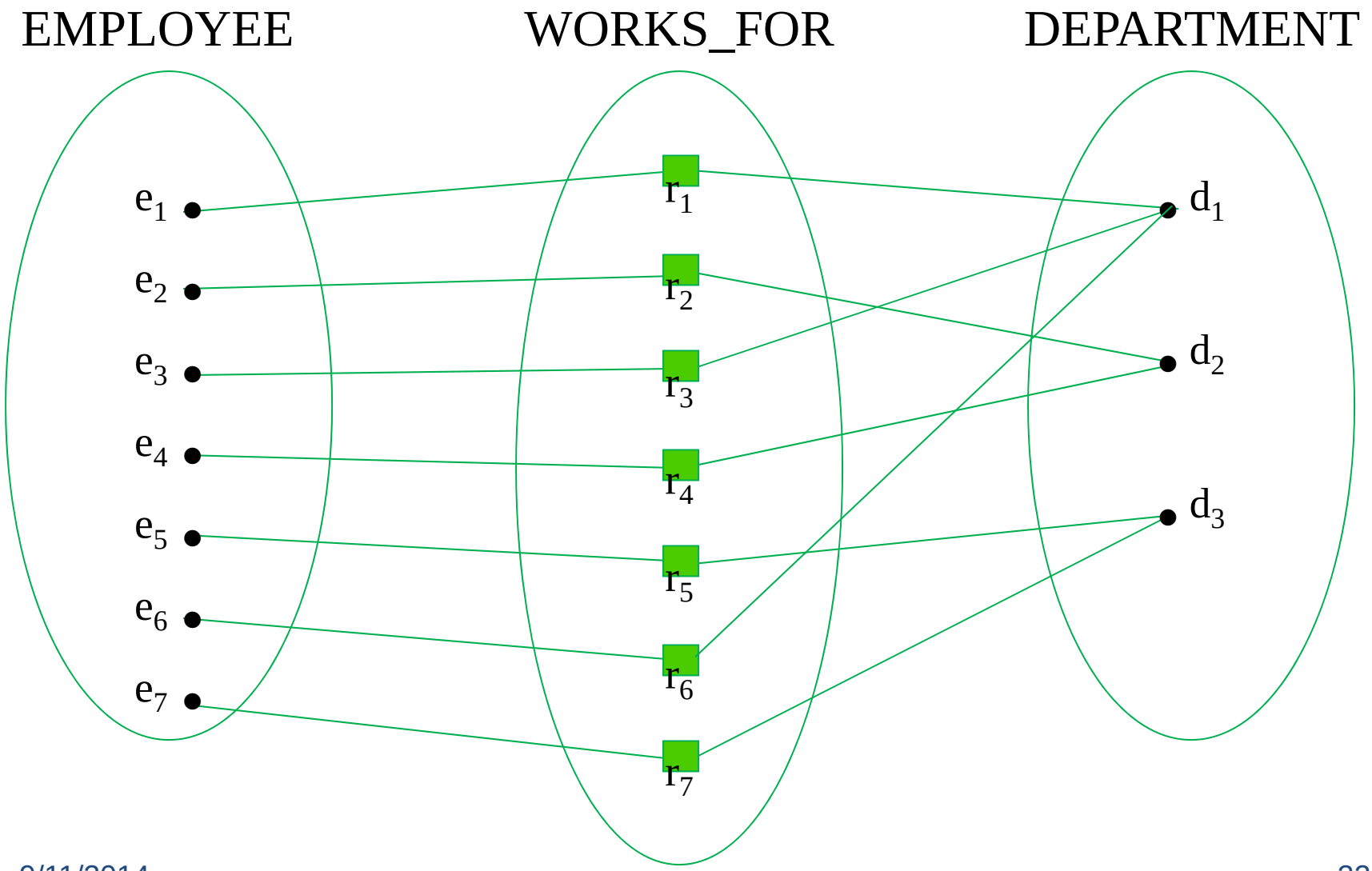
Instances of a relationship



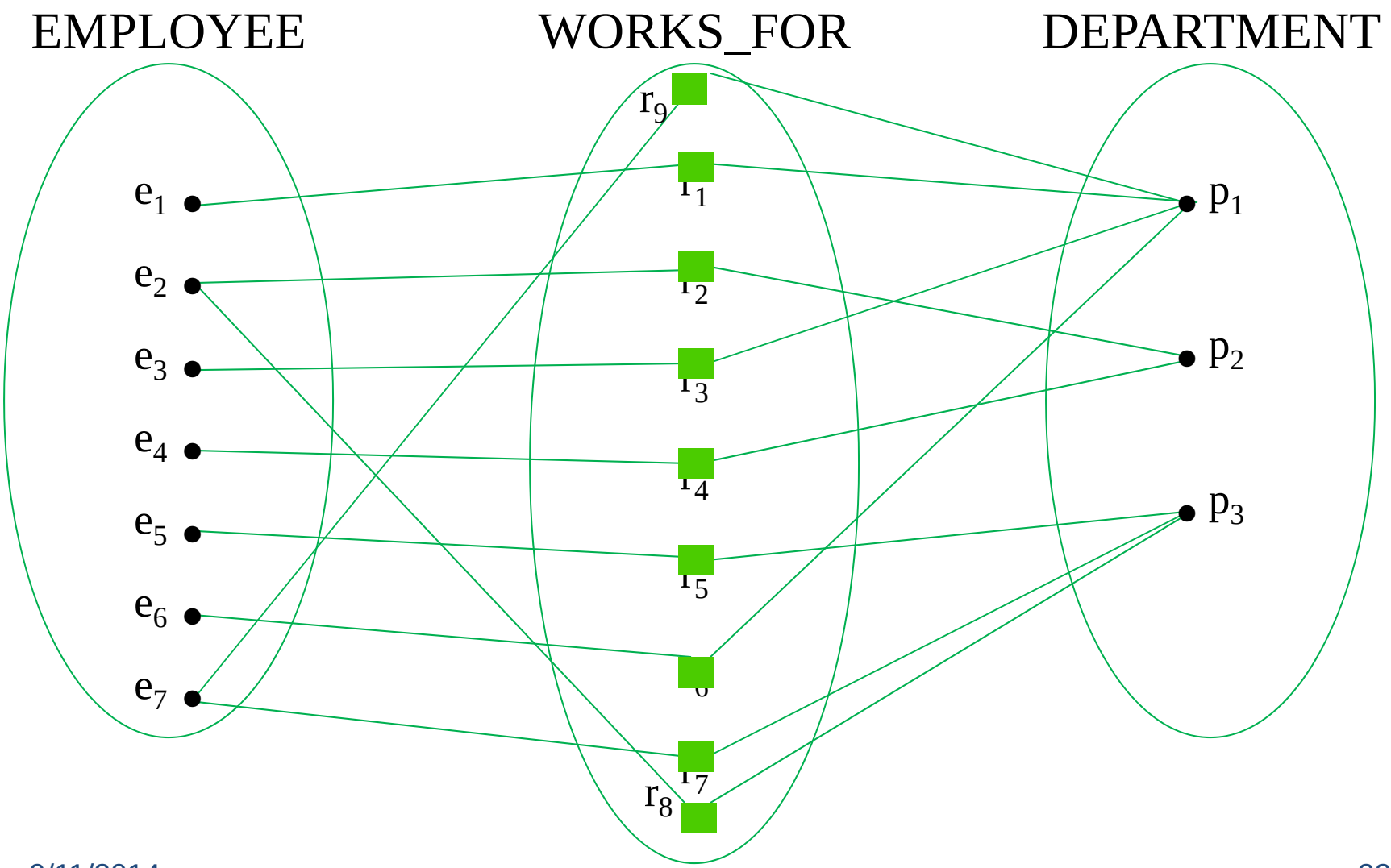
Structural Constraints (I)

- Maximum Cardinality
  - One-to-one (1:1)
  - One-to-many (1:N) or Many-to-one (N:1)
  - Many-to-many

