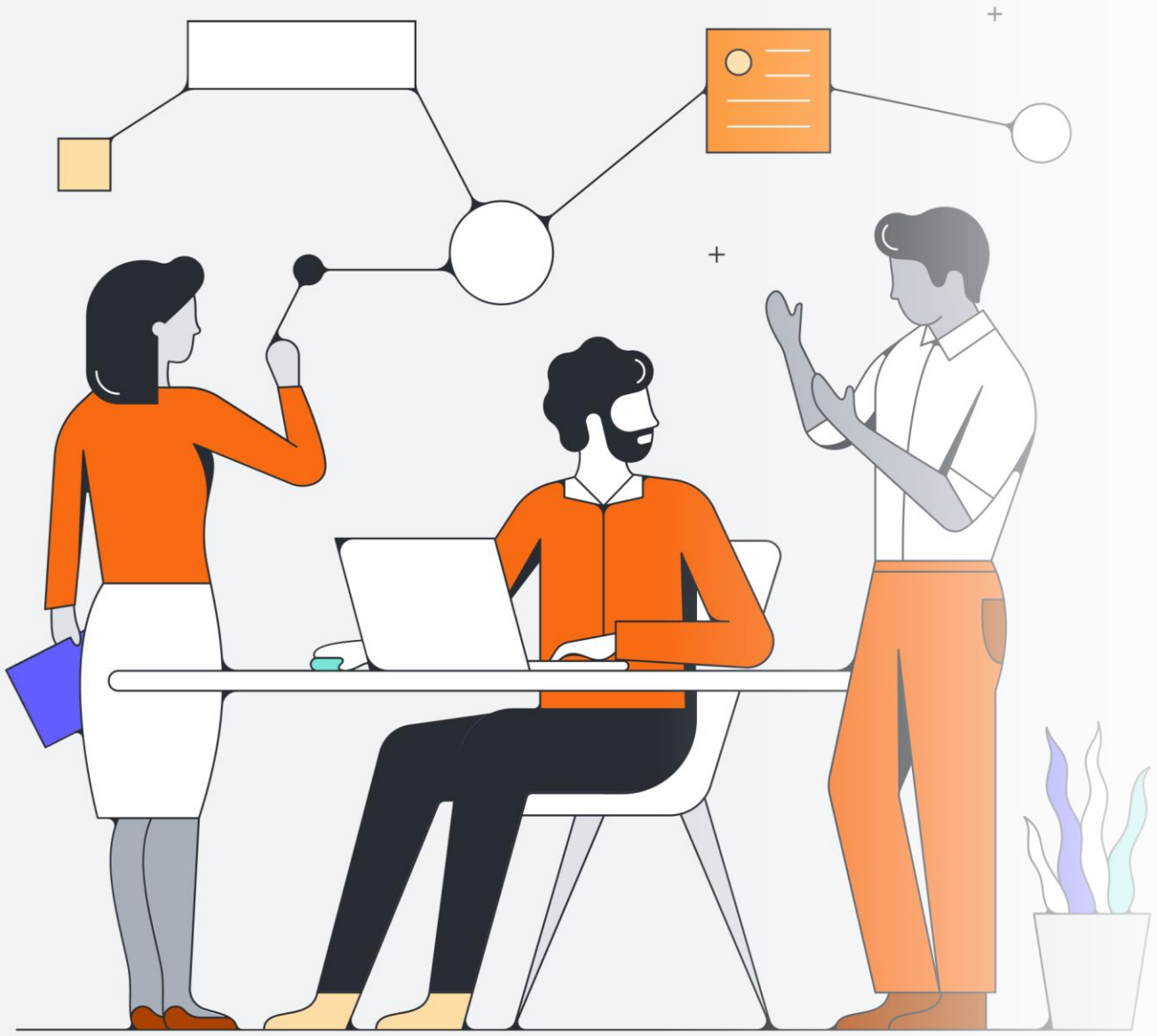


Enhanced Entity Relationship Diagrams

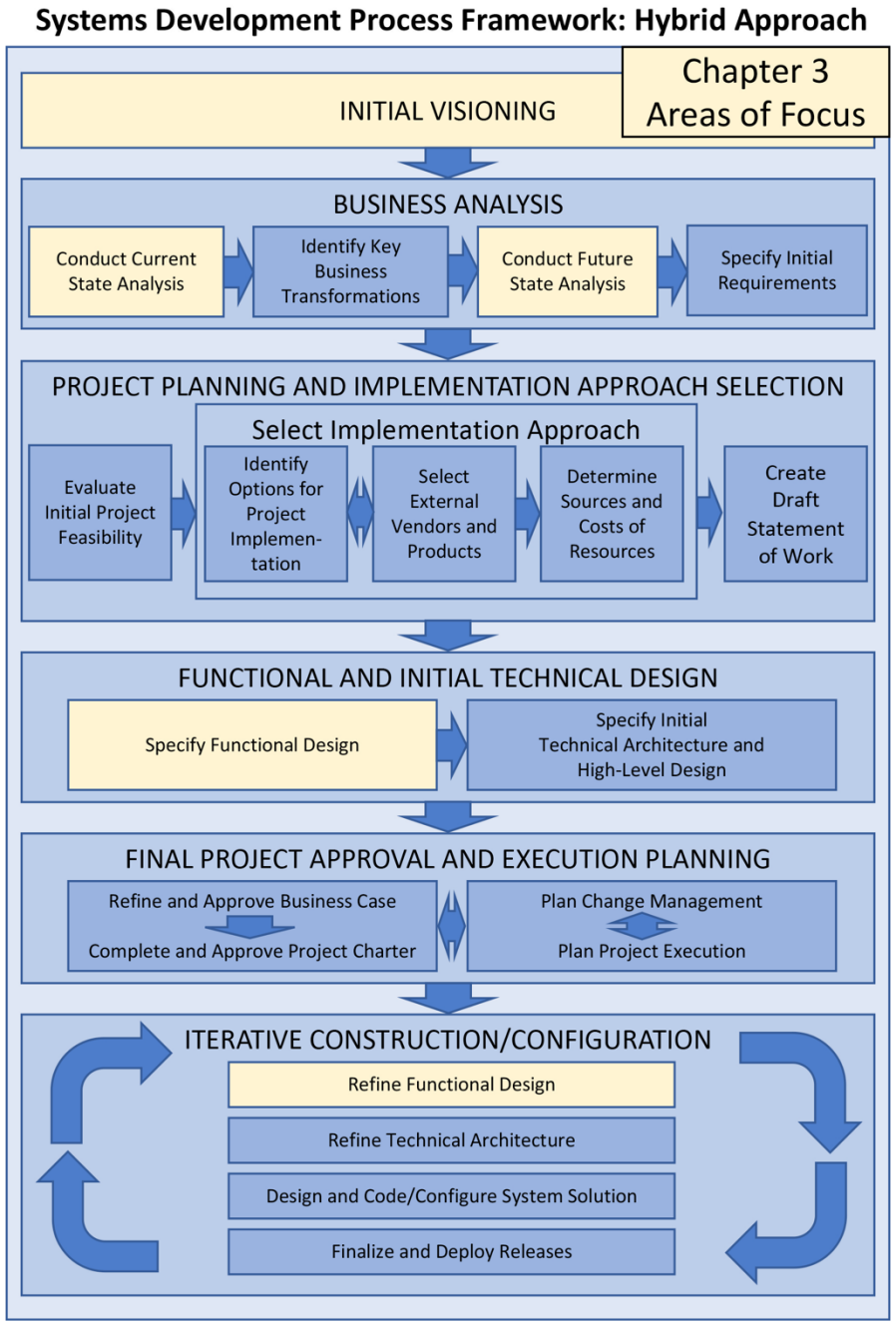


Figure 3-1

CHAPTER 3 AREA OF FOCUS

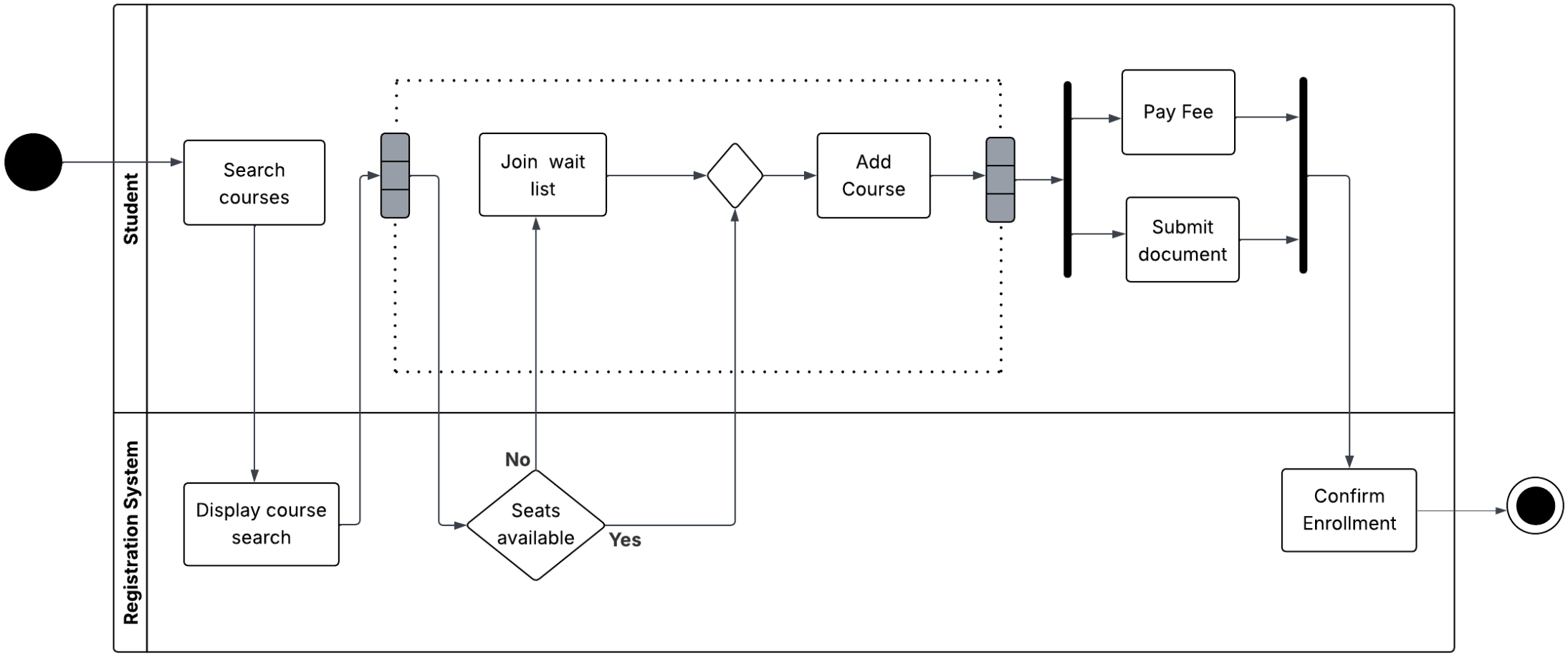
Types of diagrams that the business analyst prepares

- There are two major types of diagrams that the business analyst prepares during the analysis phase of the systems development life cycle.

