KOÇ UNIVERSITY

Database Management Systems

Entity-Relationship Model

M. Emre Gürsoy

Assistant Professor
Department of Computer Engineering
www.memregursoy.com

Basic Definitions

- **Data:** Known facts that can be recorded and have an implicit meaning.
- **Database (DB):** A collection of related data.
- **Database Management System (DBMS):** A software package or system to facilitate the creation and maintenance of a computerized database.

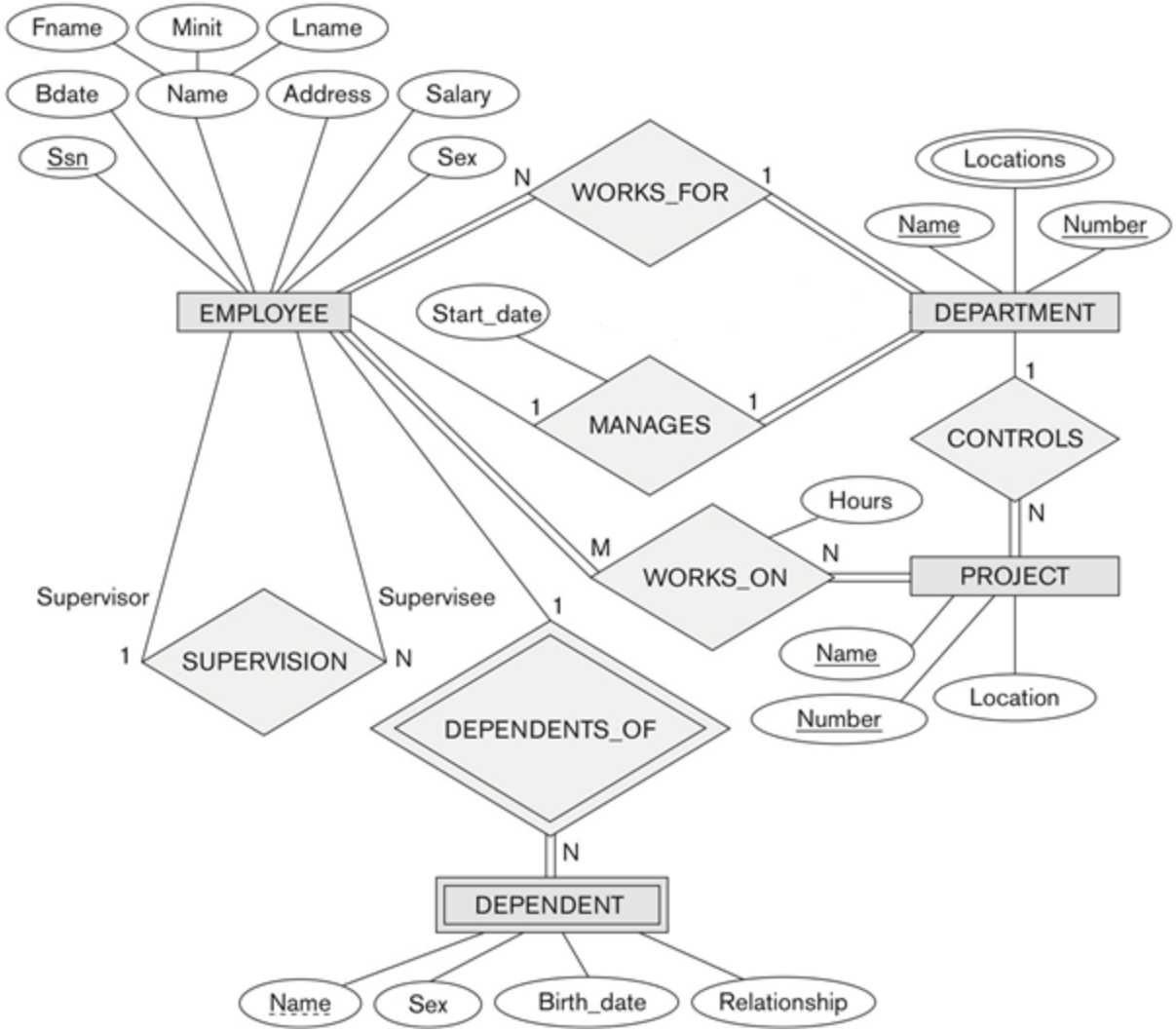
- **Metadata:** Data that provides information about other data, i.e., data about data.
  - E.g., table names, data types (int/string/..), ...
- **Mini-world:** Some part of the real world about which data is stored in a database.
  - E.g.: instructors, students, courses at a university

Conceptual Design

- As in any (software) project, you should **think** and **plan** before you take action.
- In the realm of databases:
  - Understand the mini-world
  - Collect and analyze the requirements
  - Conceptual design of DB
  - Implementation of DB

- **Entity-Relationship Model (ER model)** is a widely accepted standard for conceptual design of DBs.
  - Not tied to a language or DBMS software

Sample ER Diagram



Entities

- Two key concepts of the ER model:
  - Entities
  - Relationships

- **Entity** is a "thing" or "object" in the mini-world that exists and can be distinguished from other "things" or "objects".
  - STUDENT, COURSE
  - EMPLOYEE, DEPARTMENT, PROJECT

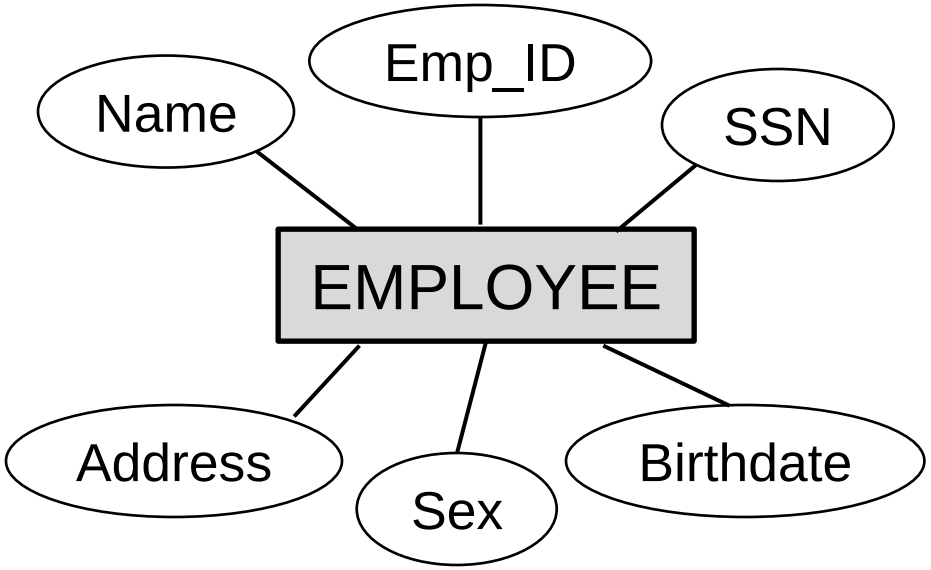
STUDENT

COURSE

EMPLOYEE

Attributes

- **Attributes** are properties used to describe an entity.
  - EMPLOYEE entity has: Name, Emp_ID, SSN, Address, Sex, Birthdate, ...
- A specific entity has a value for each of its attributes.
- An employee:
  - Name = "John Smith"
  - EMP_ID = "00829"
  - SSN = "123456789"
  - Address = "156 Peachtree St, Apt 13, Atlanta, GA"
  - Sex = "Male"
  - Birthdate = "12 February 1972"

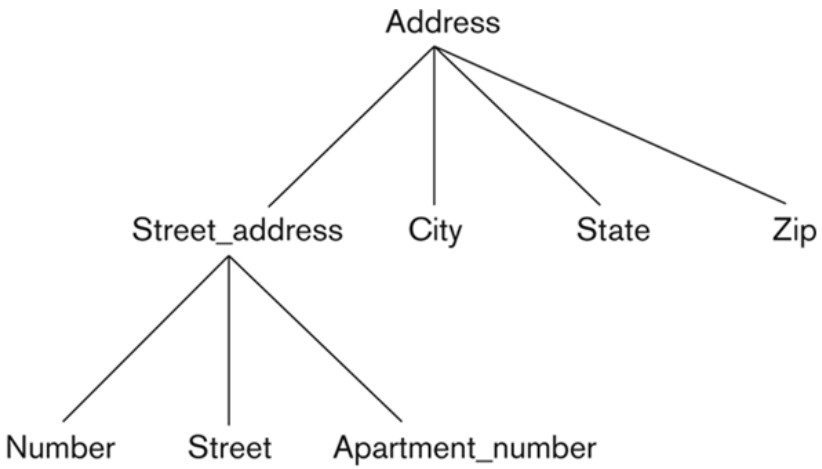


Types of Attributes

- Simple attribute: Each entity has a single, atomic value for the attribute.
  - Examples: SSN, Sex
- Composite attribute: The attribute is composed of several components.
  - Examples: Address, Name
  - Address (Apt#, House#, Street, City, Country, Zip)
  - Name (FirstName, MiddleName, LastName)
- Multi-valued attribute: An entity may have multiple values for that attribute.
  - Example: Color attribute of CAR entity.