Many-to-one (N:1) RELATIONSHIP

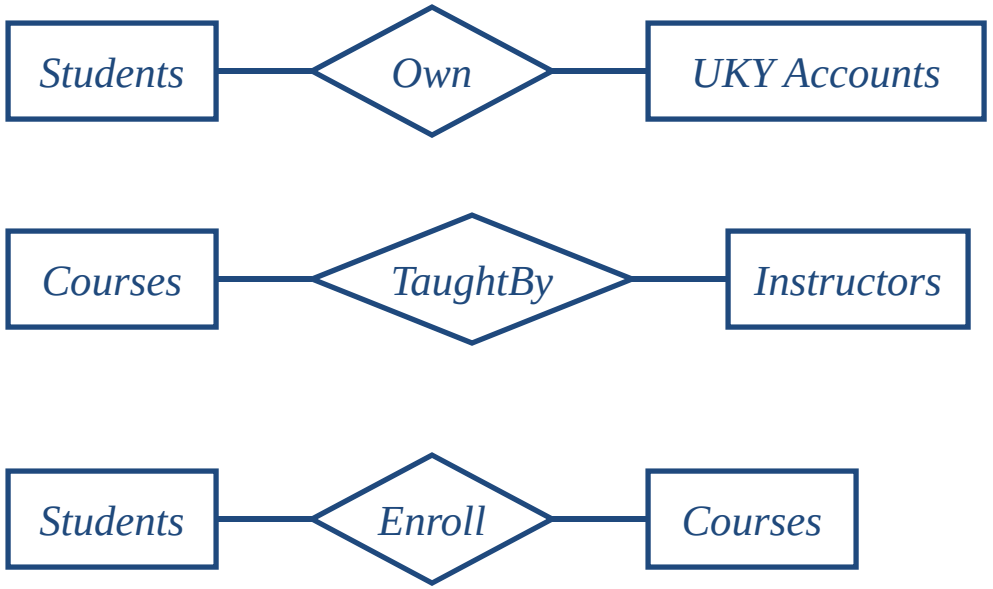


Many-to-many (M:N) RELATIONSHIP



More Examples

- Each student may have exactly one account.
- Each faculty may teach many courses
- Each student may enroll many courses



Structural Constraints (II)

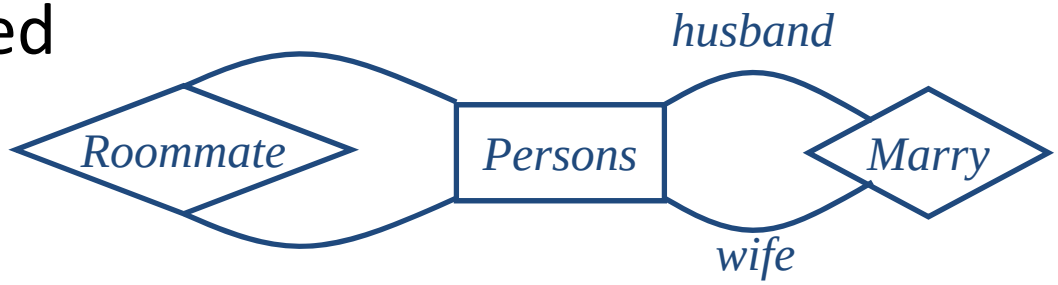
- Minimum Cardinality (also called participation constraint or existence dependency constraints)
  - Zero (partial participation)
  - One or more (total participation)

Roles in relationships

- An entity set may participate more than once in a relationship set

☞ May need to label edges to distinguish roles

- Examples
  - People are married as husband and wife; label needed
  - People are roommates of each other; label not needed