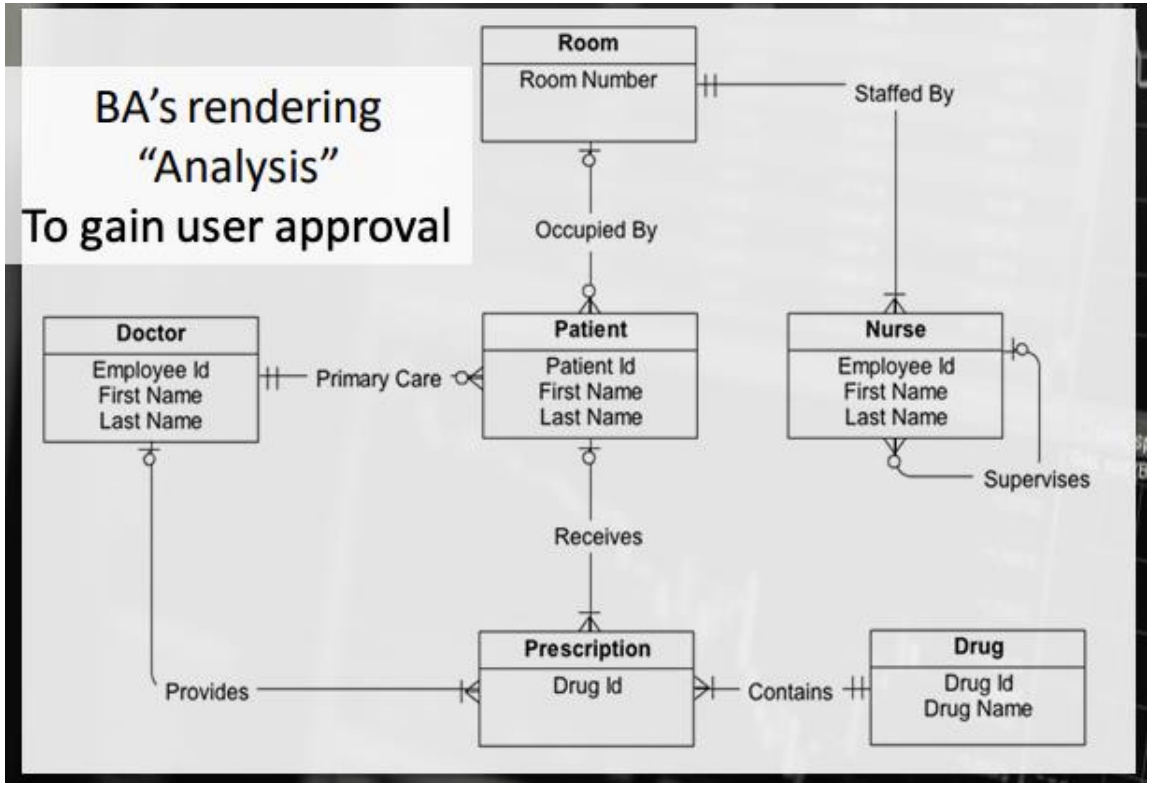
  - Activity diagrams -> model the Processes
  - Entity relationship -> model data that we