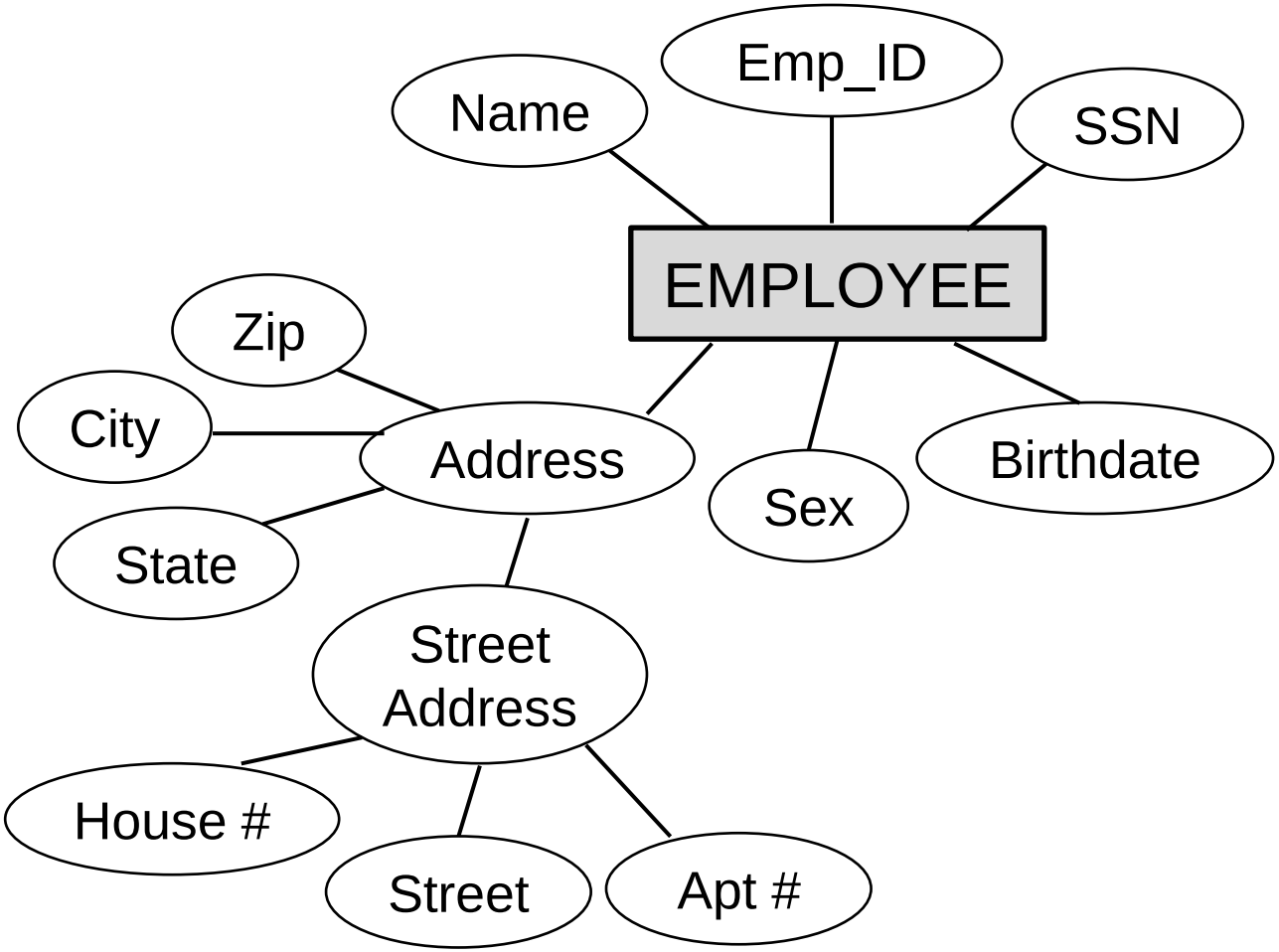
Composite Attributes

- It is possible to have multiple levels in composite attributes (nesting), but this is rare.
  - Rule of thumb: No more than 2 levels
  - If you need more levels, it may be a hint that you're doing something wrong
    - Maybe the attribute should have been an entity



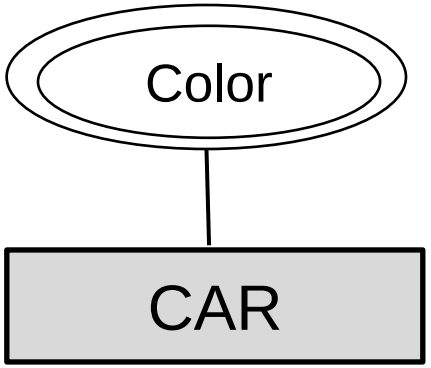
Examples

- Example with nested composite attribute:

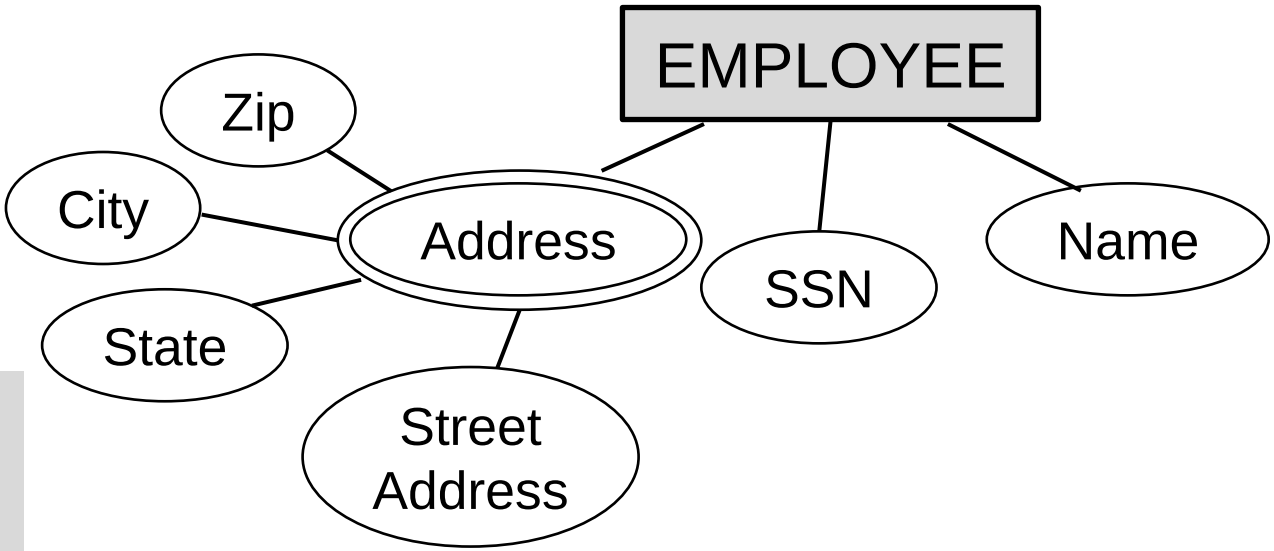


Examples

- Example with multi-valued attribute:
  - Must draw double ellipse



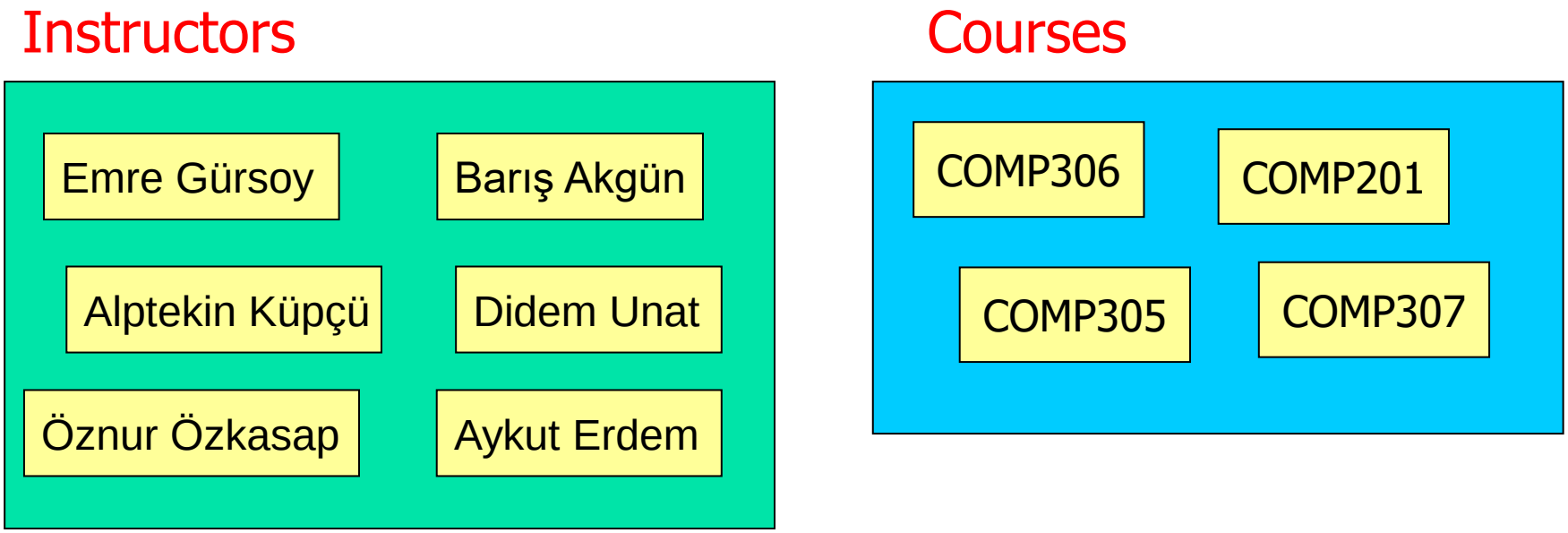
- It is possible for an attribute to be composite and multi-valued simultaneously.



Double ellipse around Address but not Zip, City, State. Why?

Entity Set

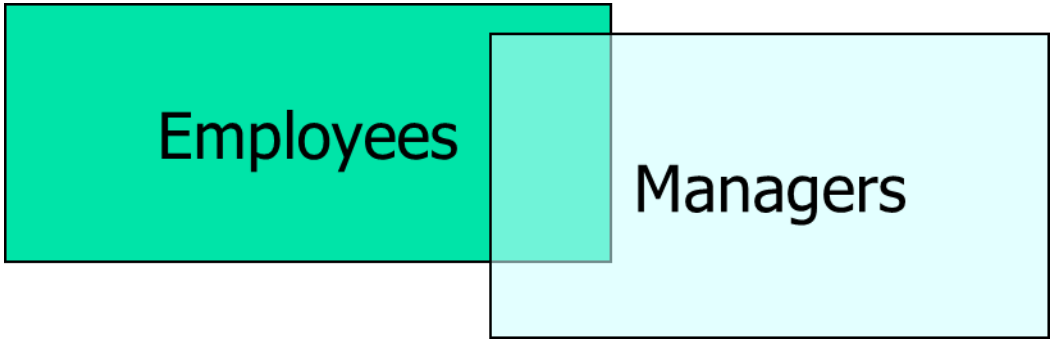
- Each entity has multiple instances stored in the DB
  - Many employees, many departments, ...
  - They share the same properties
  - This is called the **entity set** or **entity collection**



Entity Set

- Each entity has multiple instances stored in the DB
  - Many employees, many departments, ...
  - They share the same properties
  - This is called the **entity set** or **entity collection**

- **Question:** Can entity sets overlap?
  - Yes!

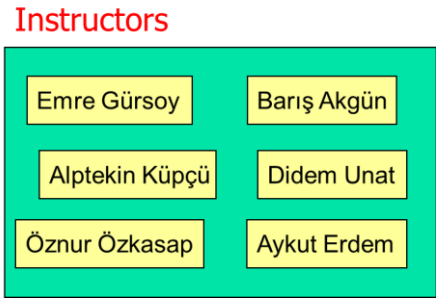


Value Set

- Each simple attribute is associated with a **value set**
  - Specifies the set of values associated with the attribute, i.e., domain of permitted values
- Examples:
  - SSN -> nine digits
  - LastName -> Character string, up to 15 characters
  - State (under Address) -> value set corresponds to the states in the USA, e.g.: {CA, GA, IL, NY, NJ, TX, ...}

Key Attribute

- **Key attribute:** An attribute for which each entity in the entity set must have a unique value.
  - Unique to each INSTRUCTOR
  - Unique to each EMPLOYEE
  - Unique to each COURSE, etc.