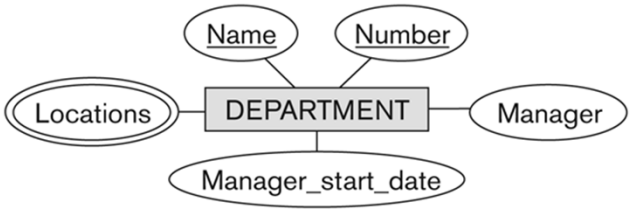


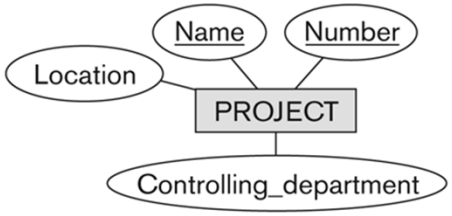
Recursive relationship

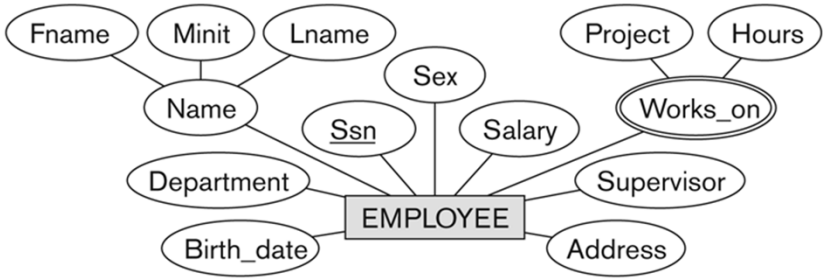
- We can also have a *recursive relationship type*.
- Both participations are same entity type in different roles.
- For example, SUPERVISION relationships between EMPLOYEE (in role of supervisor or boss) and (another) EMPLOYEE (in role of subordinate or worker).
- In ER diagram, need to display role names to distinguish participations.

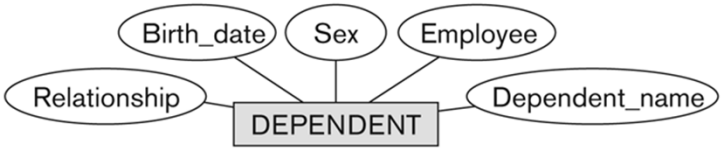
An Database Design Example

- 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.
- Each department *controls* a number of PROJECTs. Each project has a name, number and is located at a single location.
- We store each EMPLOYEE's social security number, address, salary, sex, 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 employee.


Name
Number
Locations
DEPARTMENT
Manager
Manager_start_date


Name
Number
Location
PROJECT
Controlling_department


Fname
Minit
Lname
Name
Sex
Ssn
Salary
Project
Hours
Works_on
Department
Supervisor
EMPLOYEE
Birth_date
Address