need to capture as part of those processes.

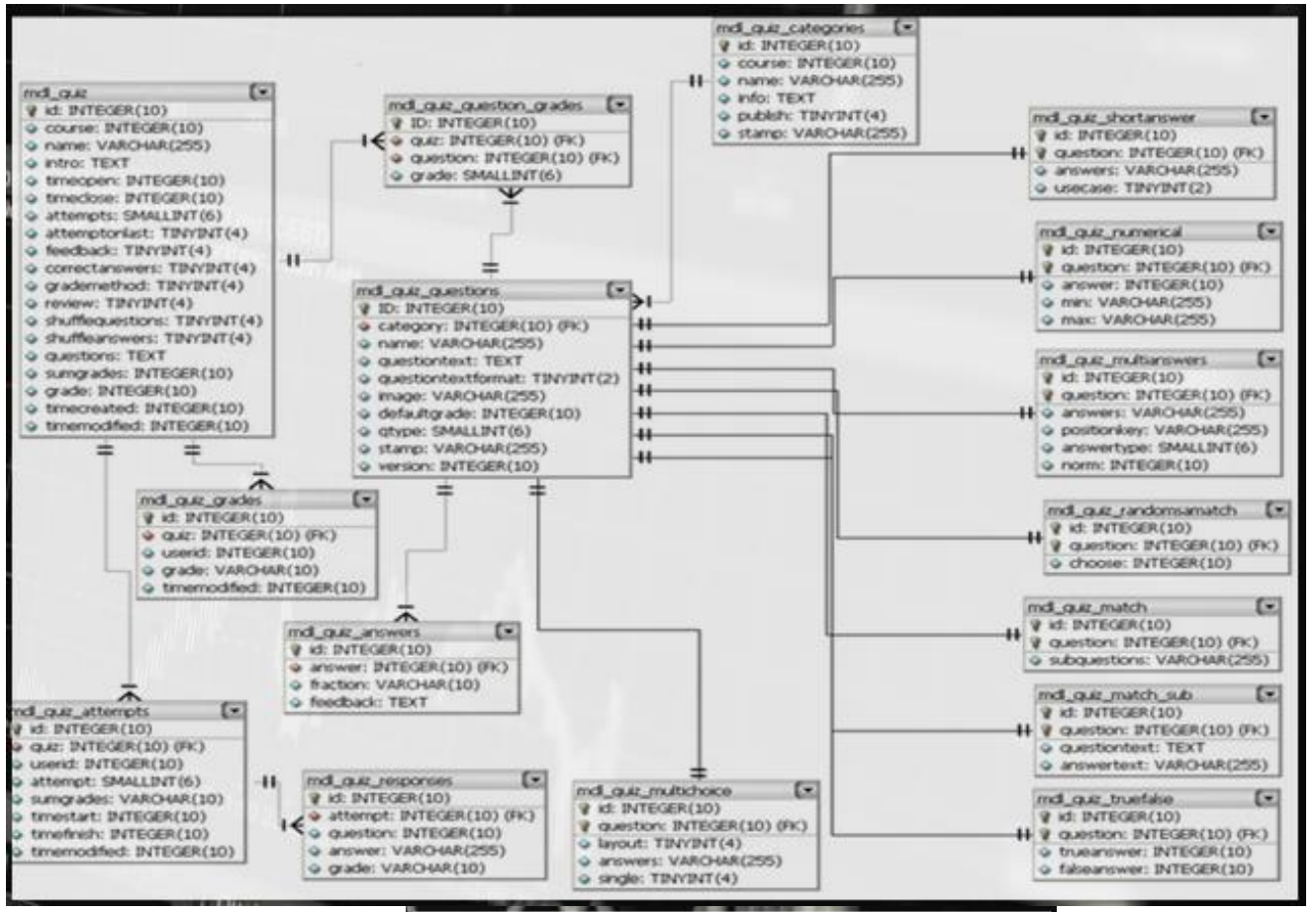
Things we do not capture from activity diagrams

