

IE301
Analysis and Design of Data Systems

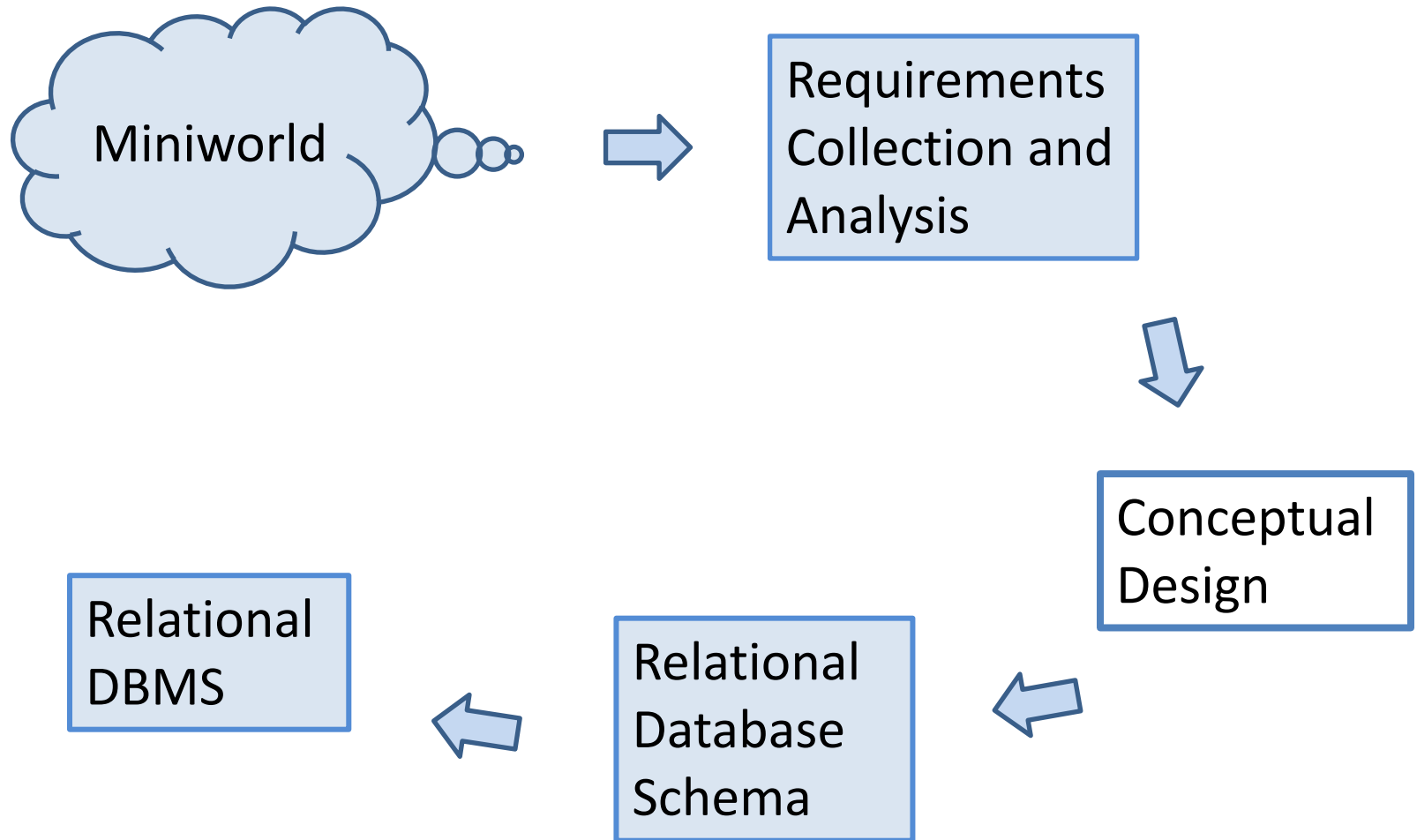
Lecture 8

Entity Relationship Model

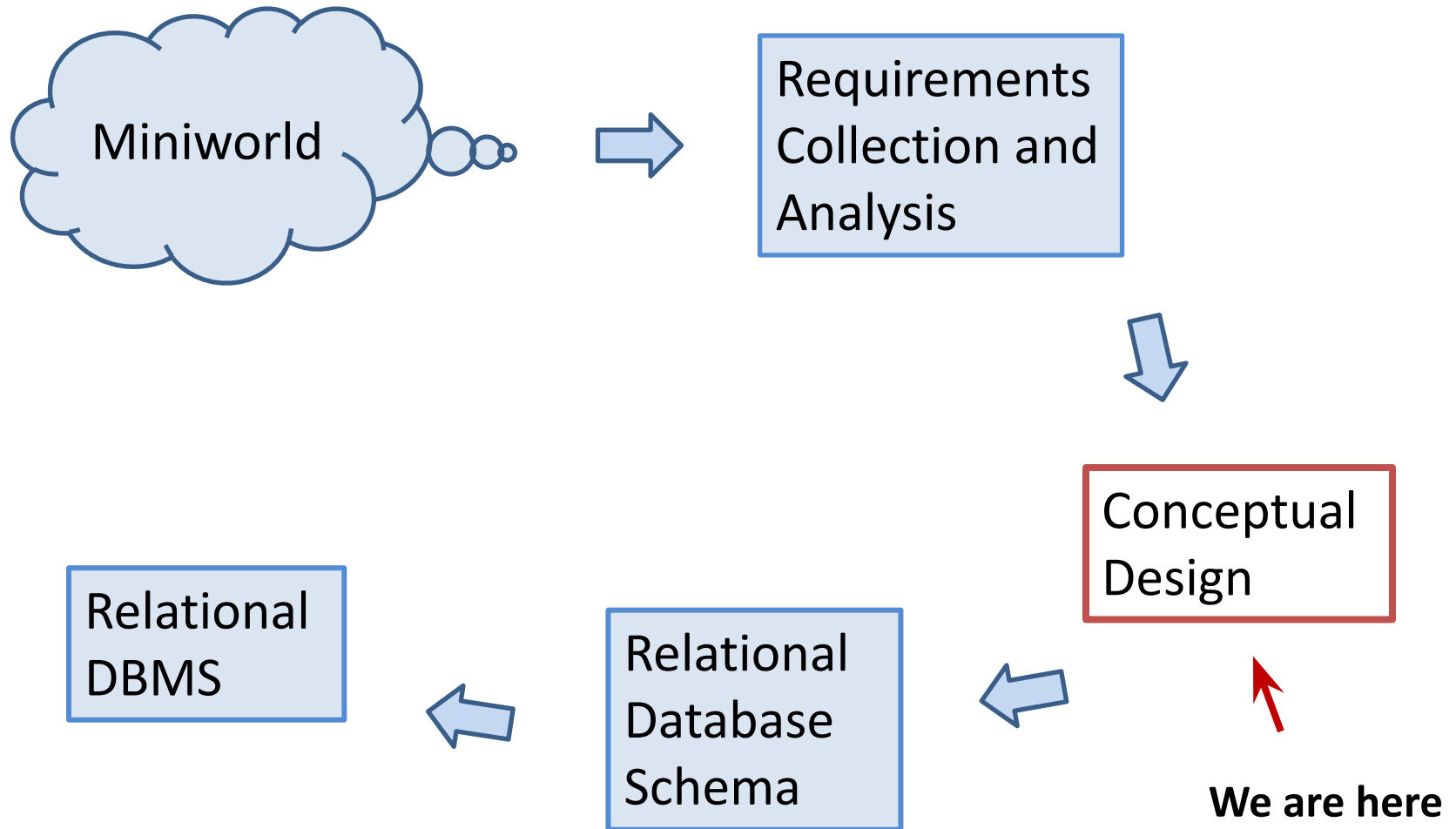
Aram Keryan

September 23, 2015

Phases of Database Design



Phases of Database Design



Requirements Collection and Analysis

- During this stage database designers interview prospective database users to understand and document their data requirements

Example of data requirements for Company database:

- Company consists of five departments
- Department is identified by department number and name
- Employee can work for only one department
- Each employee may have a supervisor who is also an employee
- Department controls a number of projects
- Employee can work on several projects
- Employees are payed on hourly basis
- And so on . . .

Conceptual Database Design

- Conceptual database design involves modelling the collected information at a high-level of abstraction without using a particular data model or DBMS.

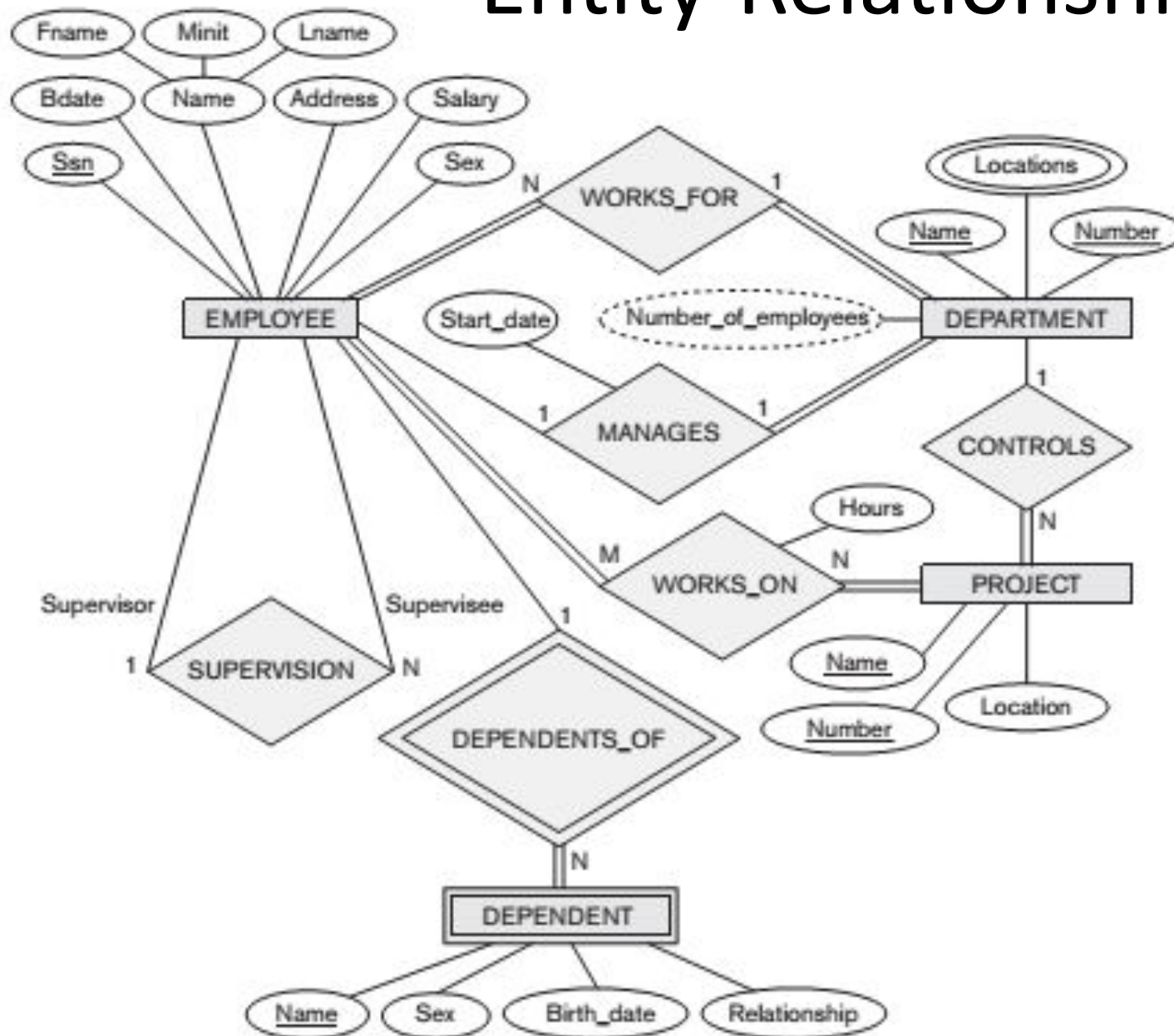
Reasons for Conceptual Modelling:

- Independent of DBMS
- Allows for easy communication between end-users and developers
- Has a clear method to convert from high-level model to relational model
- Conceptual schema is a permanent description of the database requirements

Conceptual Database Schema


- As a result of using High-Level Conceptual Data Model a Conceptual Database Schema is created
- Conceptual Schema includes detailed description of the entity types, relationships, and constraints
- Conceptual Schema reflects all the data requirements collected during the initial stage

Entity-Relationship Model



Entity-Relationship Model

- Entity-Relationship diagram models data as ***entities***, ***attributes*** and ***relationships***

Entity 
is a *thing* about which we store data
is a *thing* in the real world with independent existence
is a *thing* which can be distinctly identified

✓ e.g. a person, a bank account, a building

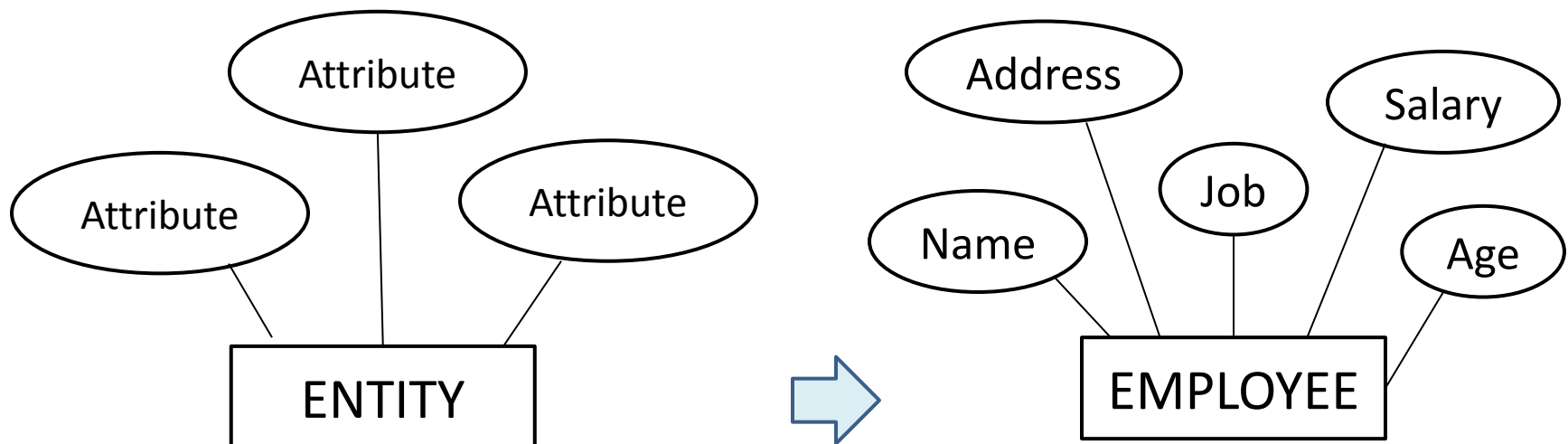
- Entity is a basic object that ER model represents

Examples of Entities Types

- Examples of a person entity would be EMPLOYEE, DOCTOR, or STUDENT
- Examples of a place entity would be STATE or COUNTRY
- Examples of an object entity would be BUILDING, AUTO, or PRODUCT
- An example of an event entity would be SALES, RETURNS, or REGISTRATION
- An example of a concept entity would be ACCOUNT or DEPARTMENT, UNIVERSITY COURSE

Attributes

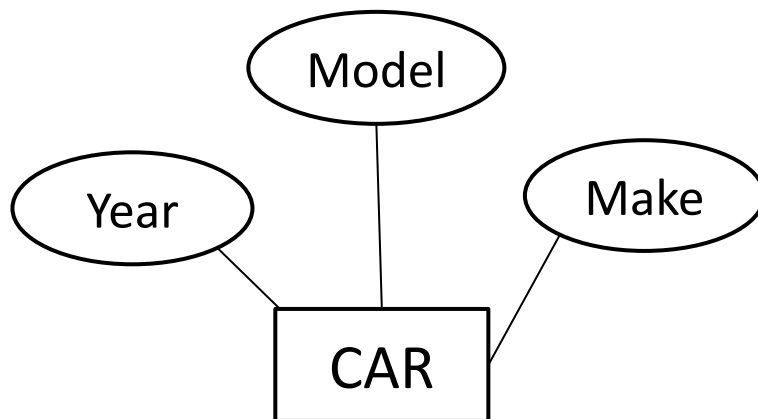
- Each entity type has ***attributes*** — the particular properties that describe it
- ✓ e.g., EMPLOYEE entity type may be described by the employee's name, age, address, salary, and job



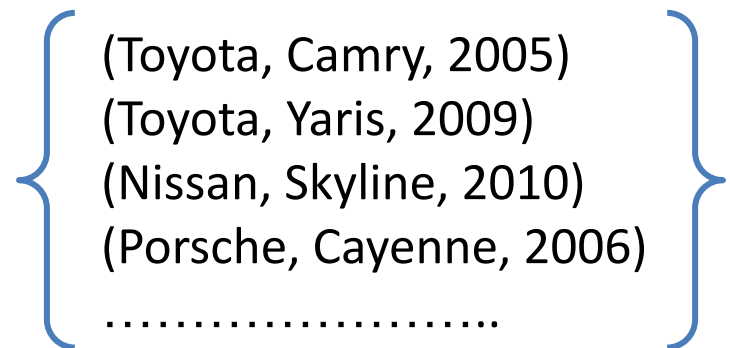
Entity Types and Entity Sets

- An **entity type** defines a *collection* (or *set*) of entities that have the same attributes.
 - ✓ Each entity type is described by its name and attributes
- An **entity set** is the collection of all entities of a particular entity type in the database at any point in time
- Entity sets usually have the same name as entity types

Entity Type



Entity Set

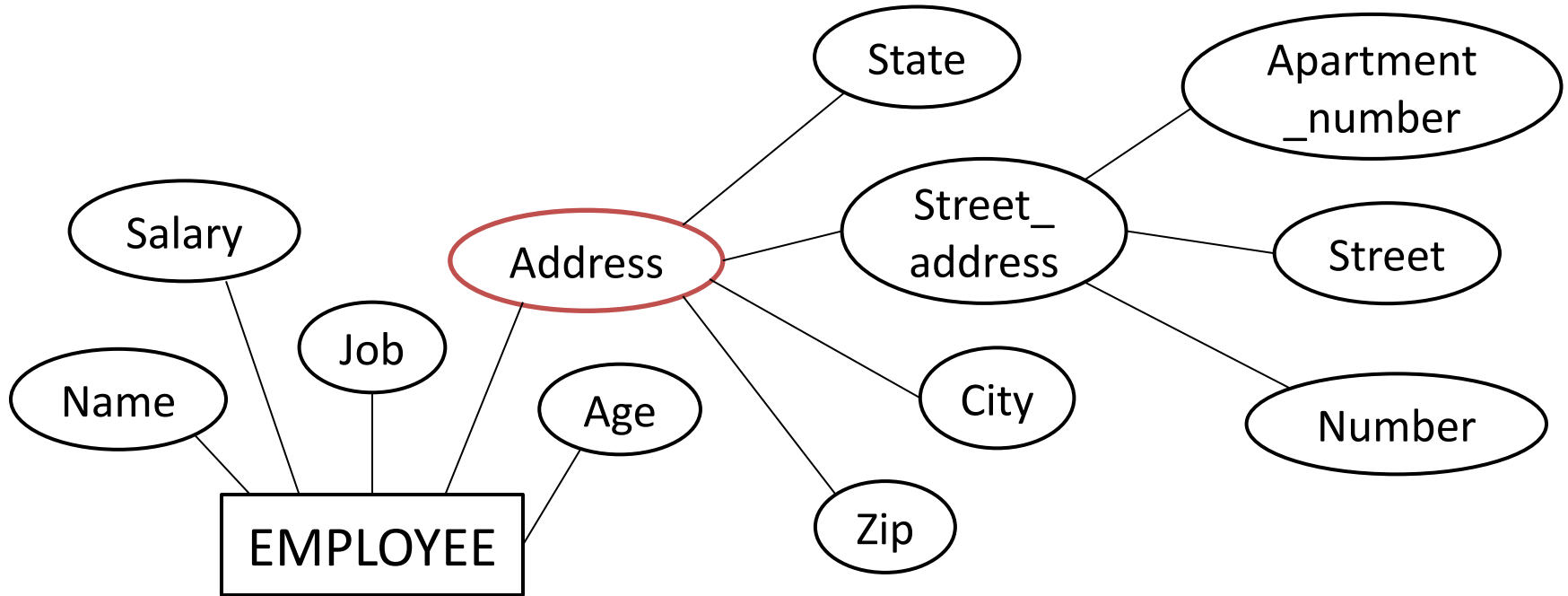


Types of Attributes

Several types of attributes occur in the ER model:

- *simple (atomic) versus composite*
- *Single-valued versus multi-valued*
- *stored versus derived*