Birth_date
Sex
Employee
Relationship
DEPENDENT
Dependent_name

Weak Entity Types

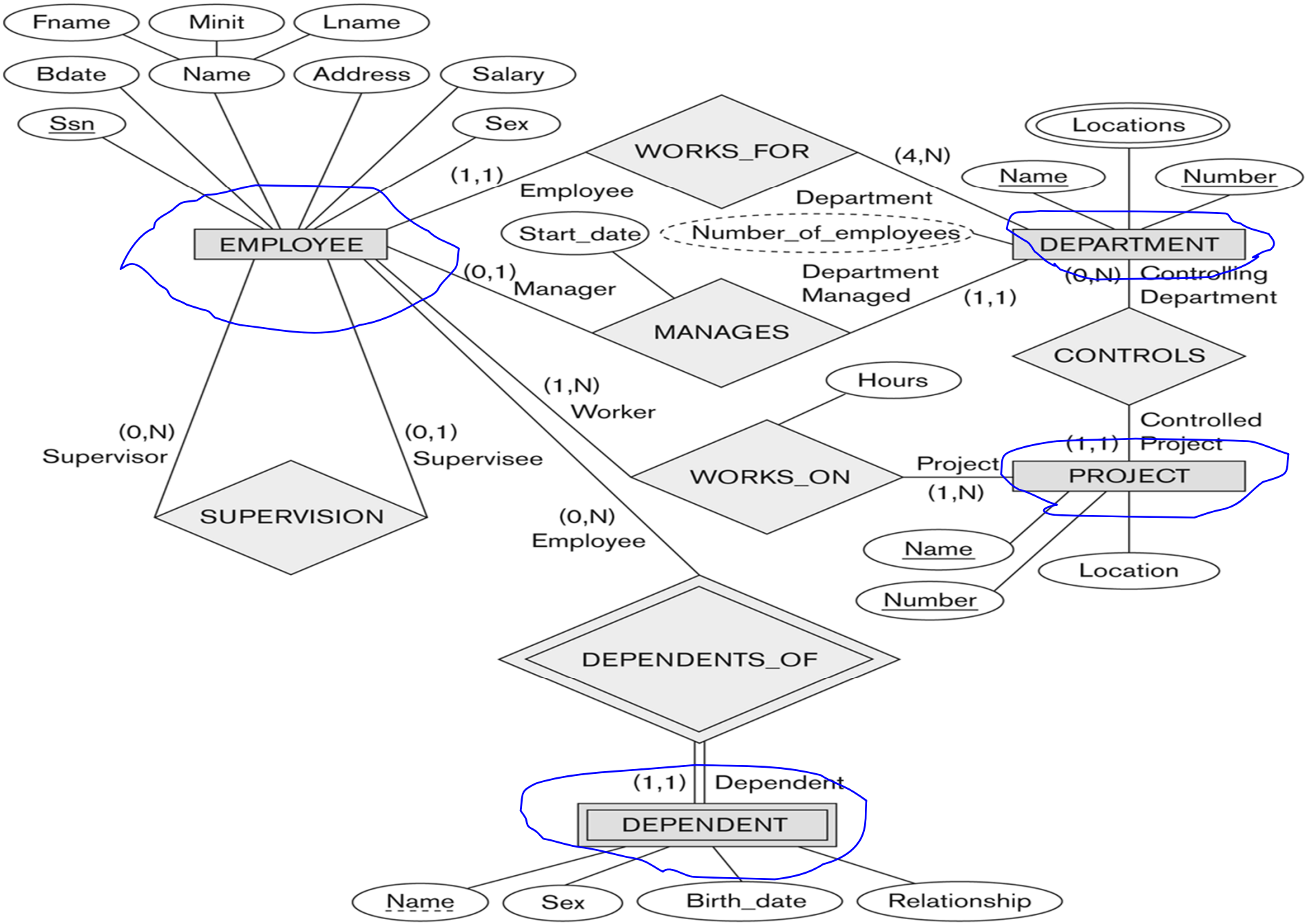
- A weak entity is an entity that does not have a key attribute
- A weak entity must participate in an identifying relationship type with an owner or identifying entity type
- Entities are identified by the combination of:
  - A partial key of the weak entity type
  - The particular entity they are related to in the identifying entity type