What do we know about the students?
What information should student know about the courses?



Example's of EER Diagram





Designer's drawing
"Design"
To give programmers direction

Requirements Gathering for EER Modeling (COMPANY Case)

The company's organizational structure and activities can be represented using an **Entity–Relationship (EER) model**. The key business elements and their relationships are as follows:

- **Departments**
  - Each department is identified by a *name* and a *number*.
  - Every department is managed by one **Employee**, with the *start date* of the manager's assignment recorded.
  - A department may operate in multiple **locations**.
- **Projects**
  - Each department is responsible for one or more projects.
  - A project is defined by a *unique name* and *number*, and is associated with a single *location*.
- **Employees**
  - Employees are associated with a single department but may participate in multiple projects.
  - For each project assignment, we track the *number of hours per week* the employee contributes.
  - Each employee has a designated *direct supervisor*.
  - Employee details include *SSN, name, salary, sex,* and *birthdate*.
- **Dependents**
  - Employees may have zero or more dependents.
  - For each dependent, we record *name, sex, birthdate,* and the *relationship* to the employee.

Entities and Entity Sets

- **Entity** : A Physical object, person, community, event, or other such concept that an organization is sufficiently interested in to include it in the domain model
  - *Example:* A specific EMPLOYEE , a specific DEPARTMENT

- **Entity Instance:** An individual entity that exist in real world
  - Example: an employee called John Smith

- **Entity type :** A class of entity instances that belong to the same class of thigs. Often refer to as entity
  - *Example:*
    - EMPLOYEE – set of all employees in company.
    - DEPARTMENT – set of all departments in the company.
    - PROJECTS – set of all projects.