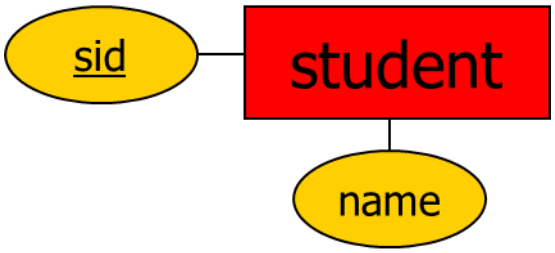


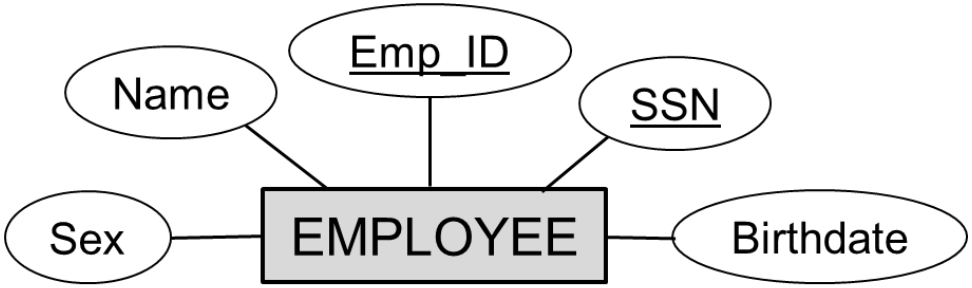
- Which of the following are key attributes of EMPLOYEE?
  - Name, SSN, Emp_ID, Address, Sex, Birthdate
- A key attribute may be composite.
  - VehicleRegistration is a key of CAR entity
  - VehicleRegistration contains (Number, State)

Key Attribute

1. An entity type may have more than one key.
   - EMPLOYEE: Emp_ID and SSN
2. Each key is underlined in the ER diagram.

- Note that these two properties are true for the ER model, but not for the Relational model.
  - In relational model, only one primary key is underlined.





Example

(a)

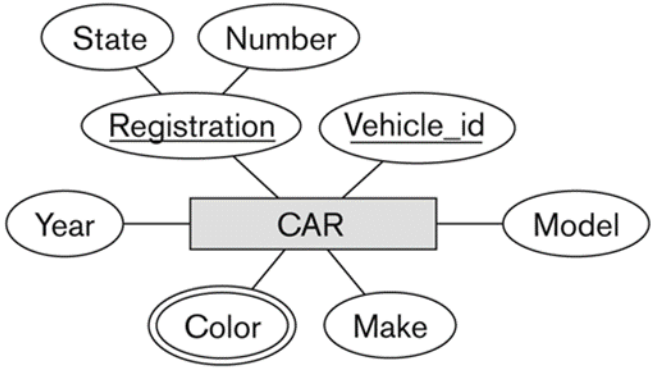


Figure 3.7
The CAR entity type with two key attributes, Registration and Vehicle_id. (a) ER diagram notation. (b) Entity set with three entities.

(b)

CAR
Registration (Number, State), Vehicle_id, Make, Model, Year, {Color}

CAR_1
((ABC 123, TEXAS), TK629, Ford Mustang, convertible, 2004 {red, black})

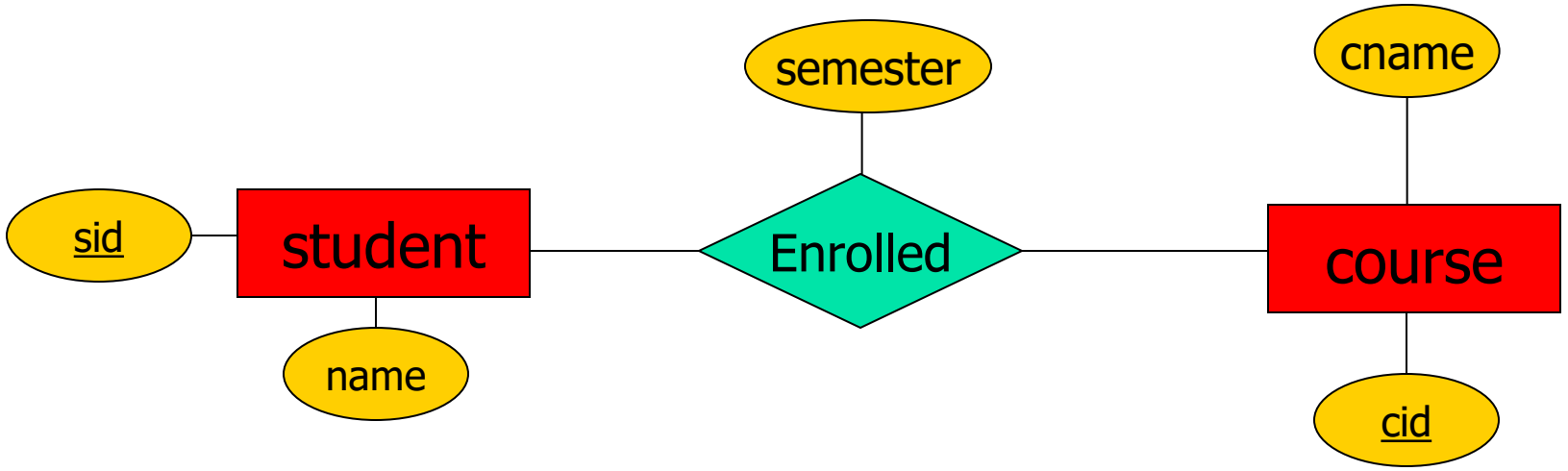
CAR_2
((ABC 123, NEW YORK), WP9872, Nissan Maxima, 4-door, 2005, {blue})

CAR_3
((VSY 720, TEXAS), TD729, Chrysler LeBaron, 4-door, 2002, {white, blue})

⋮

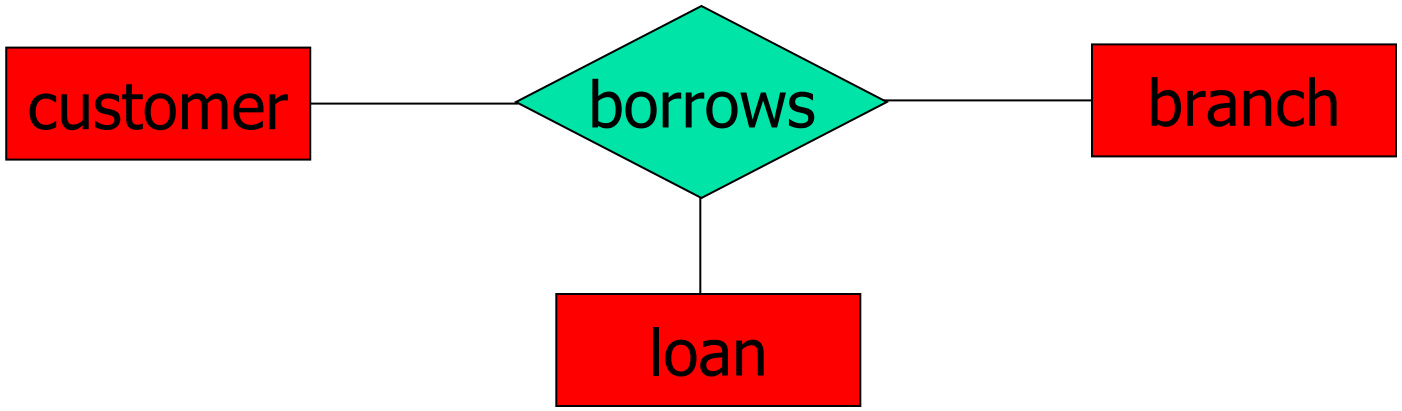
Relationships

- A **relationship** relates two or more distinct entities with a specific meaning.
  - Student is ENROLLED in a course.
  - Employee WORKS IN a department.
- Represented using diamonds in an ER diagram.
- Relationships may also have attributes.



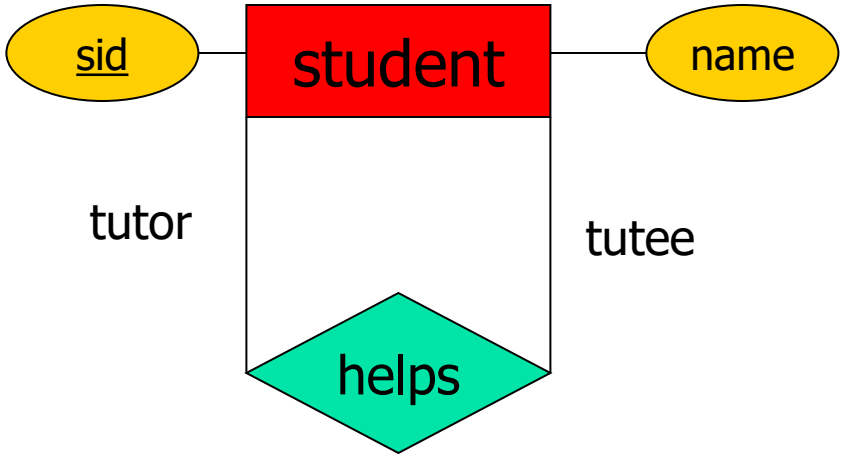
Degrees of Relationships

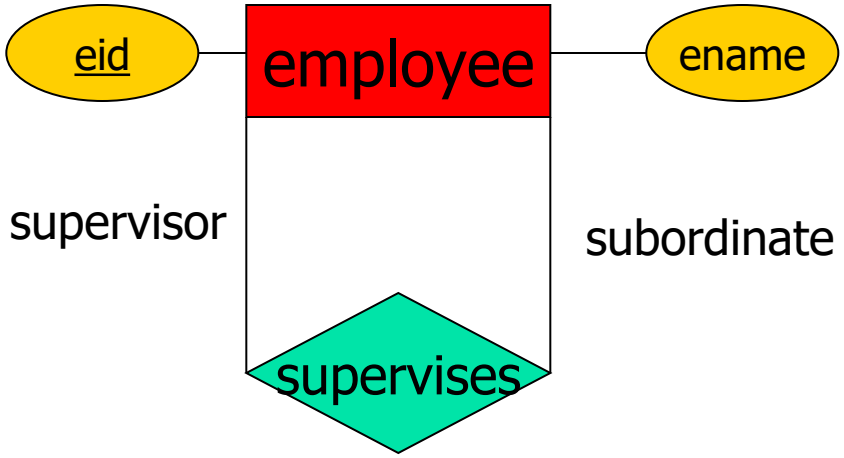
- **Degree** of a relationship is the number of entity sets that participate in that relationship.
  - Binary relationship: degree = 2
    - Example: ENROLLED relationship from previous slide
  - Ternary relationship: degree = 3
    - Example: BORROWS relationship
  - Higher degrees are uncommon (and not preferred)



Recursive Relationship

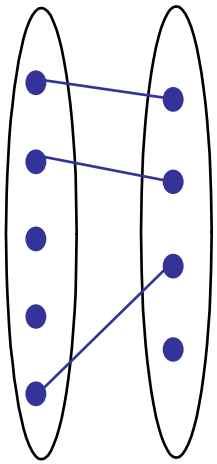
- **Recursive (aka "self-referencing") relationship:** Relationship among entities that belong to the same entity set.
- In recursive relationships, each entity has a role.
- Roles must be written explicitly on the ER diagram.
  - Only necessary in recursive relationships



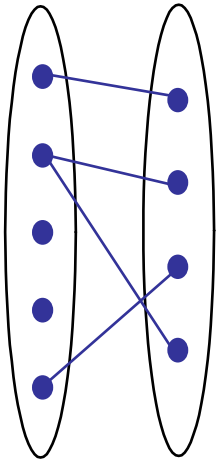


Relationship Cardinality

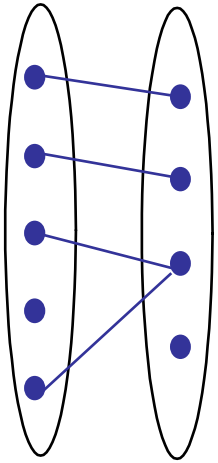
- One-to-one (1-1): employee manages department
- One-to-many (1-N): an instructor teaches multiple courses, but a course is taught by one instructor
- Many-to-one (N-1): one employee works in at most one dept, but one dept has many employees
- Many-to-many (N-M): students take courses