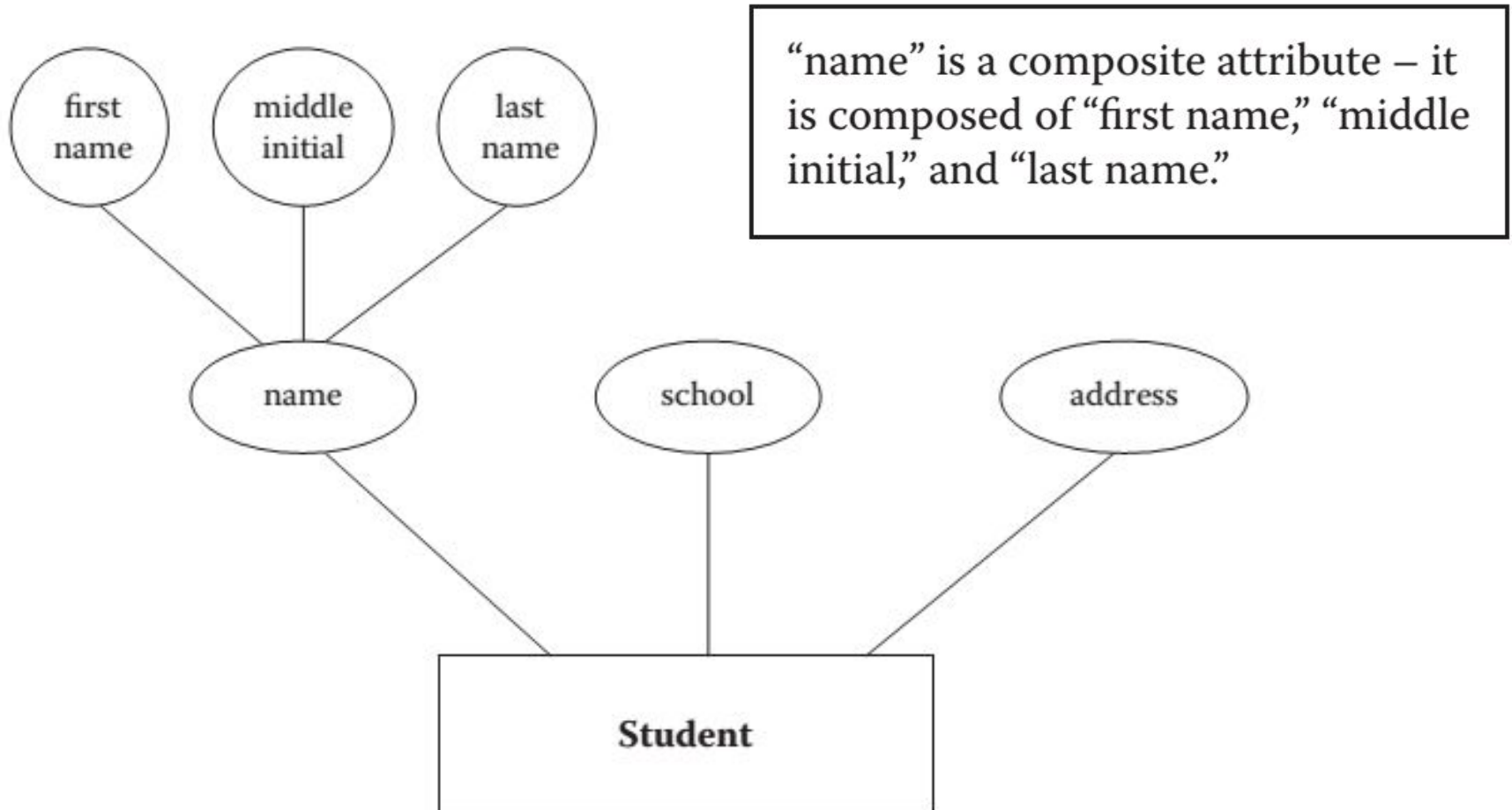
Simple vs. Composite Attributes



“Address” is a **composite** attribute – it can be divided into smaller subparts representing more basic attributes with independent meaning.

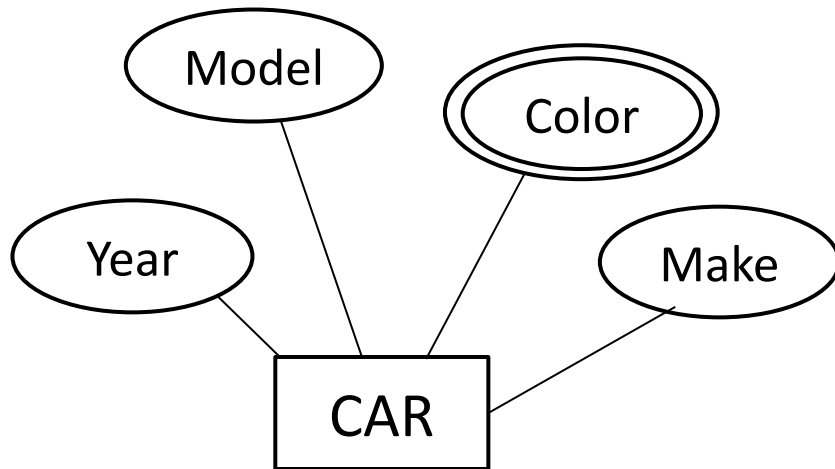
Simple or **atomic** attributes are not divisible.

Another example of a composite attribute

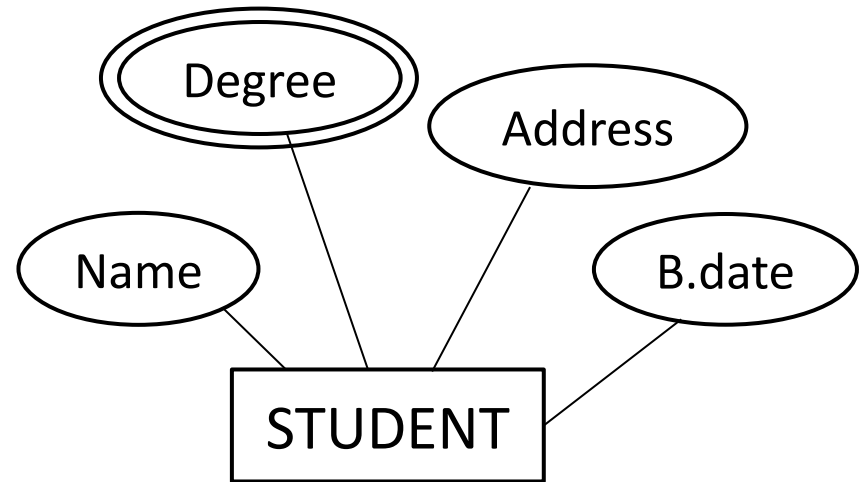


Single-valued vs. multivalued attributes

- An attribute can have a set of values for the same entity

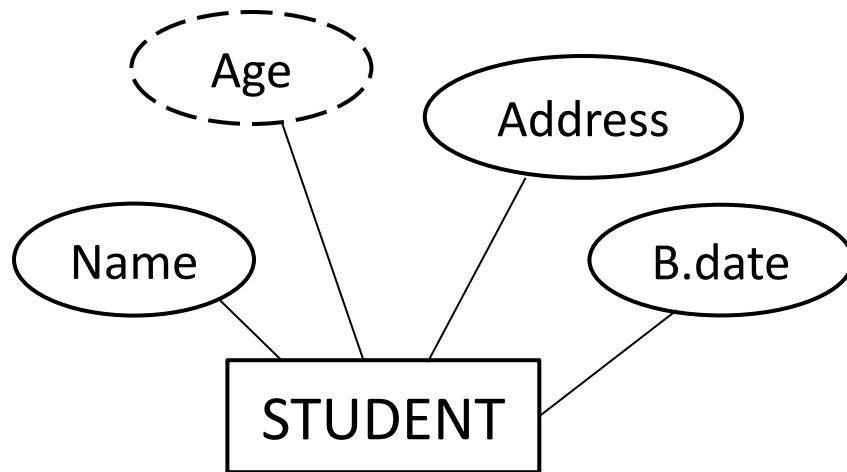


Car can be of one color or can be of multi-color



A student can have one or several degrees

Stored vs. Derived Attributes



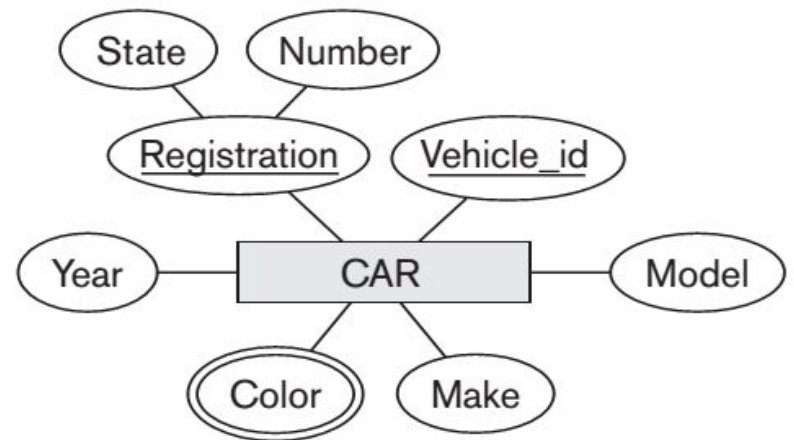
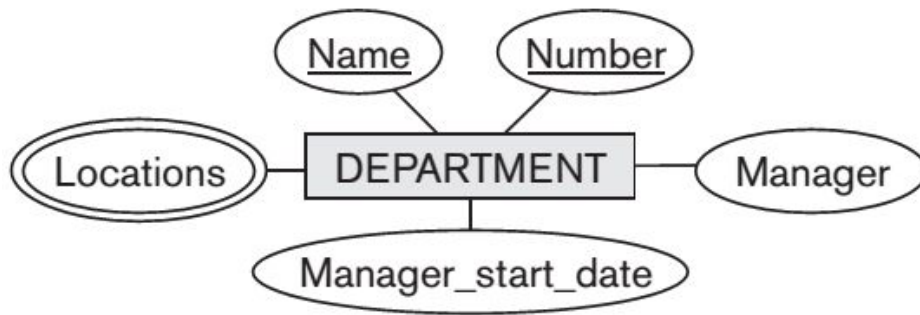
In some cases, two (or more) attribute values are related

“Age” and “B.date” are related since for a particular student his/her age can be determined from the current date and his/her birth date.