Attributes

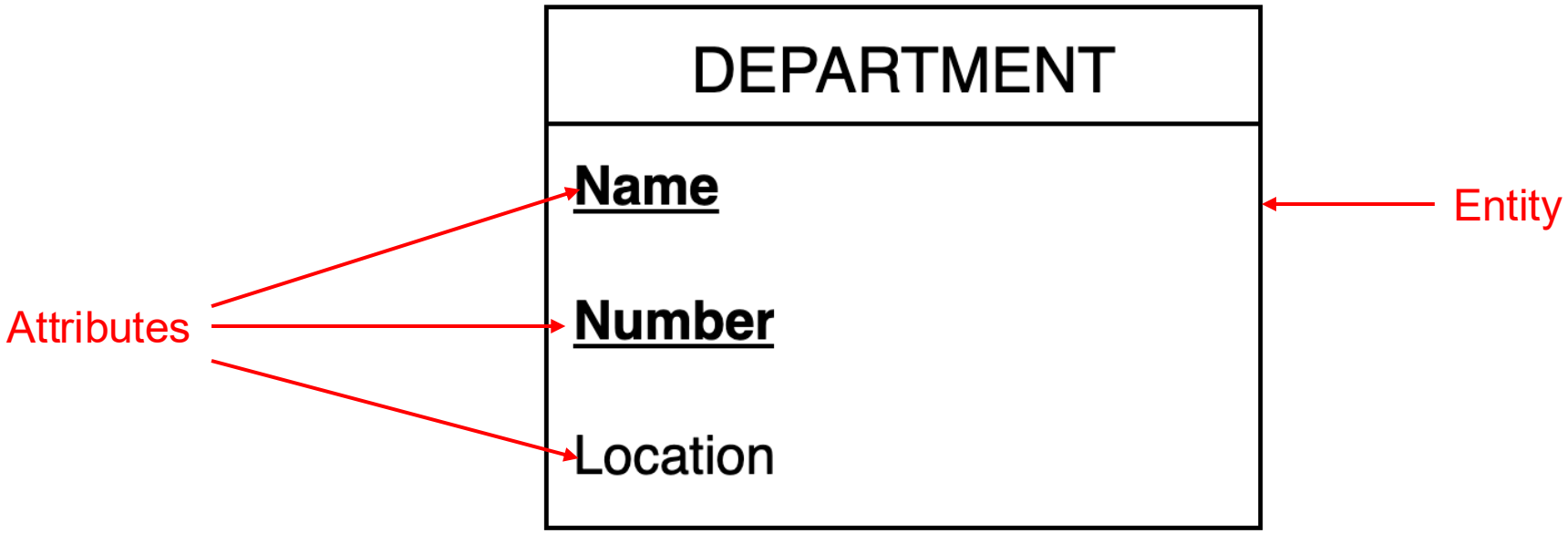
- Attributes are properties used to describe an entity.
  - For example EMPLOYEE entity set may have the attributes Name, Department, Birth_date, SSN, Sex, Salary, Address

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

- Null Values
  - In some cases a particular entity may not have an applicable value for an attribute.
  - For example, CollegeDegrees attribute applies only to persons with college degrees. For such situations, a special value called null is created.
  - Null is used if the attribute value is unknown or not applicable.

EER Notations for Entities and Attributes



Types of 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:
    - 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, PreviousDegrees of a EMPLOEE.
- 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 EMPLOYEE is a composite multi-valued attribute
  - Multiple PreviousDegrees values can exist
  - Each has four subcomponent attributes:
    - College, Year, Degree, Field
- **Derived attributes**
  - Attributes that can be computed from other attributes
  - E.g. *age*, given date of birth

Keys (Identifier)

- A key is a minimal set of attributes that uniquely identifies an entity and separate entity instances from each other
  - For example, SSN of EMPLOYEE.

- A key attribute may be composite.

- An entity type may have more than one key.
  - The PROJECT entity type may have two keys:
    - Project Name
    - Project Number

- Each key is underlined

Candidate key attribute

Any attribute that contains a unique value in each row of the table

Staff ID	Staff name	Staff title	Subjects	Contact number
01	Bob	Lecturer	Physics, Chemistry	01452852
02	Dave	Assistant Lecturer	Media, Film	04785254
03	Lucinda	Senior Lecturer	History, Archaeology	06985354
04	Heather	Lecturer	Math, Computer Science	02874852
05	John	Assistant Lecturer	Illustration, Graphic Design	03658742

Composite key attribute

A key composed of two or more attributes to form a unique value in each new row

Staff ID	Staff name	Staff title	Subjects	Contact number
01	Bob	Lecturer	Physics, Chemistry	01452852
02	Dave	Assistant Lecturer	Media, Film	04785254
03	Lucinda	Senior Lecturer	History, Archaeology	06985354
04	Heather	Lecturer	Math, Computer Science	02874852
05	John	Assistant Lecturer	Illustration, Graphic Design	03658742

Primary Key

- Uniquely identify every row in a table

Example: *CustomerID, as well as Phone, are candidate keys for relation CUSTOMER but CustomerID can be chosen as the primary key (only one out of many candidate keys).*

CUSTOMER Table

CustomerID **(PK)**	Name	Phone	Email
101	Ada	9865278251	ada@example.com
102	Alan	9655470231	alan@example.com
103	Grace	7514290359	grace@example.com
110	Grace	8564103258	grace.1998@example.com

Primary Key

Foreign Key

A foreign key is a column or group of columns in a relational database table that provides a link between data in two tables.