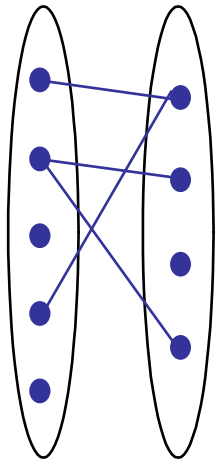
1-to-1



1-to Many



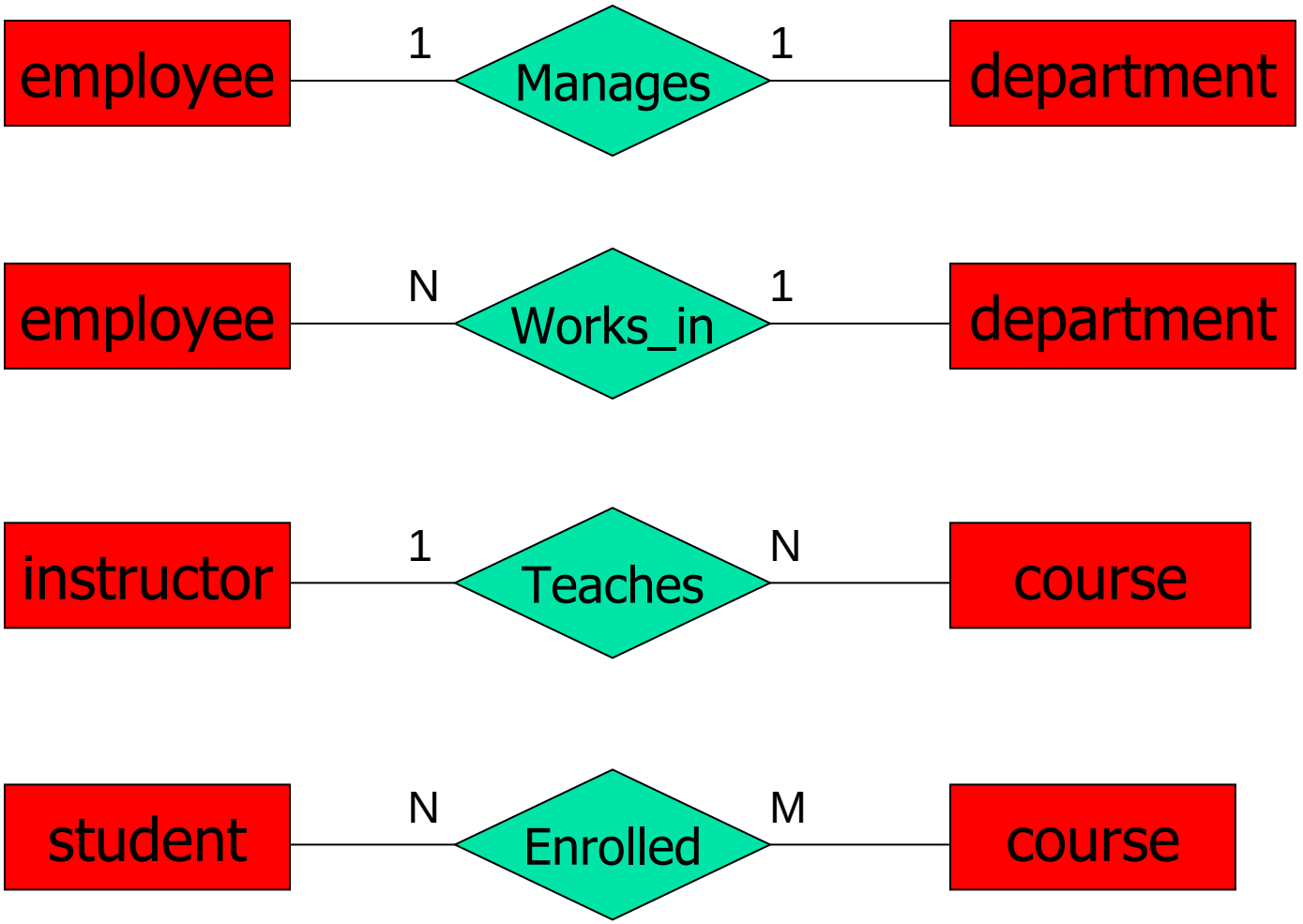
Many-to-1



Many-to-Many

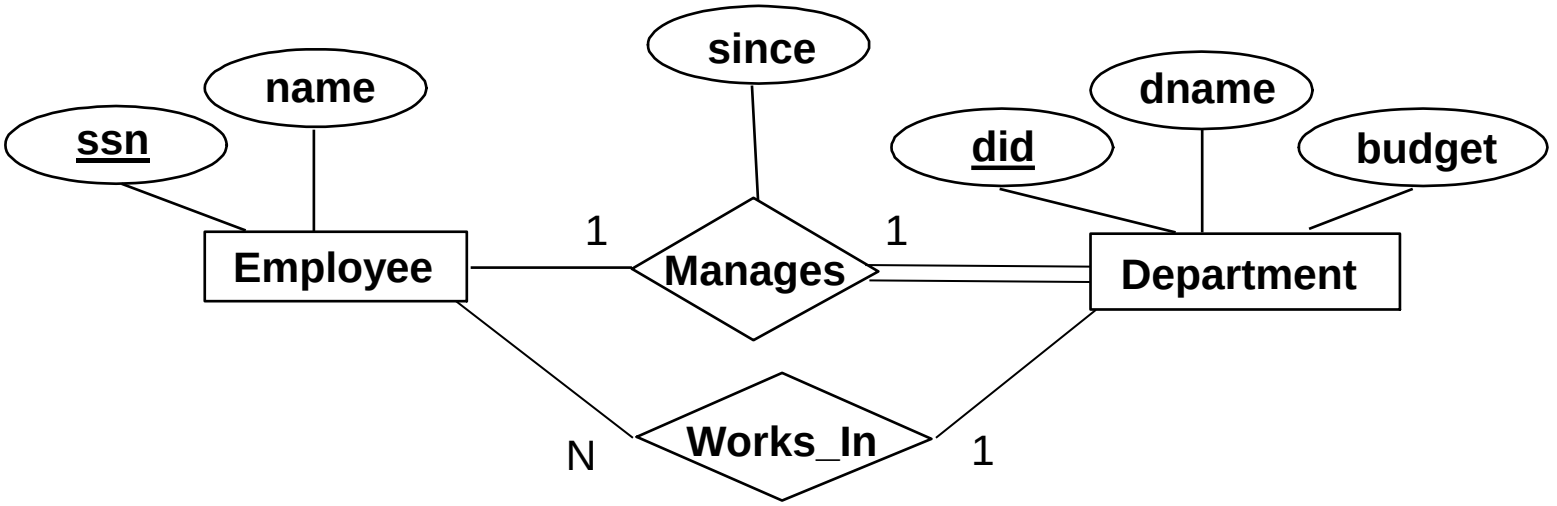
Relationship Cardinality

- You must show cardinalities on the ER diagram:



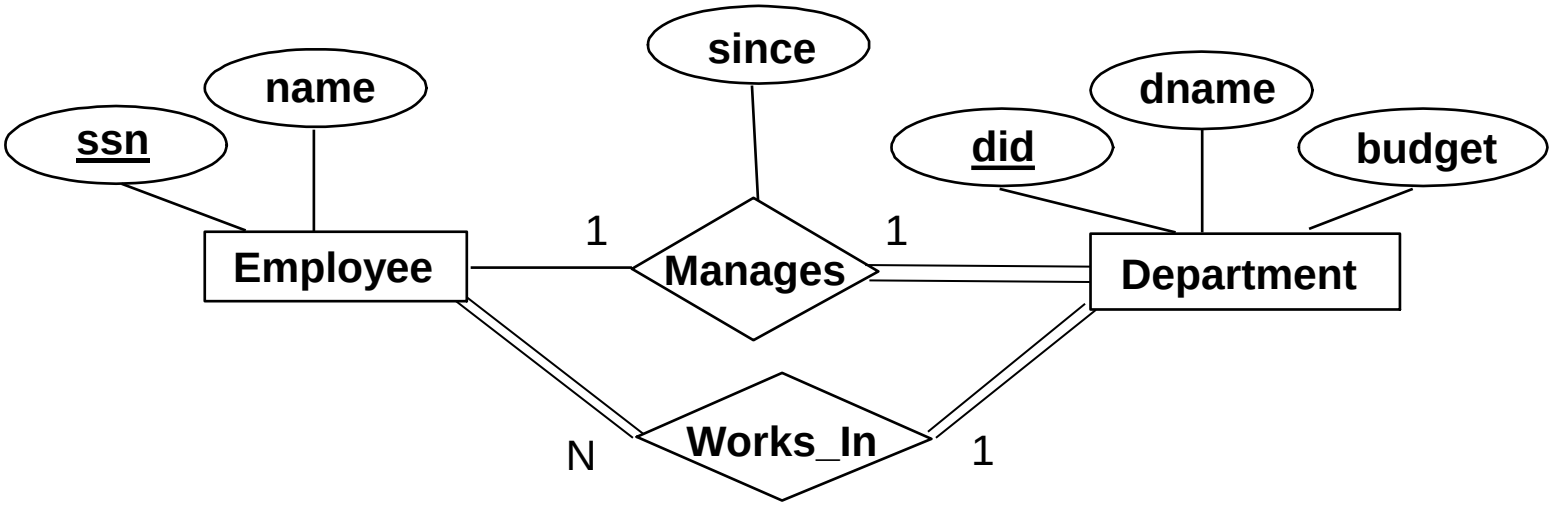
Participation Constraint

- If every department MUST have a manager, this is called a **participation constraint**.
  - Must participate in the MANAGES relationship.
- Participation constraints are denoted with double lines.



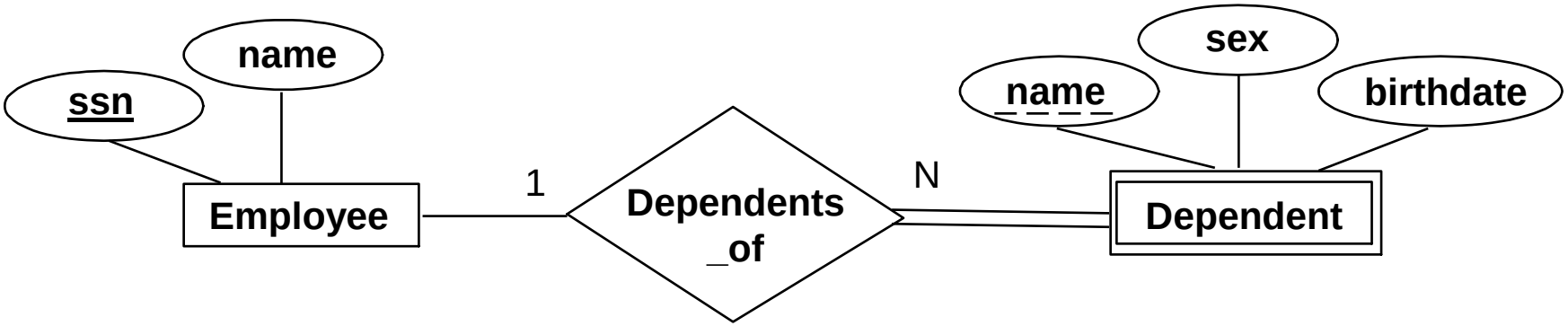
Participation Constraint

- Other examples?
  - Every employee must work in a department
  - Every department must have an employee working in that department
  - Must every employee manage a department?