- “Age” is called a **derived** attribute
- “B.date” is called a **stored** attribute

Key Attributes

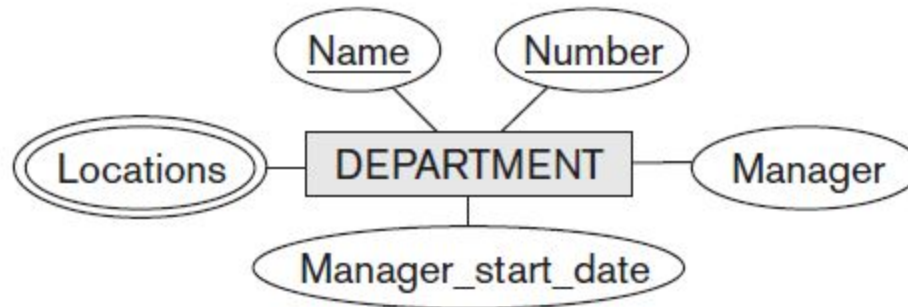
- Entity types usually have one or more attributes whose values are distinct for each individual entity in the entity set
- Such an attribute is called a **key attribute**, and its values can be used to identify each entity uniquely



Example: Requirements Collection and Analysis

“COMPANY”

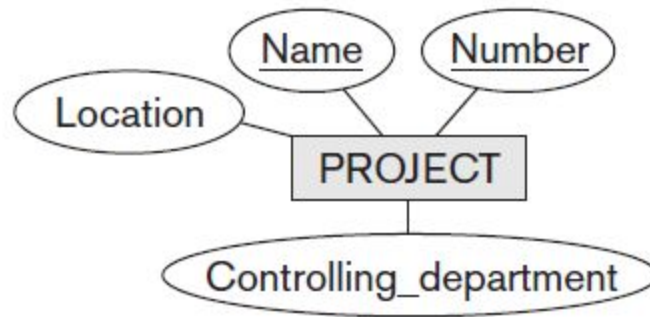
An entity type **DEPARTMENT** with attributes Name, Number, Locations, Manager, and Manager_start_date. Locations is the only multivalued attribute. We can specify that both Name and Number are (separate) key attributes because each was specified to be unique.



Example: Requirements Collection and Analysis

“COMPANY”

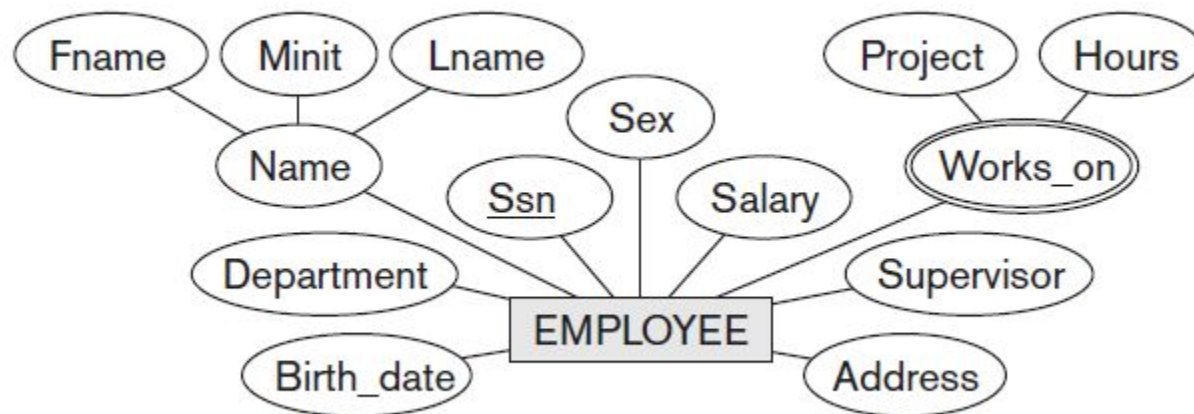
An entity type **PROJECT** with attributes Name, Number, Location, and Controlling_department. Both Name and Number are (separate) key attributes.



Example: Requirements Collection and Analysis

“COMPANY”

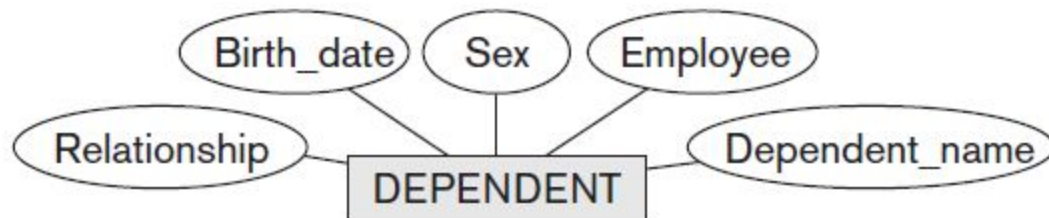
An entity type **EMPLOYEE** with attributes Name, Ssn, Sex, Address, Salary, Birth_date, Department, and Supervisor. Both Name and Address may be composite attributes; however, this was not specified in the requirements. We must go back to the users to see if any of them will refer to the individual components of Name—First_name, Middle_initial, Last_name—or of Address. In our example, Name is modeled as a composite attribute, whereas Address is not, presumably after consultation with the users.



Example: Requirements Collection and Analysis

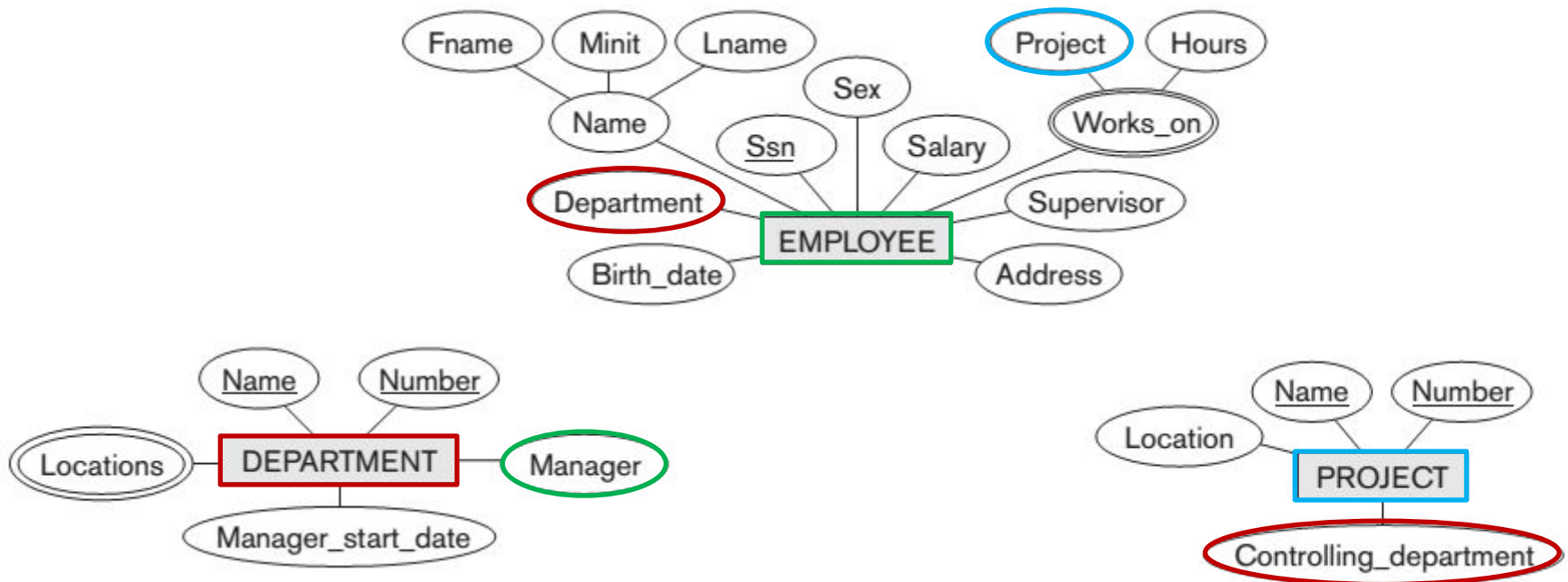
“COMPANY”

An entity type **DEPENDENT** with attributes Employee, Dependent_name, Sex, Birth_date, and Relationship (to the employee).

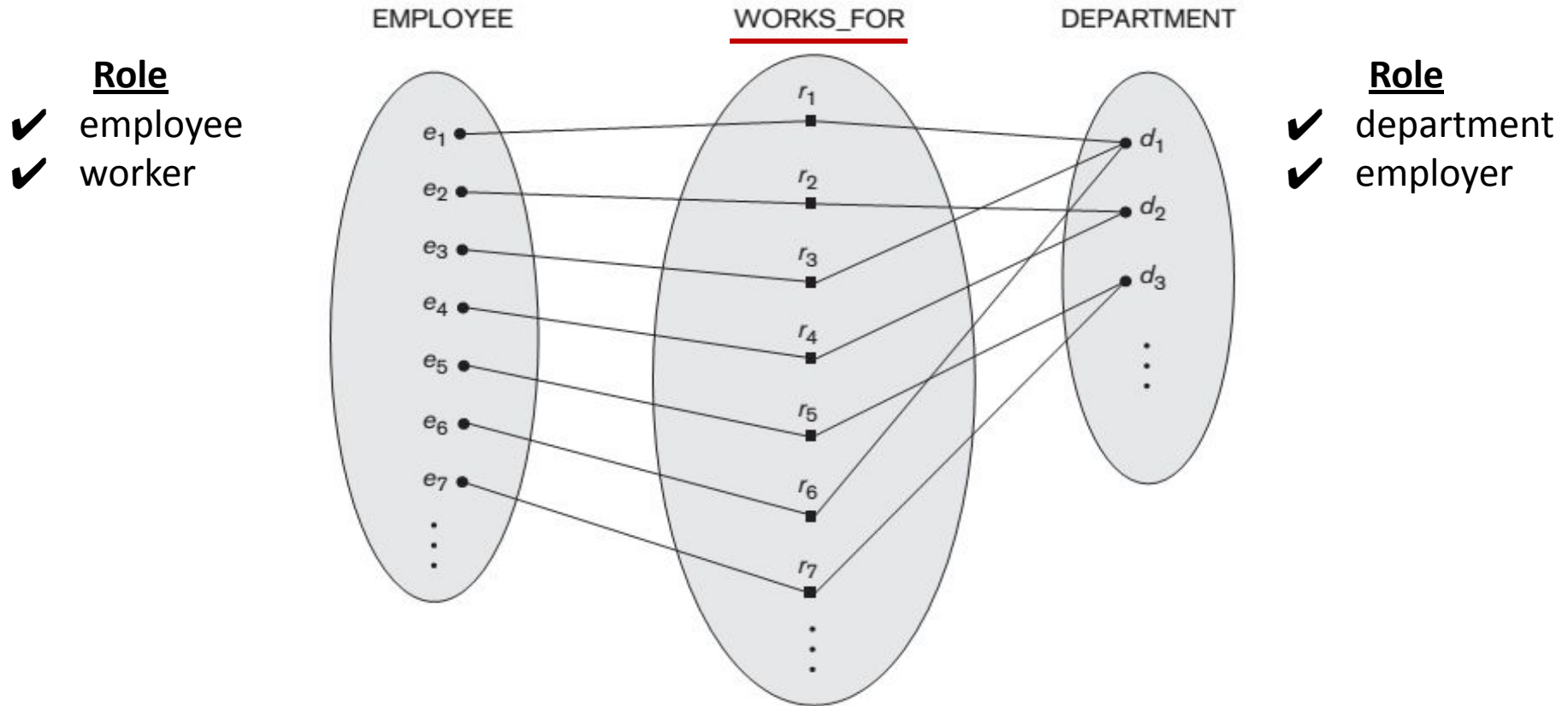


Identifying Relationships

- Whenever an attribute of one entity type refers to another entity type, some relationship exists.