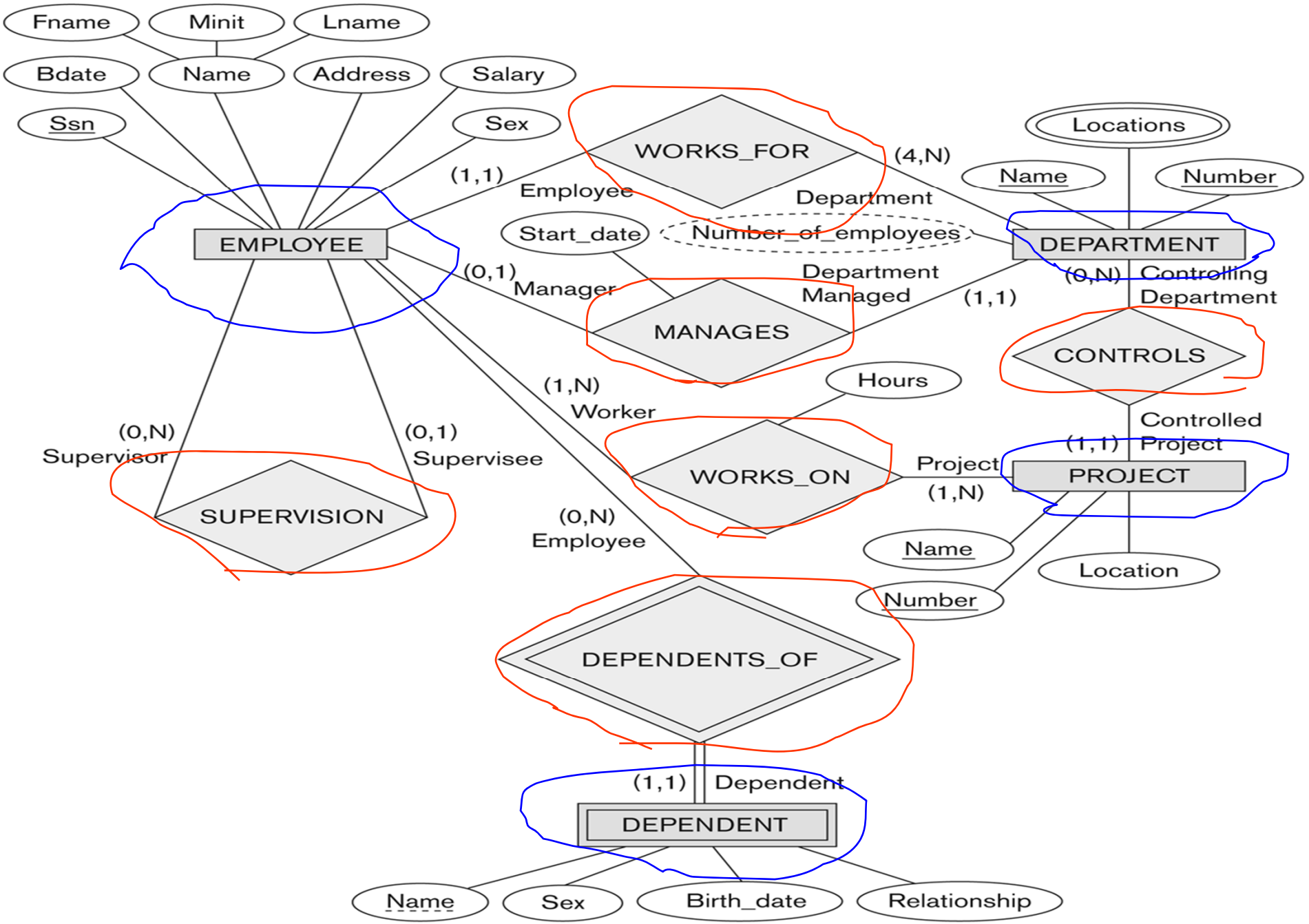
Example:

Suppose that a DEPENDENT entity is identified by the dependent's first name and birthdate, *and* the specific EMPLOYEE that the dependent is related to. DEPENDENT is a weak entity type with EMPLOYEE as its identifying entity type via the identifying relationship type DEPENDENT_OF

Weak Entity-Set Rules

- A weak entity set has one or more many-one relationships to other (supporting) entity sets.
  - Not every many-one relationship from a weak entity set need be supporting.
- The key for a weak entity set is its own underlined attributes and the keys for the supporting entity sets.
  - E.g., player-*number* and team-*name* is a key for *Players* in the previous example.


Fname
Minit
Lname
Name
Bdate
Address
Salary
Ssn
Sex
EMPLOYEE
WORKS_FOR
(1,1)
Employee
(4,N)
Department
Locations
Name
Number
DEPARTMENT
Start_date
Number_of_employees
(0,1)
Manager
MANAGES
Department
Managed
(1,1)
(0,N)
Controlling
Department
CONTROLS
Hours
(1,N)
Worker
Controlled
Project
(1,1)
(0,N)
Supervisor
(0,1)
Supervisee
SUPERVISION
WORKS_ON
Project
(1,N)
PROJECT
(0,N)
Employee
Name
Location
Number
DEPENDENTS_OF
(1,1)
Dependent
DEPENDENT
Name
Sex
Birth_date
Relationship


Fname
Minit
Lname
Name
Bdate
Address
Salary
Ssn
Sex
EMPLOYEE
WORKS_FOR
(1,1)
Employee
(4,N)
Department
Locations
Name
Number
Start_date
Number_of_employees
DEPARTMENT
(0,1)
Manager
Department
Managed
MANAGES
(1,1)
(0,N)
Controlling
Department
CONTROLS
Hours
(1,N)
Worker
Controlled
Project
(1,1)
(0,N)
Supervisor
(0,1)
Supervisee
WORKS_ON
Project
(1,N)
PROJECT
SUPERVISION
(0,N)
Employee
Name
Location
Number
DEPENDENTS_OF
(1,1)
Dependent
DEPENDENT
Name
Sex
Birth_date
Relationship