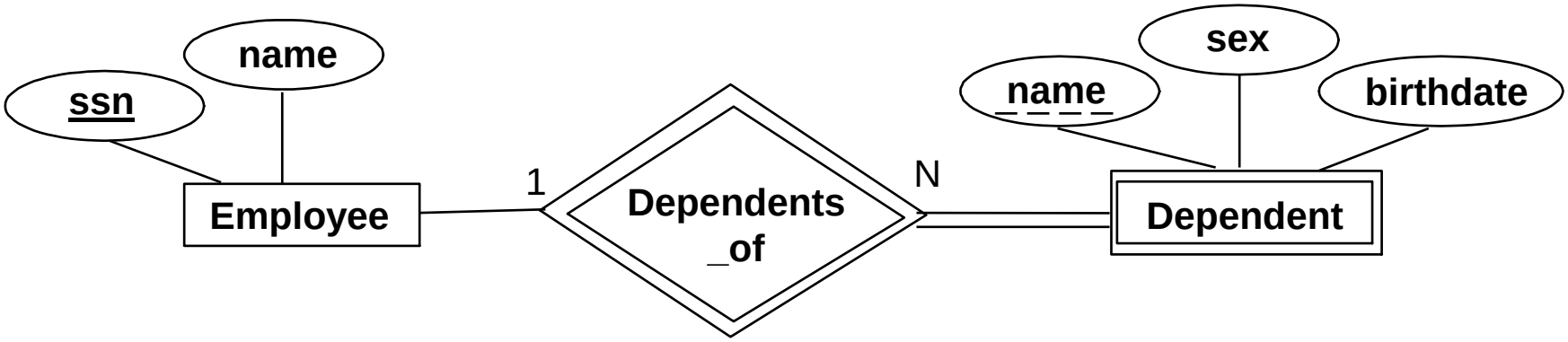
Weak Entities

- **Weak entity:** an entity type that depends on another entity type for identification.
  - Denoted with double lines around the entity
  - Doesn't have a key; instead has a partial key
  - Partial key is denoted with dashed underline



Identifying Relationship

- A weak entity must participate in an **identifying relationship** with its "owner" entity.
  - Dependents_of is an identifying relationship
  - Denoted with double lines around the relationship
- Whenever you have a weak entity, always think about what is its identifying relationship.



Company Database

- A company consists of several departments. Each department has a unique name and unique department number. A department may have multiple locations, such as New York, Los Angeles, Atlanta.
- The company has many employees. The unique identifier of an employee is his/her social security number (SSN). In addition, each employee has a birthdate, address, salary, sex, and name. The name consists of the first name, middle initial, and last name.
- Each employee works for exactly one department. A department may contain many employees, but it must contain at least one employee. Furthermore, each department is managed by an employee, who is the manager of the department. Each department must have a manager. An employee cannot manage multiple departments. Management start date must be included in the database.
- A department controls several projects in the company, and each project must have exactly one department that is controlling it. The project name and project number is unique in each project. The project is carried out at a single location.

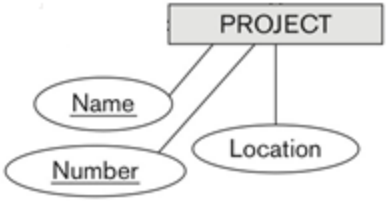
Company Database

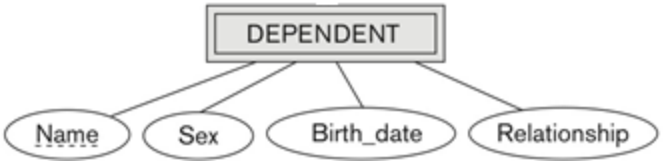
- Each employee can work on multiple projects and each project can have multiple employees working on it. The database should keep track of the number of hours spent by an employee on a project. Employees from different departments are allowed to work on the same project. There cannot be empty projects (i.e., no employee working on them). Every employee in the company must be working on at least one project.
- The company has a supervision policy. An employee (supervisee) can be supervised by an older employee (supervisor), who is more experienced. A supervisor can have multiple supervisees, but a supervisee has at most one supervisor. Not every employee must be a supervisor, and not every employee must be supervised by someone.
- Finally, employees can have dependents. Each employee can have several dependents, but a dependent is tied to a particular employee. The company keeps track of employees' dependents, but if the employee leaves, the dependents become irrelevant to the company. A dependent has a unique name. In addition to their name, the dependent’s sex, birthdate, and relationship to the employee must be stored in the database.

Entities in Company DB?

- DEPARTMENT (Name, Number, Locations)
- PROJECT (Name, Number, Location)
- EMPLOYEE (SSN, Birthdate, Name (Fname, Minit, Lname), Address, Salary, Sex)
- DEPENDENT (Name, Sex, Birth_date, Relationship)
  - Suspected weak entity

Company Database

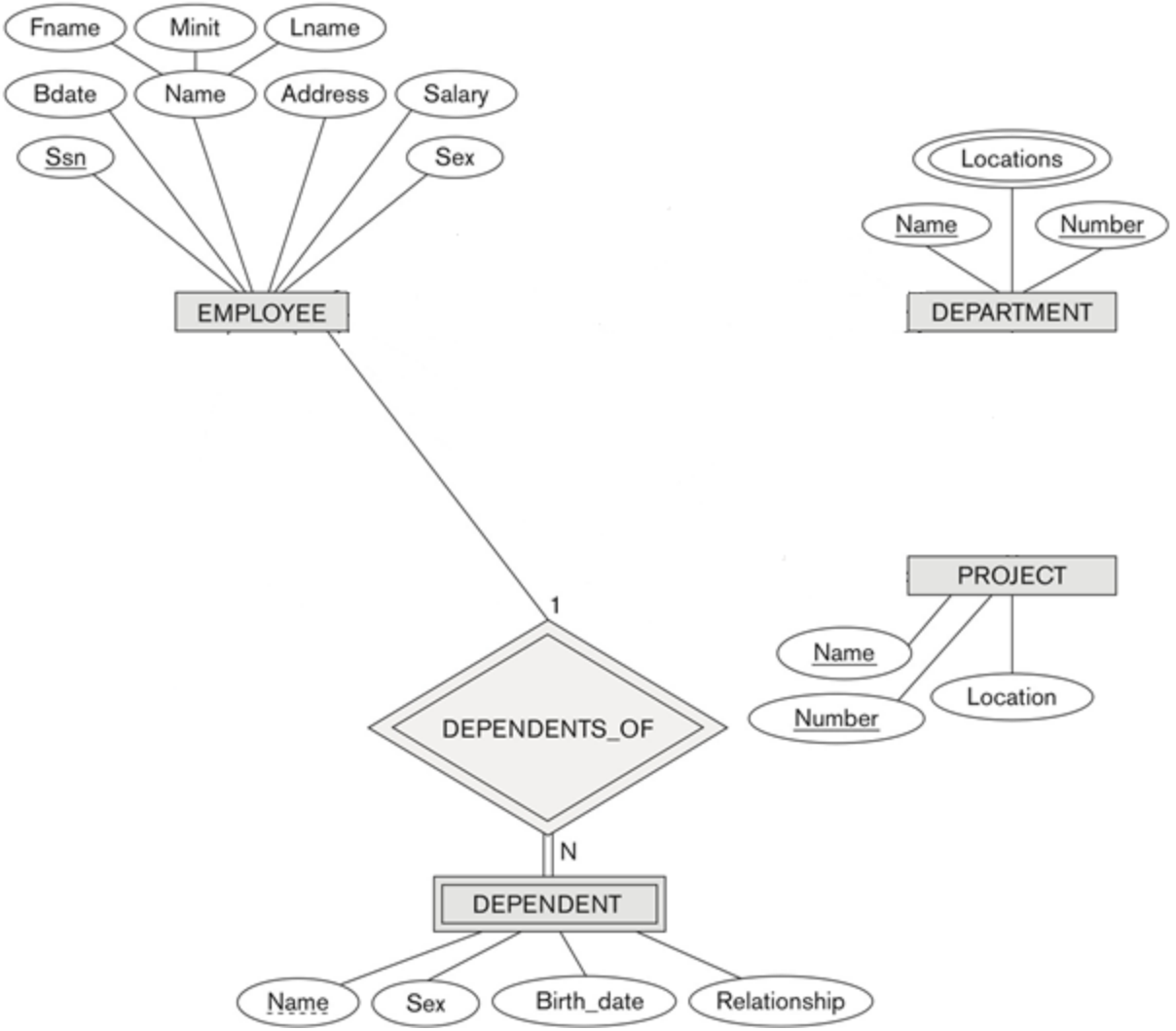




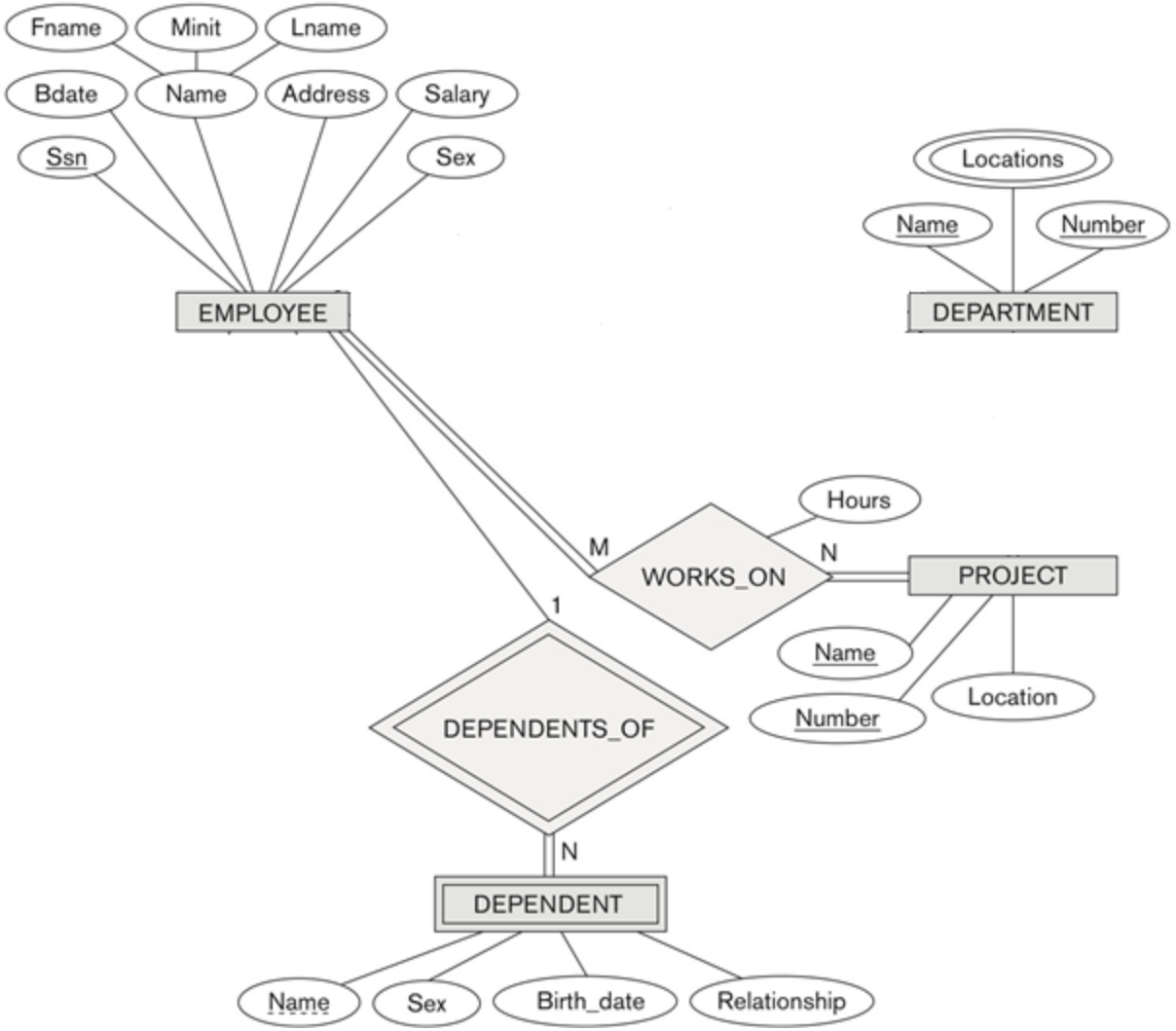
Relationships in Company DB?