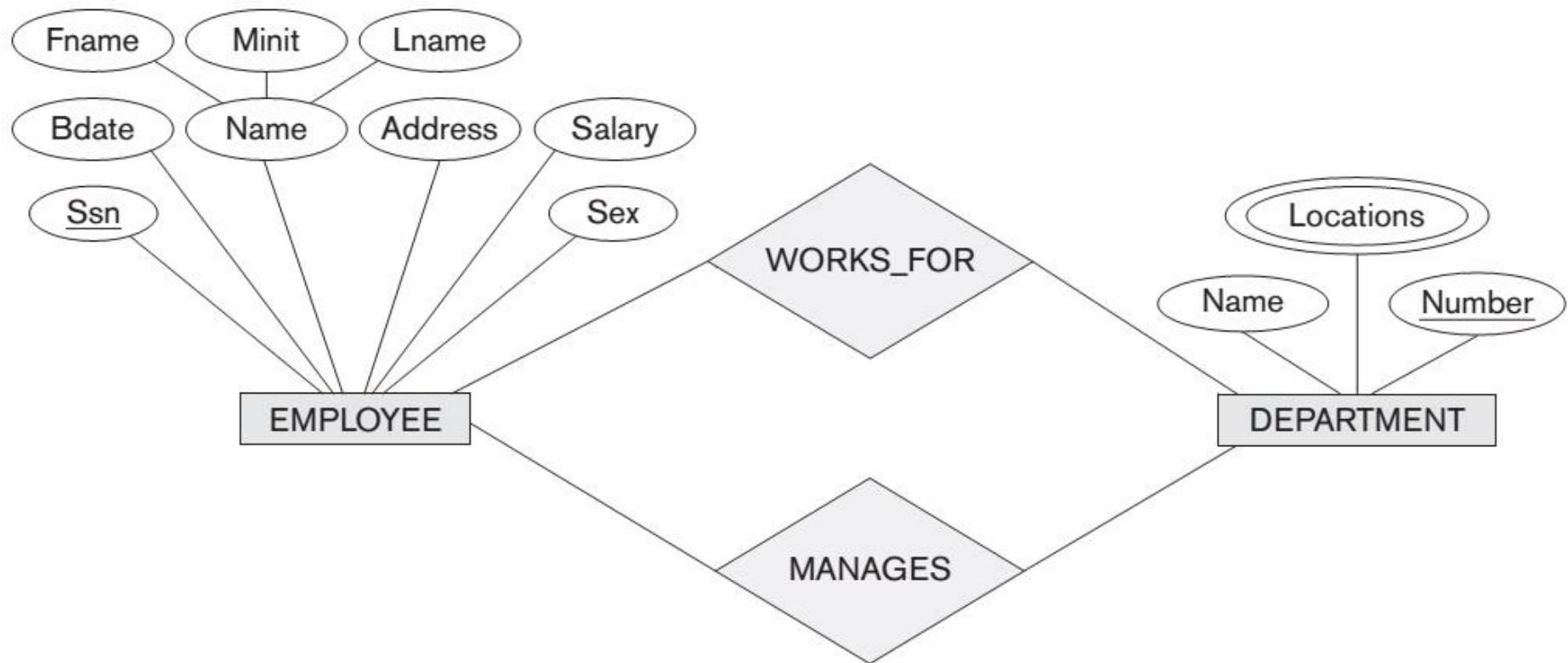


Understanding Relationships



Role name signifies the role that a participating entity from the entity type plays in each relationship instance, and helps to explain what the relationship means.

Representation in ER Schema



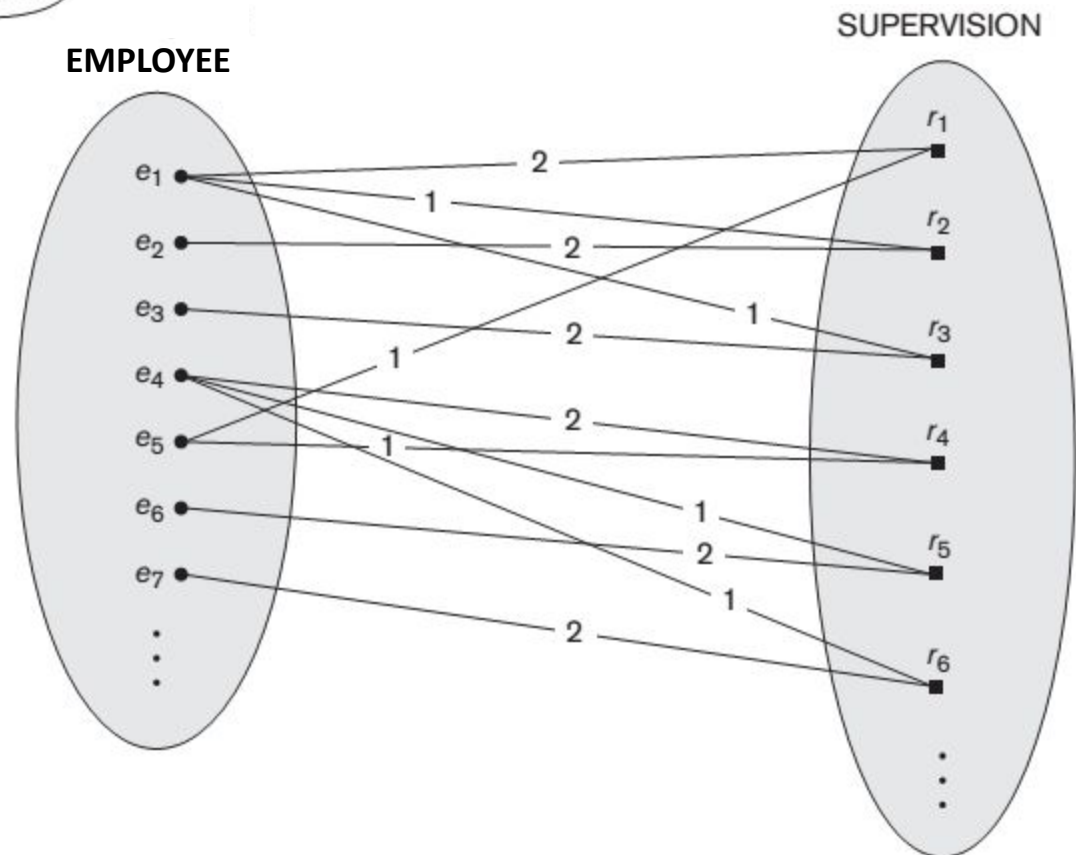
Recursive Relationships



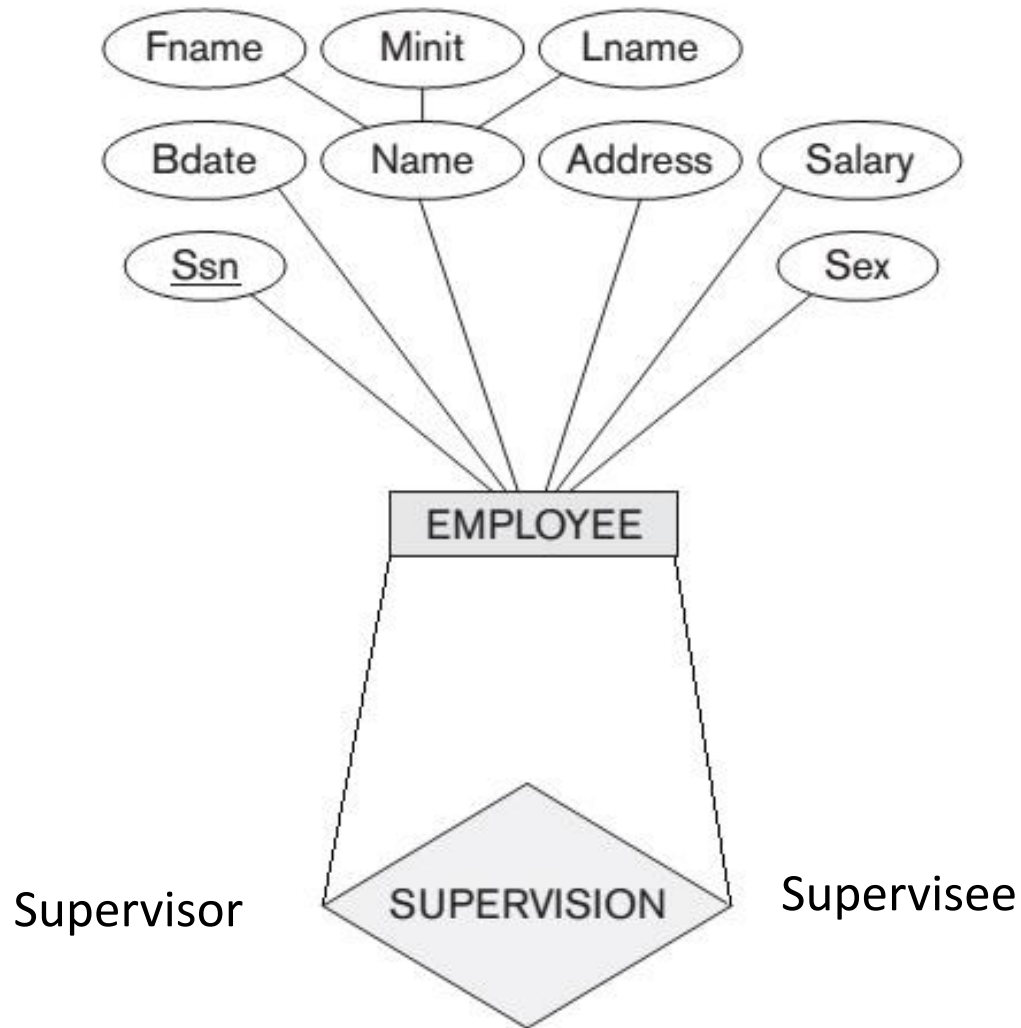
In some cases an entity type participates more than once in a relationship in different roles

□ Such relationships are called **recursive**

□ Each instance of EMPLOYEE type plays one of two roles:
supervisor(1) or
employee (2)

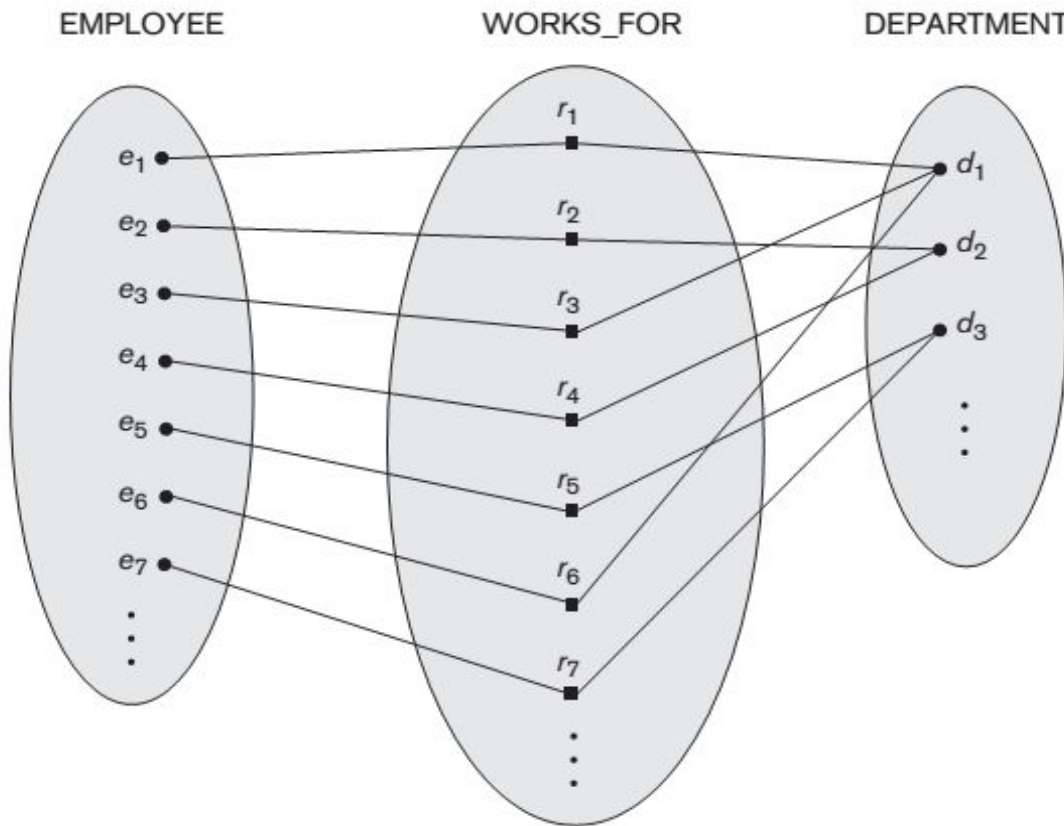


Representation in ER Schema



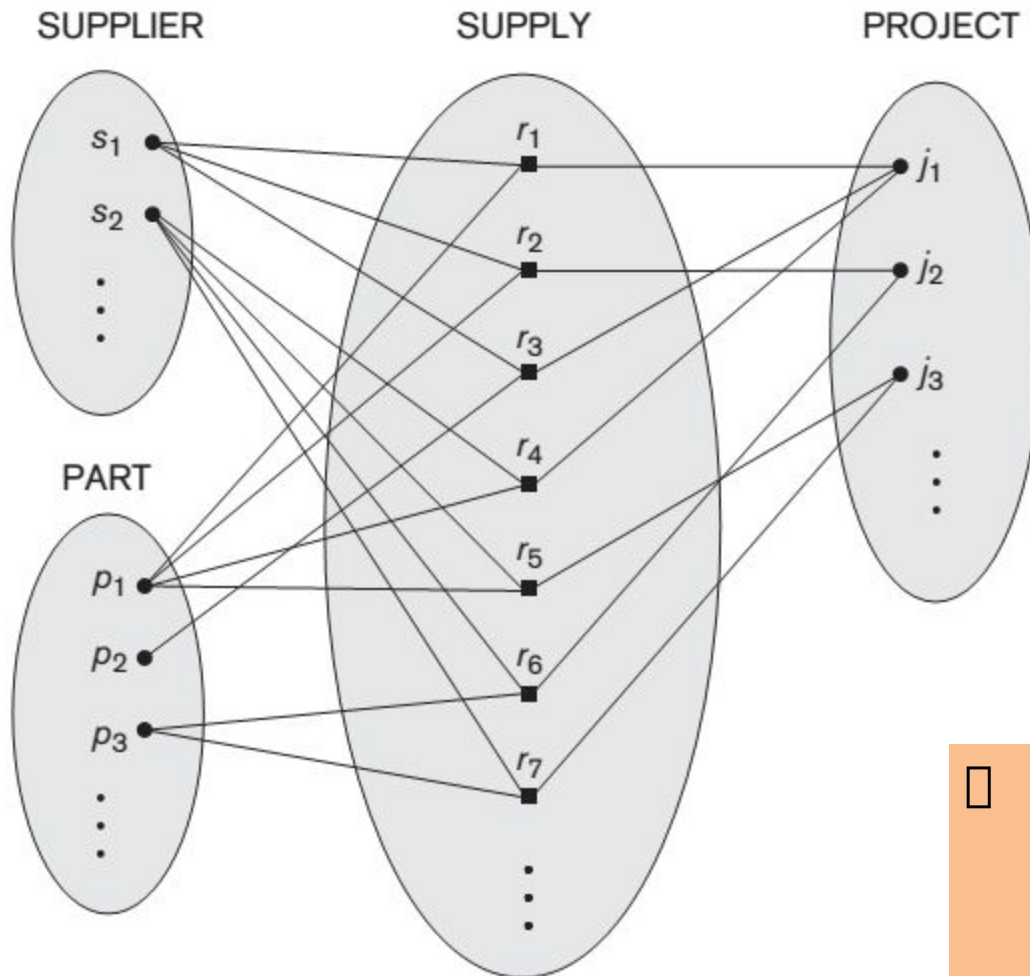
Degree of a Relationship

□ The **degree** of a relationship is the number of participating entity types



- ✓ WORKS_FOR relationship is of degree two
- ✓ A relationship type of degree two is called **binary**

Degree of a Relationship



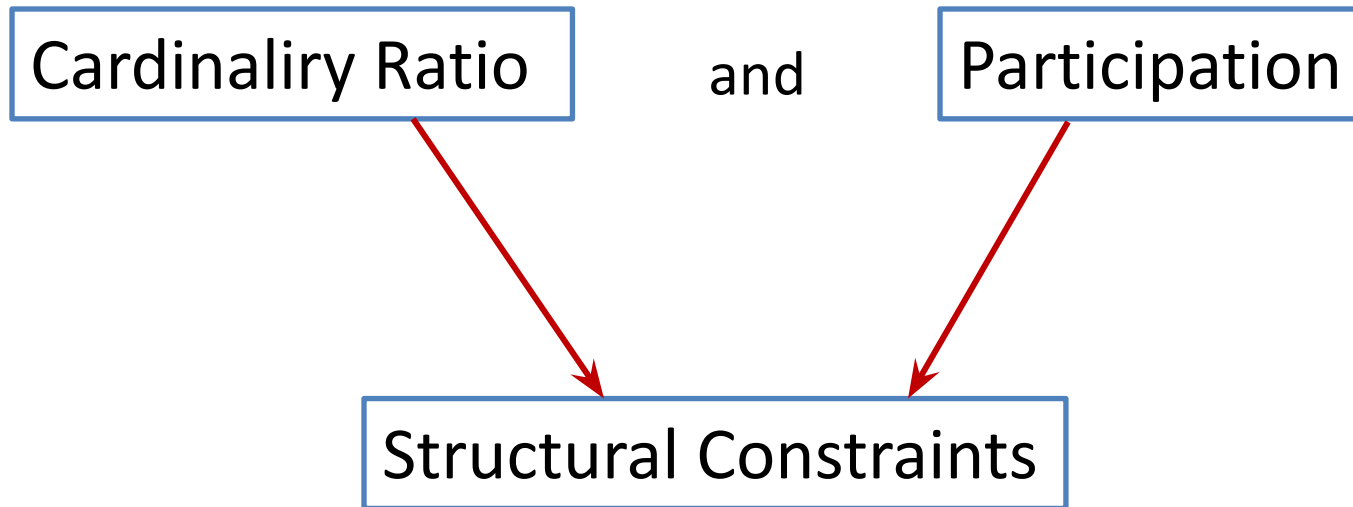
✓ SUPPLY relationship is of degree three

✓ A relationship type of degree three is called **ternary**

□ If N entity types participate in a relationship then such relationship is of degree N

Constraints on Binary Relationships

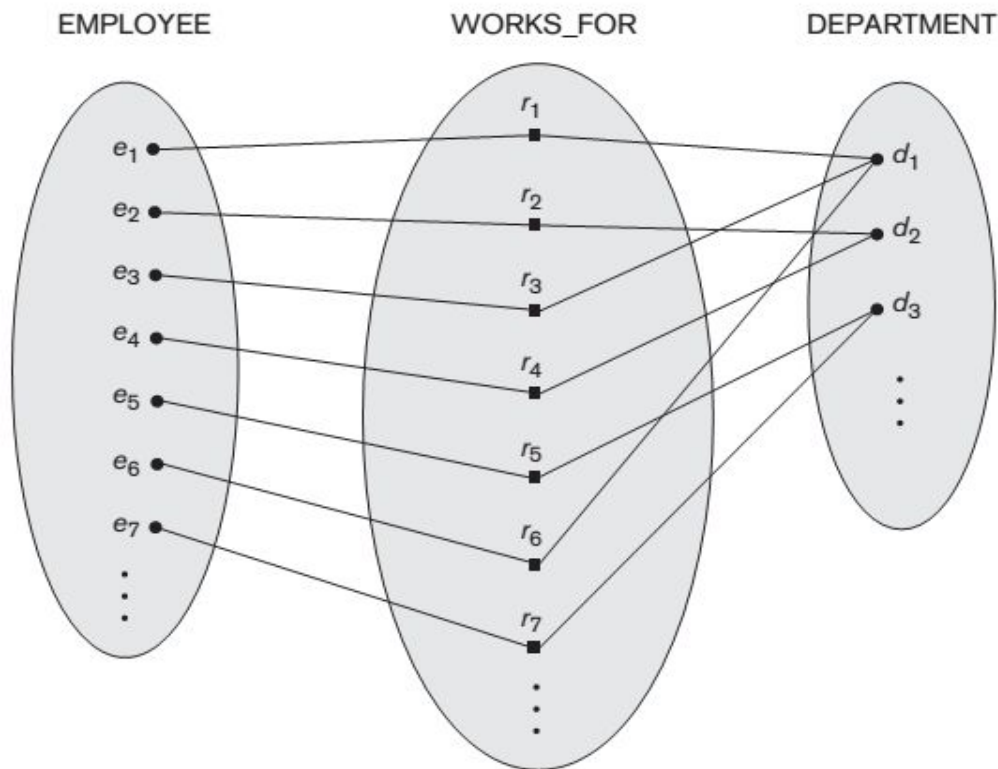
Two main types of binary relationship constraints:



These constraints are determined from the miniworld situation that the relationships represent

Cardinality Ratios for Binary Relationships

- The **cardinality ratio** for a binary relationship specifies the *maximum* number of relationship instances that an entity can participate in (determined from the Miniworld situation)



For WORKS_FOR relationship
DEPARTMENT : EMPLOYEE
cardinality ratio is 1:N

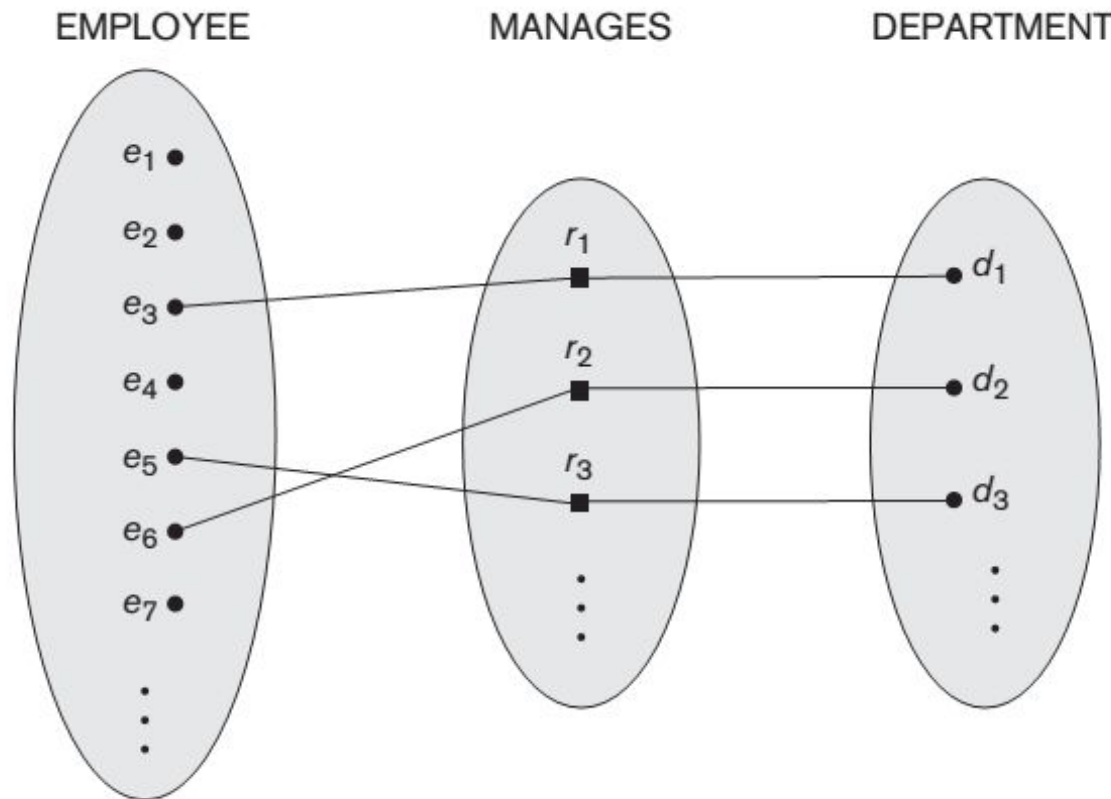
Possible cardinality ratios
for binary relationships
are:

- 1:1 (one to one)
- 1:N (one to many)
- M:N (many to many)

Example of 1:1 relationship

Miniworld rules

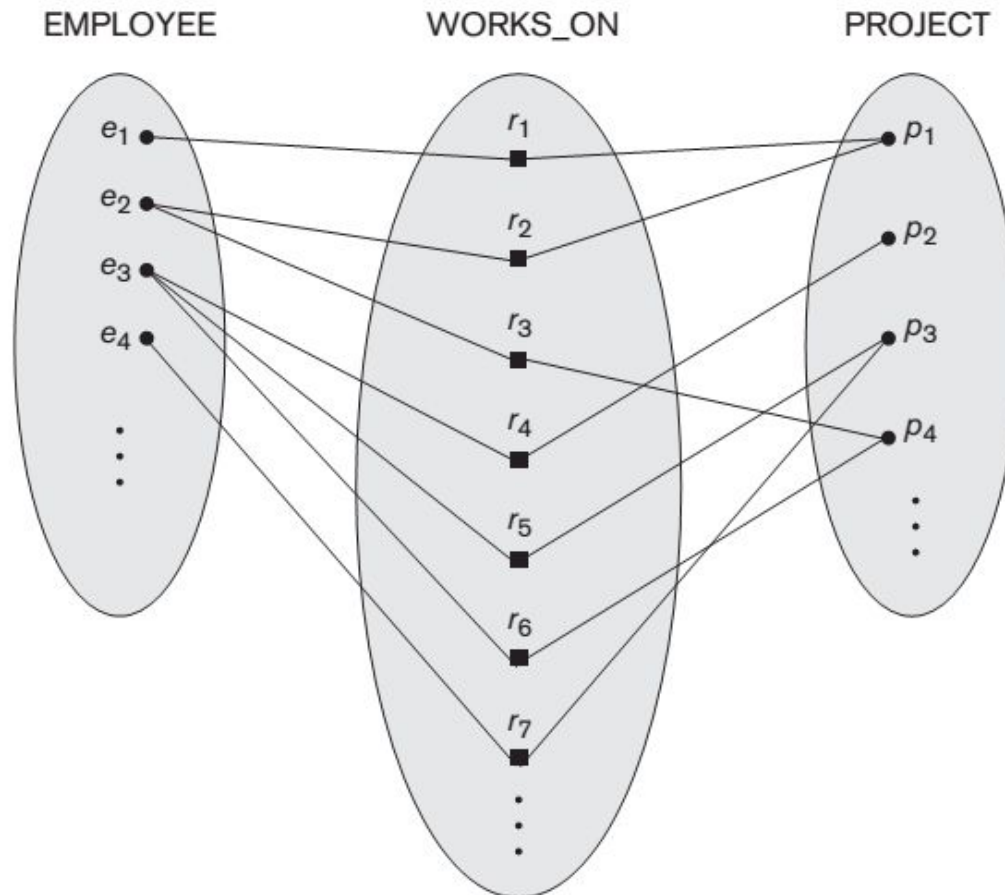
- Employee can manage one department only
- Department can have one manager only