- Department CONTROLS Projects
- Employee WORKS FOR Department
- Employee MANAGES Department
  - Must include start date of management
- Employee WORKS ON Project
  - Must include hours
- Employee SUPERVISES Employee
  - Supervisor – supervisee roles
- Dependents are DEPENDENTS_OF Employees
  - Identifying relationship

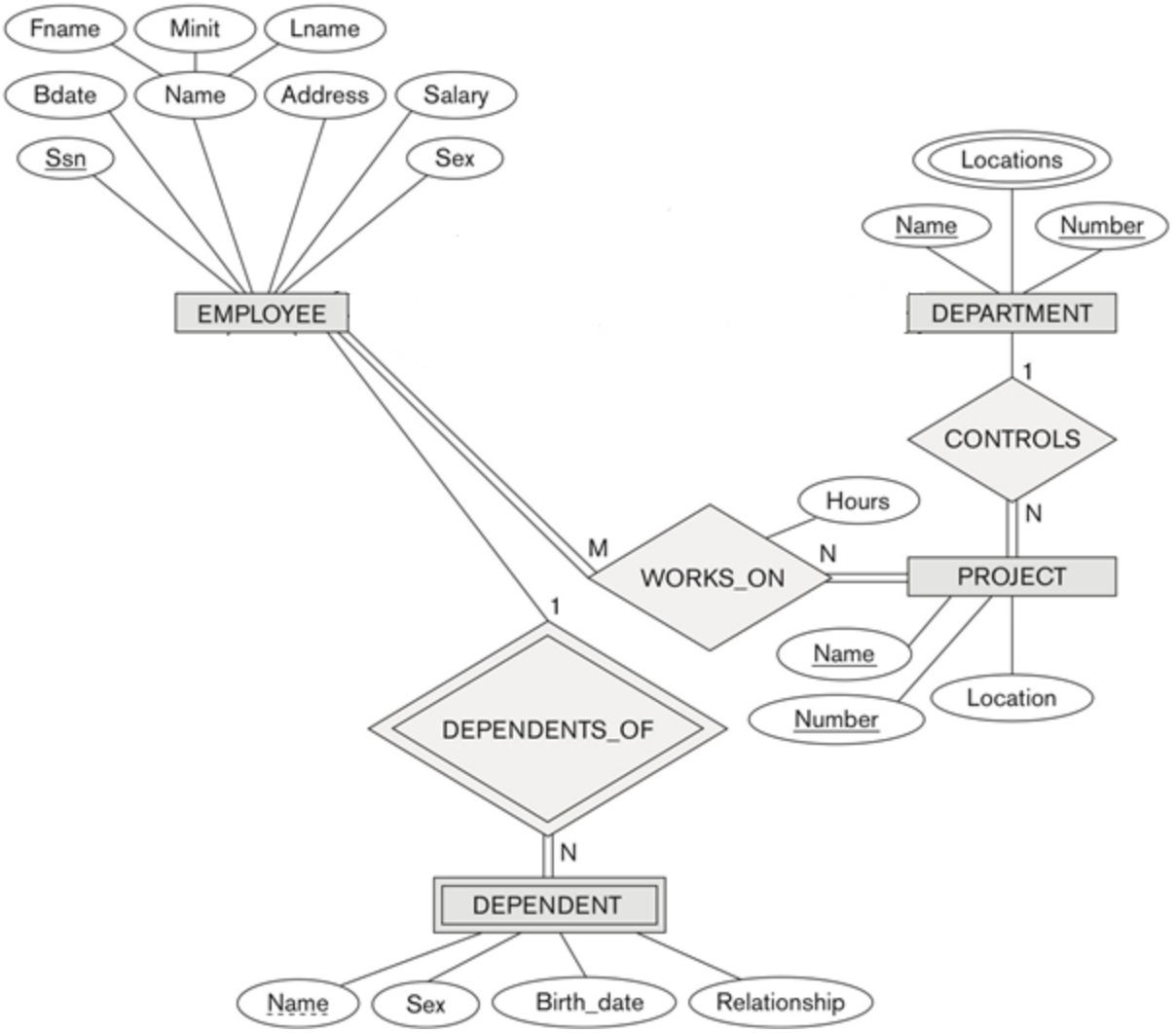
Company Database



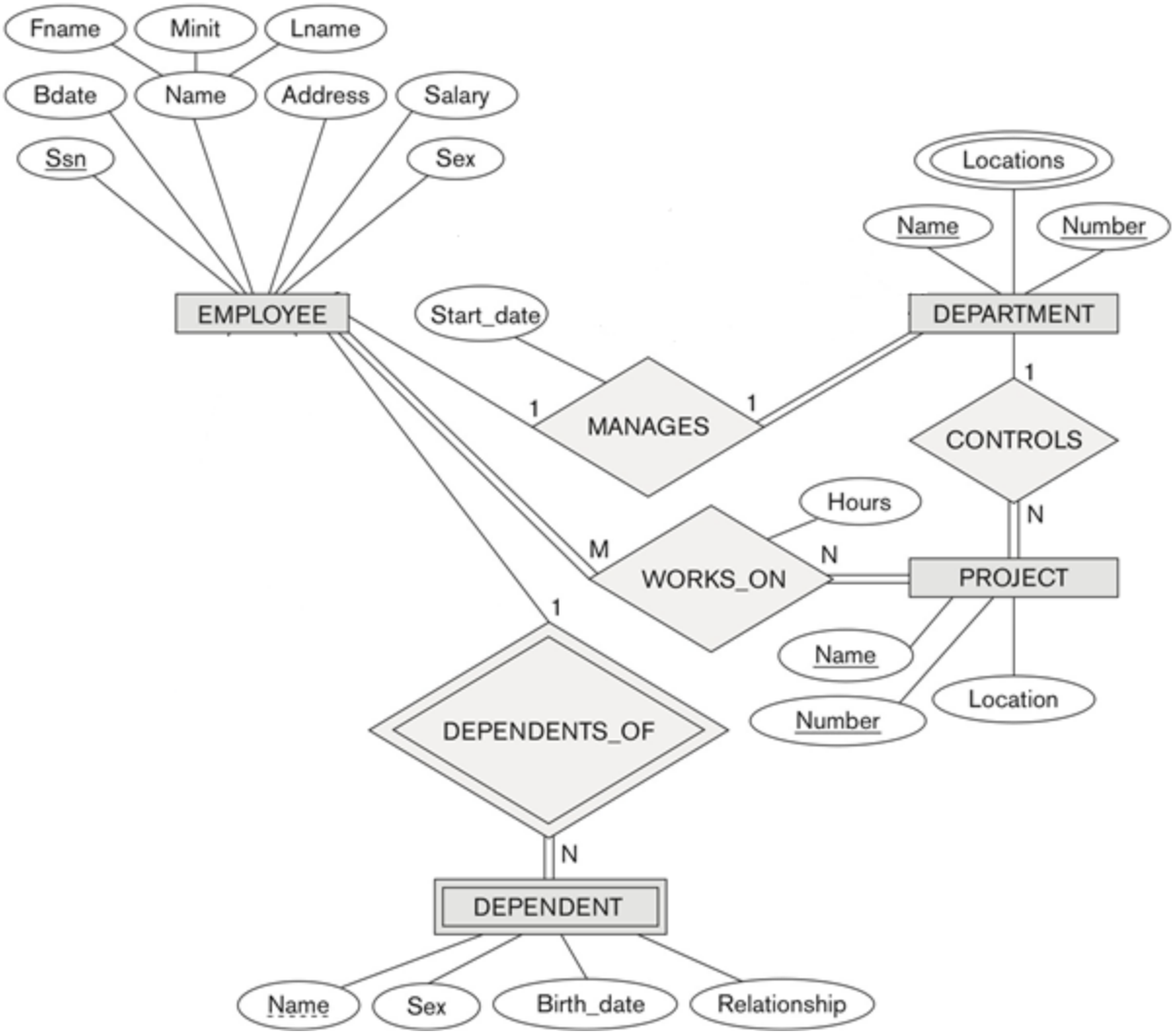
Company Database



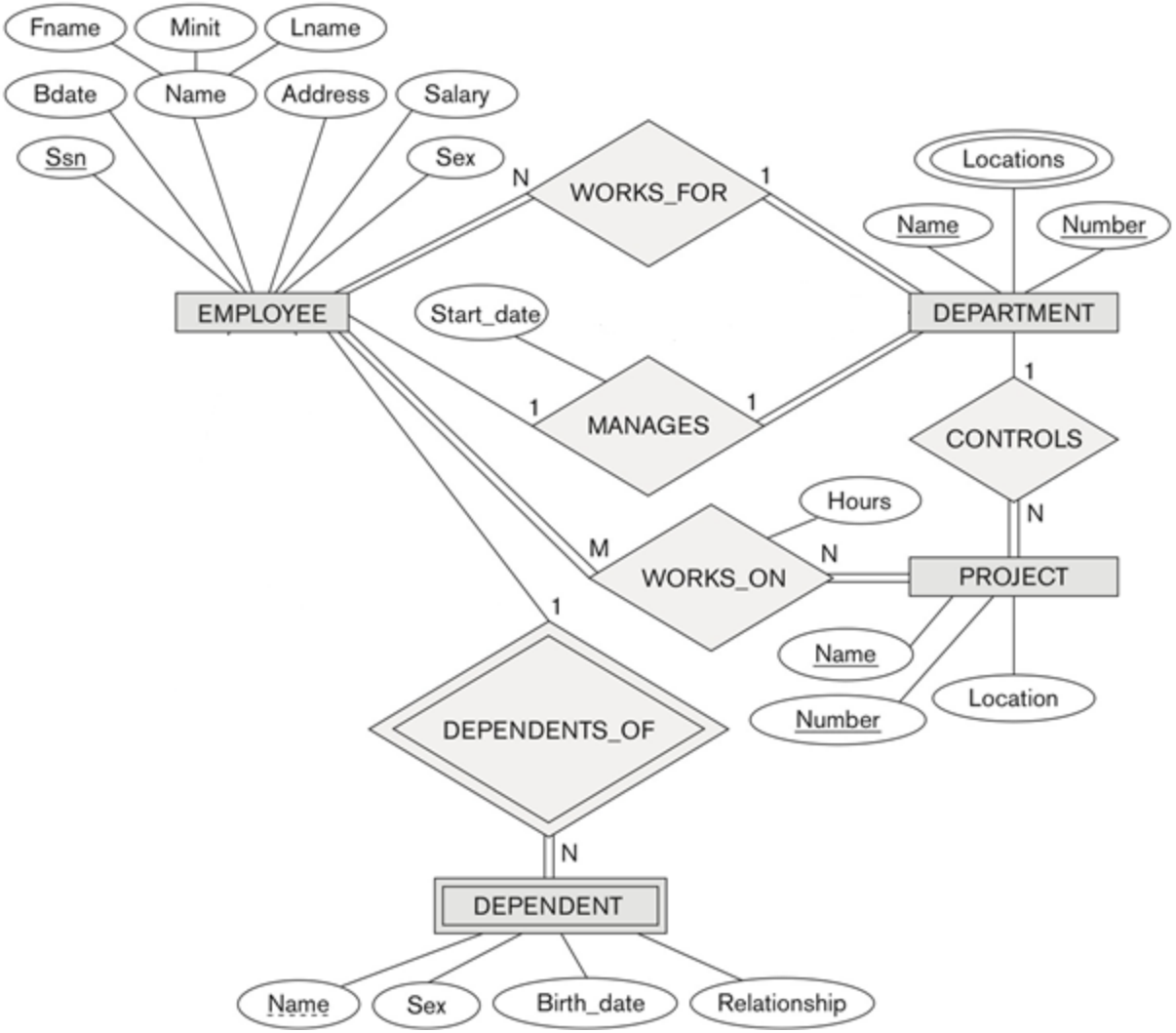
Company Database



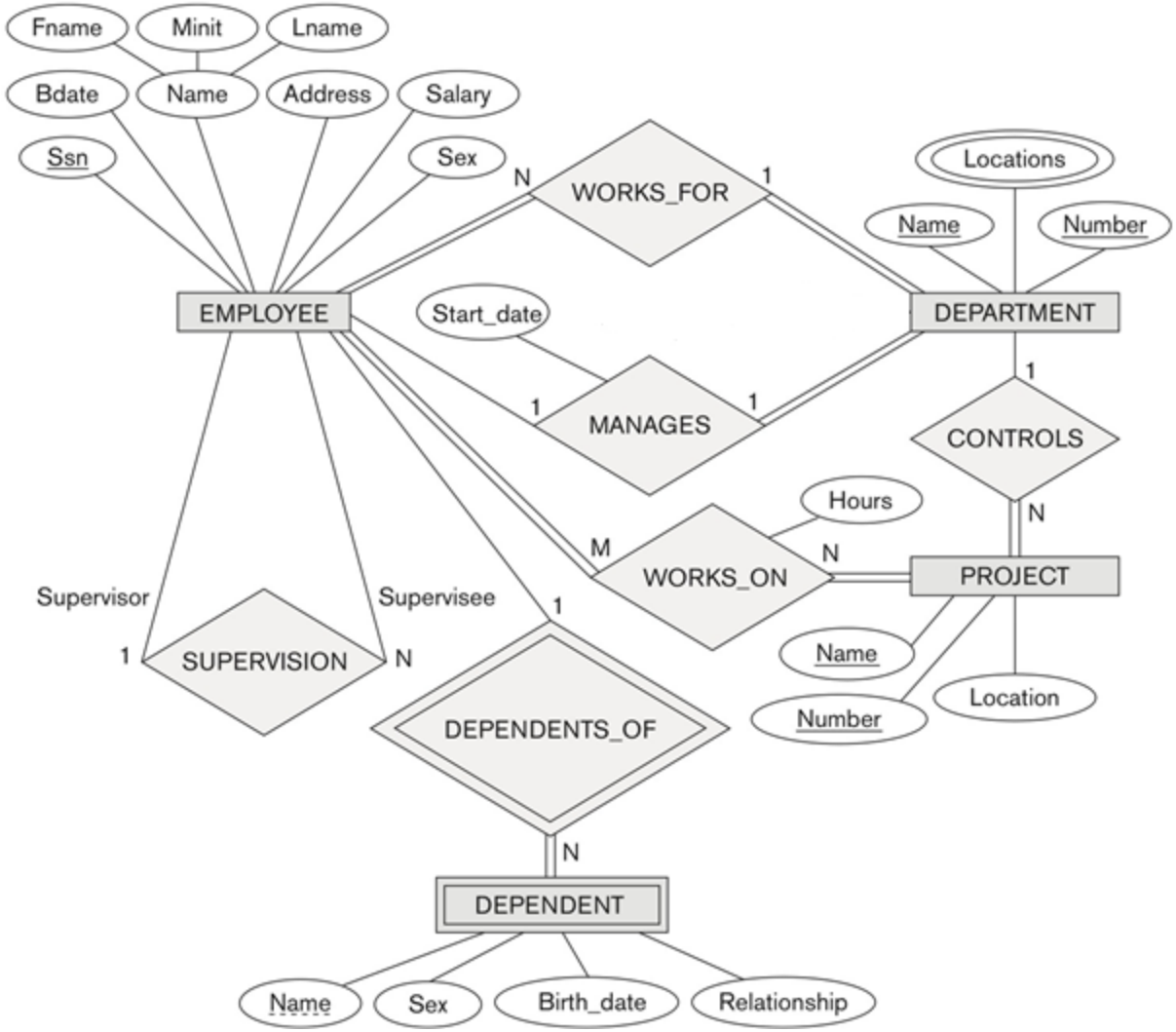
Company Database



Company Database



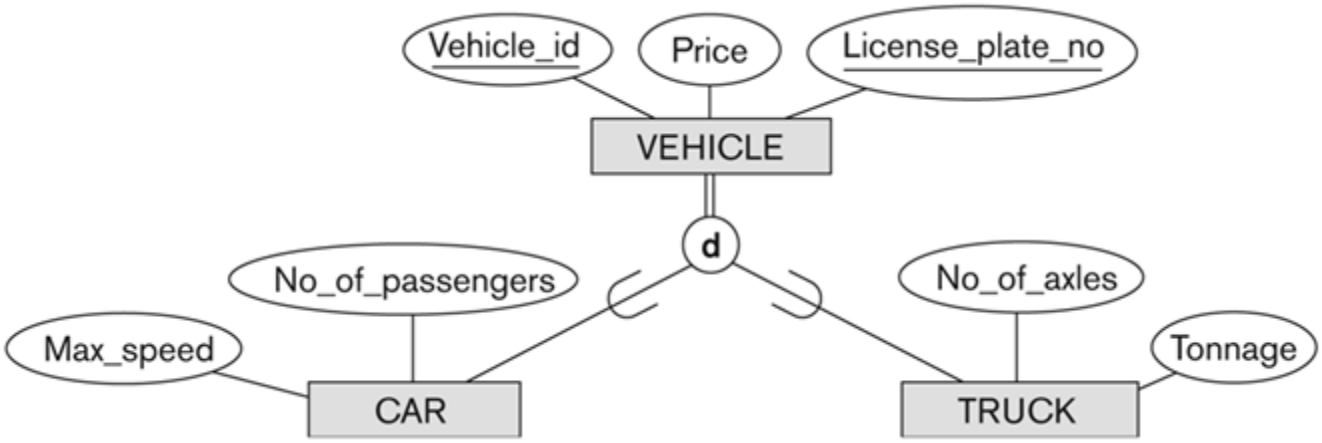
Company Database



Subclass / Superclass

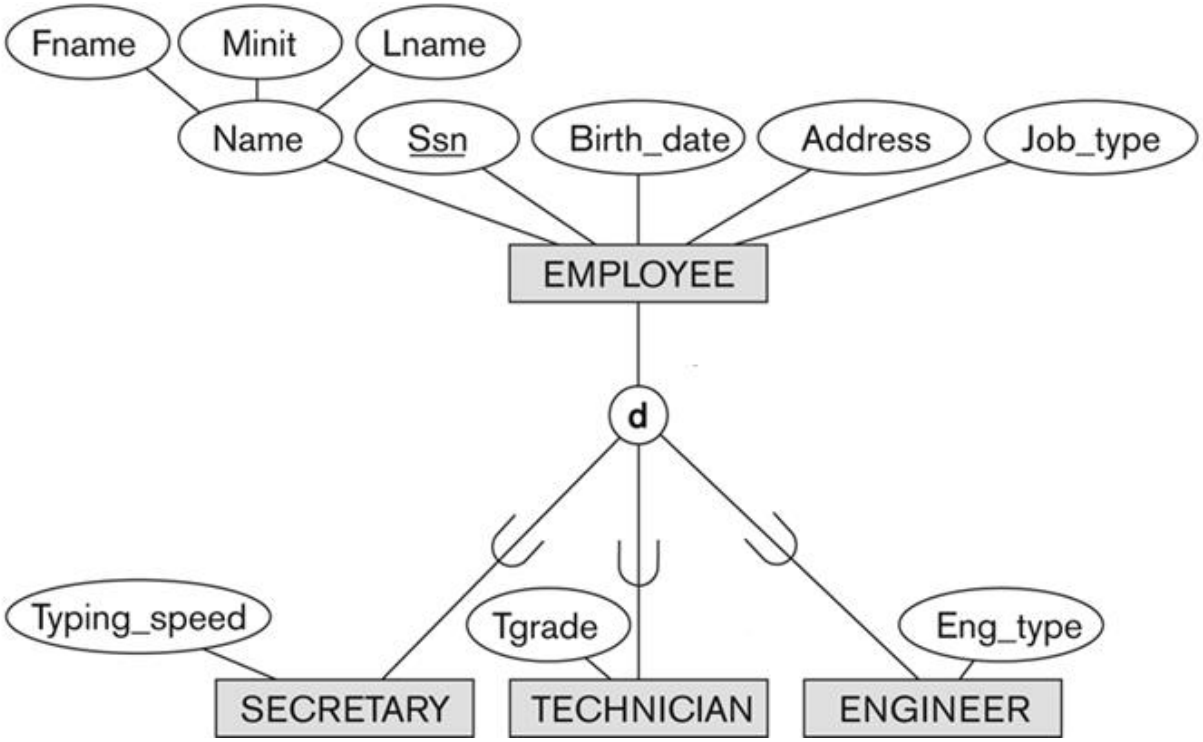
- An entity may have meaningful subgroupings, similar to subclasses of a superclass.
  - EMPLOYEE: Hourly_EMP, Salaried_EMP
  - VEHICLE: Car is a VEHICLE, Truck is a VEHICLE
- Subclasses inherit attributes of the superclass, and they may have additional attributes that the superclass does not have.
  - VEHICLE attributes: vehicle_id, price
  - Car inherits them, in addition: max_speed, no_of_passengers
  - Truck also inherits them, in addition: tonnage