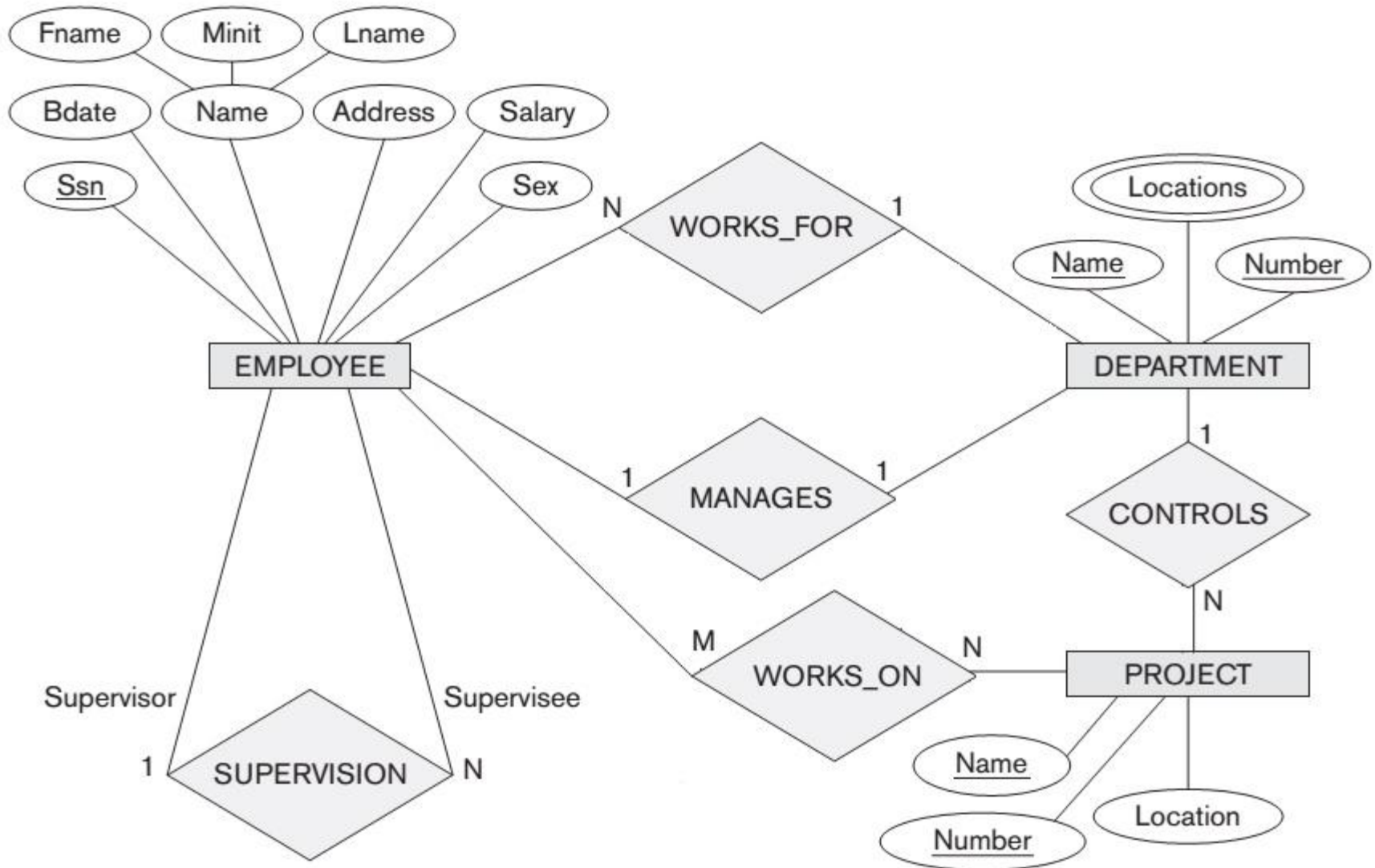
Example of M:N relationship

Miniworld rules

- Employee can work on several projects
- Project can have several employees



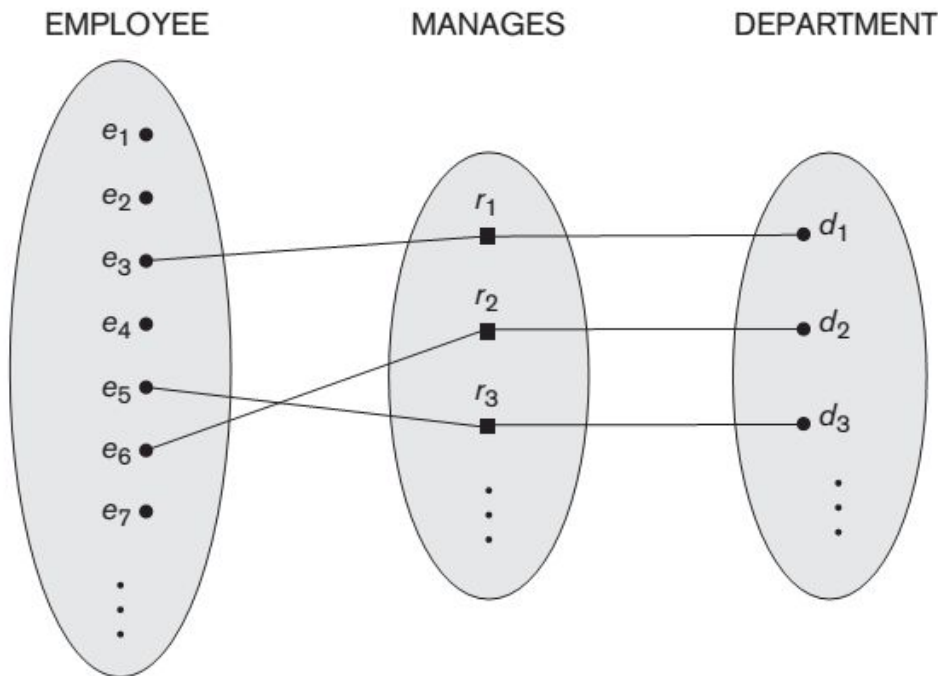
Representation in ER Schema



Participation constraints

□ **Participation** constraint specifies the *minimum* number of relationship instances that each entity can participate in

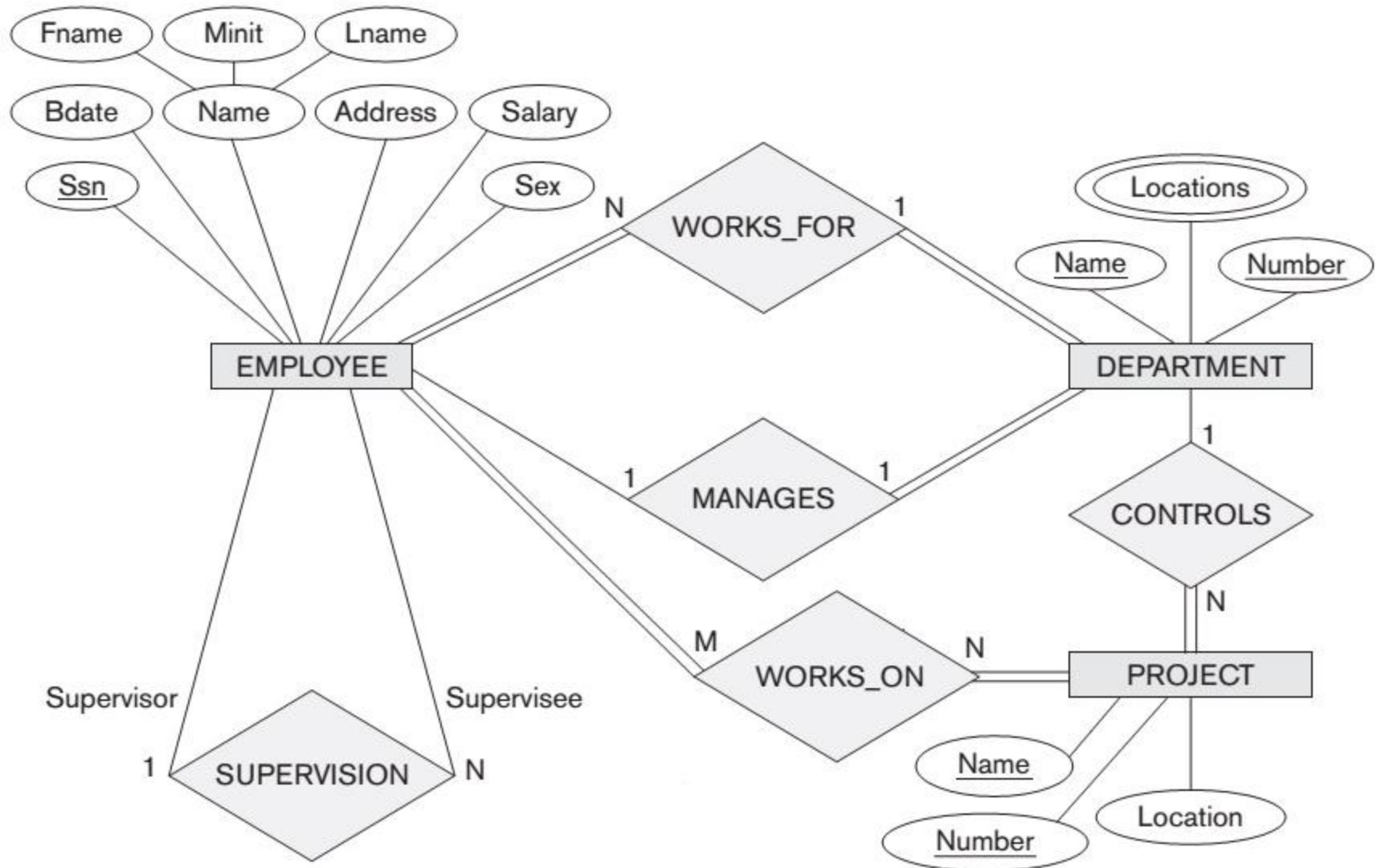
✓ There are two types of participation constraints—**total** and **partial**



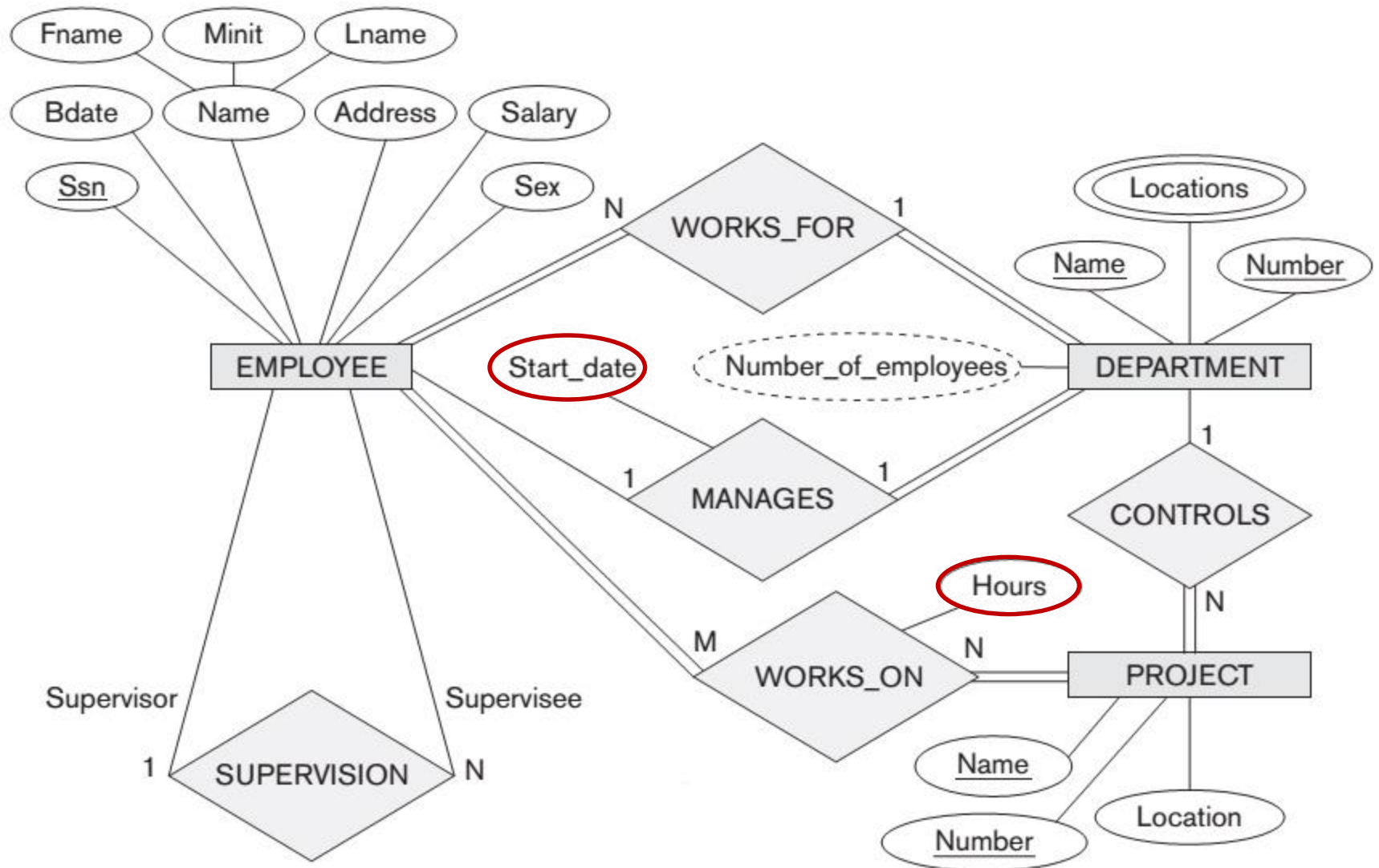
Participation of DEPARTMENT in MANAGES is called **total participation**, meaning that every department must be managed by one employee.

Participation of EMPLOYEE in MANAGES is called **partial participation**, meaning that a employee may or may not be a manager of a department.

Representation in ER Schema



Attributes of Relationships



Weak Entity Types

Entity types that do not have key attributes of their own are called **weak entity types**

In contrast, those entity types that do have a key attribute are called **strong entity types**

- ✓ A weak entity type always has a *total participation constraint* with respect to its identifying relationship because a weak entity cannot be identified without an owner entity
- ✓ A weak entity type normally has a **partial key**, which is the attribute that can uniquely identify weak entities that are *related to the same owner entity*