Subclass / Superclass



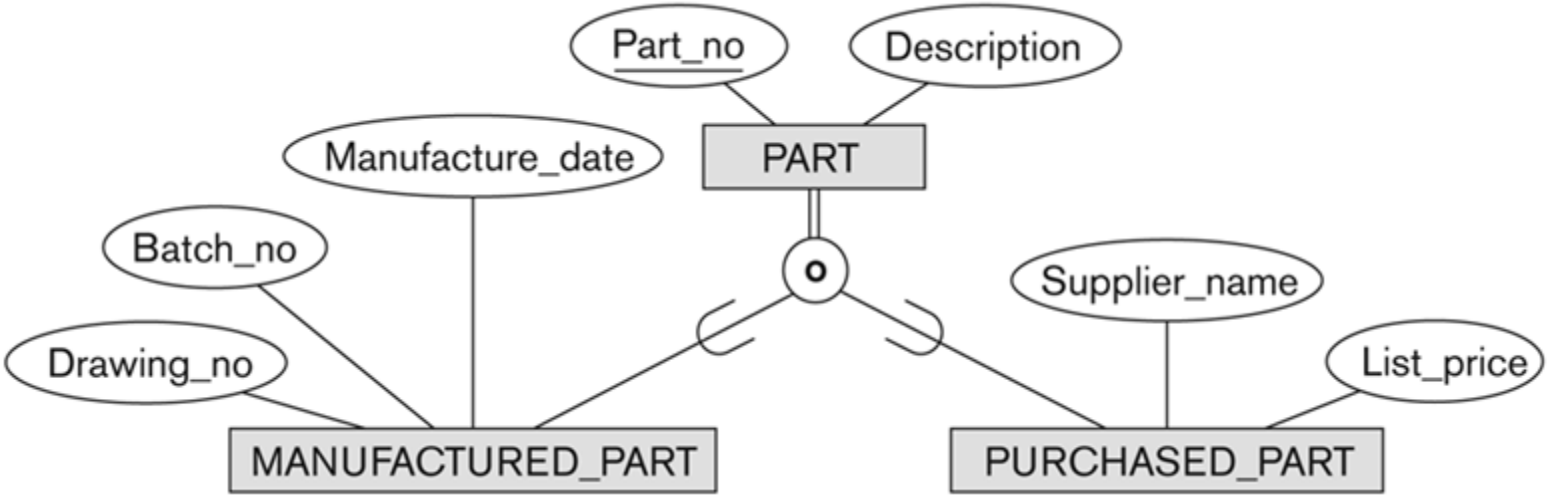
- Arrow-like U's show direction: superclass -> subclass
- What does **d** stand for?
  - **d**: disjoint, **o**: overlapping
- Two lines between VEHICLE and **d** indicate participation constraint

Subclass / Superclass



- Disjoint, but no participation constraint
  - What does this mean?

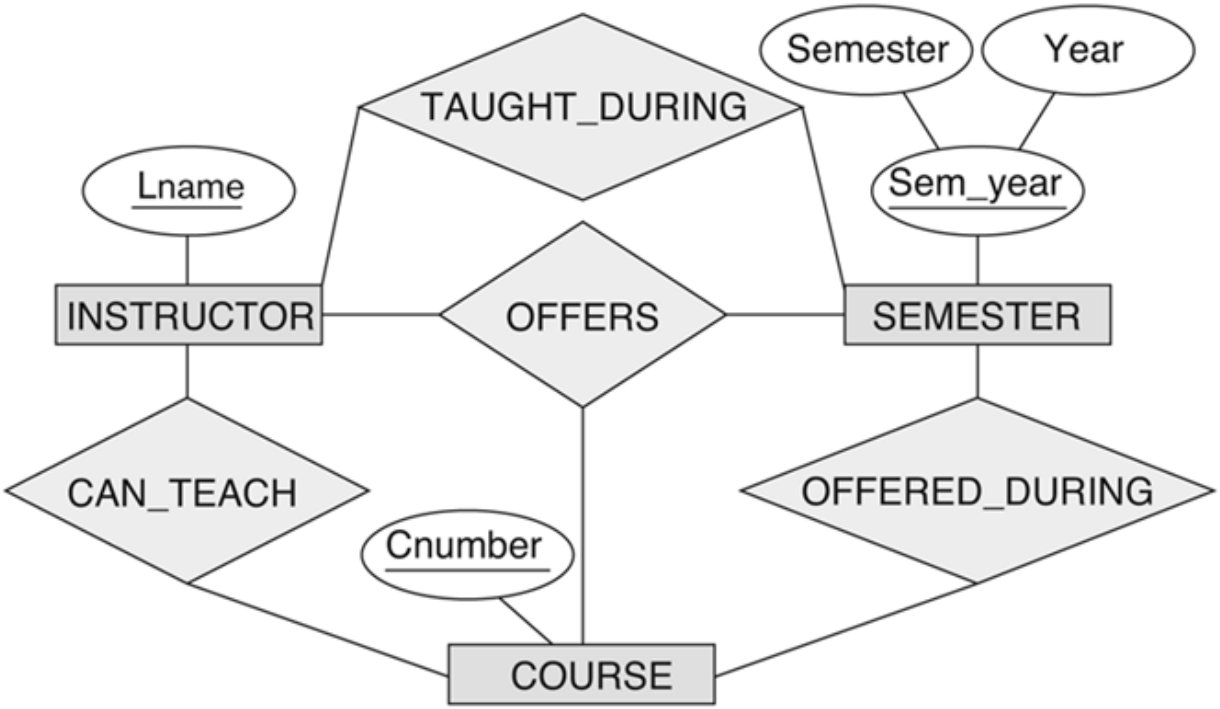
Subclass / Superclass



- Overlapping, with participation constraint
- {Overlapping vs disjoint} and {full vs partial participation} are two separate decisions

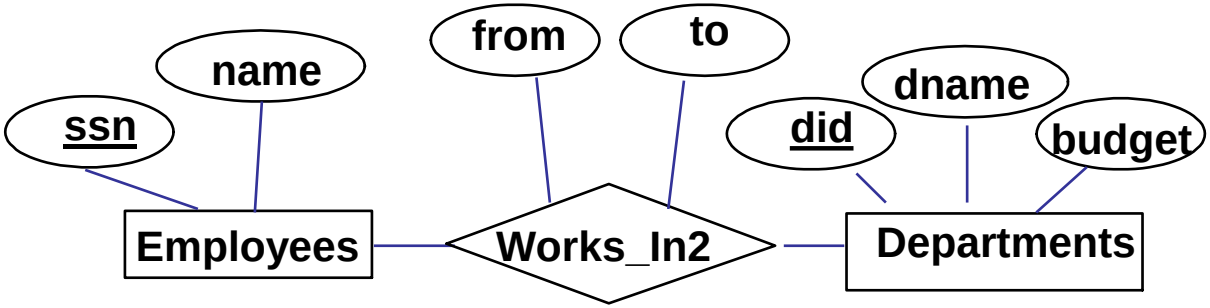
Final Remarks

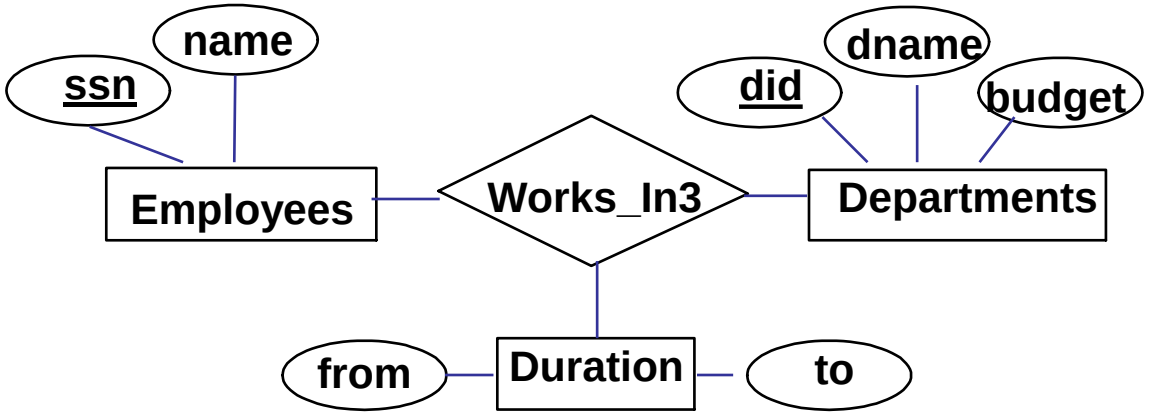
- (1) Avoid redundant relationships.
  - If an existing relationship already captures certain information, don't add new relationships.



Final Remarks

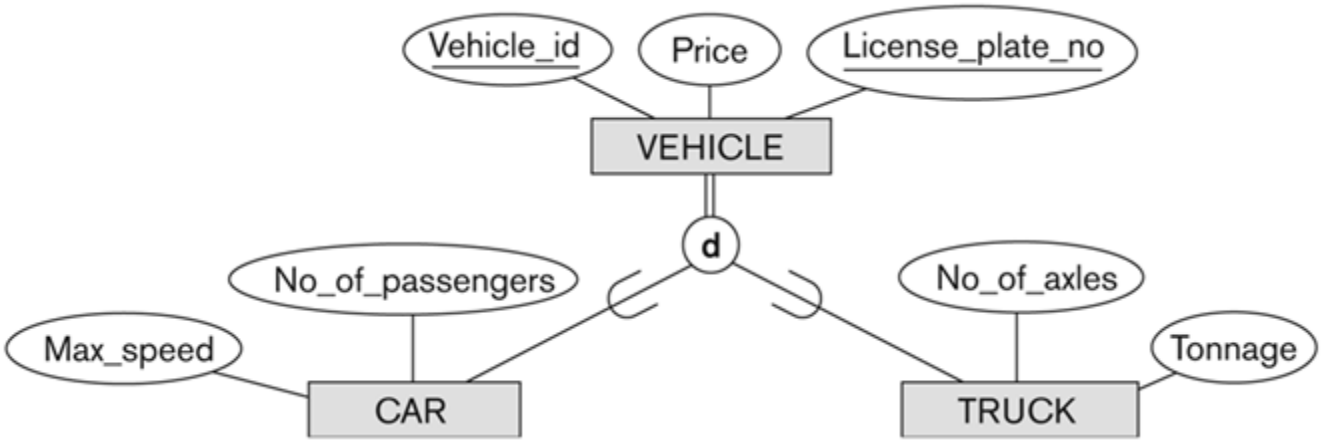
- (2) Should something be an entity or attribute?





Final Remarks

- (3) When do subclasses participate in relationships versus superclasses?



Conclusion

- ER model is popularly used for conceptual design
  - Verbal description of miniworld -> ER diagram
  - Must follow the requirements of the miniworld
  - Better ER model = better database design
- Basic constructs: Entity, relationship, attribute
- Additional constructs: weak entities, identifying relationships, cardinalities, participation constraints, subclass/superclass, ...
- ER model is expressive
- Many variations of ER model exist (w/ varying notation)
  - Actually, the version with subclasses and superclasses is sometimes called “Extended ER (EER)”