CUSTOMER Table

CustomerID **(PK)**	Name	Phone	Email
101	Ada	9865278251	ada@example.com
102	Alan	9655470231	alan@example.com
103	Grace	7514290359	grace@example.com
110	Grace	8564103258	grace.1998@example.com

Primary Key for CUSTOMER table

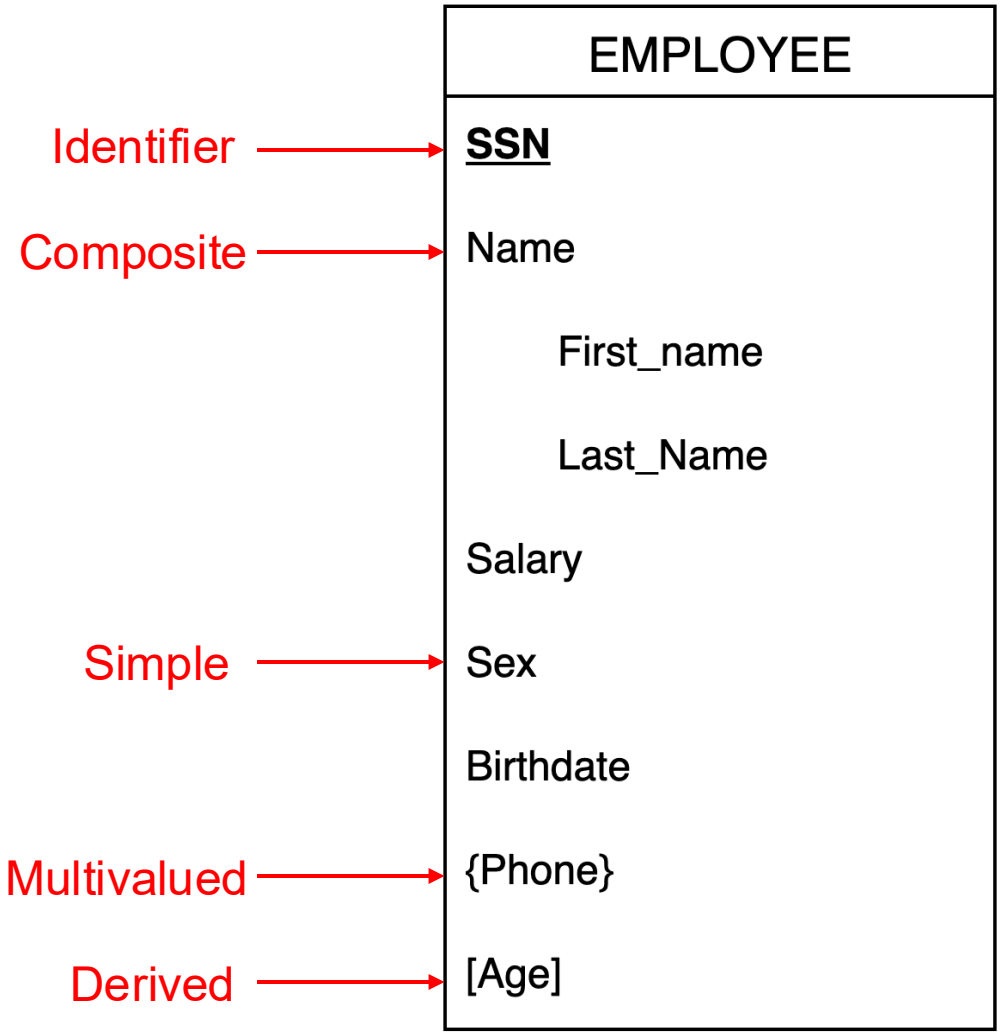
ORDER Table

OrderID **(PK)**	CustomerID (FK to CUSTOMER)	OrderDate	TotalAmount
5001	101	2025-09-01	129.99
5002	101	2025-09-03	59.00
5003	102	2025-09-04	499.00
5004	102	2025-09-06	19.50
5005	110	2025-09-07	75.00

Primary Key for ORDER table

Foreign Key referencing CUSTOMER table

EER Notations for different types of attributes



Exercise :

- Based on the requirements, we can identify four initial entity types in the company
  - DEPARTMENT
  - PROJECT
  - EMPLOYEE
  - DEPENDENT
- The initial attributes can be derived from the requirements description
- Show the initial design. For each entity set show how attributes are connected using EER diagram symbols

Requirements Gathering for EER Modeling (COMPANY Case)

The company's organizational structure and activities can be represented using an **Entity–Relationship (ER) model**. The key business elements and their relationships are as follows:

- **Departments**
  - Each department is identified by a *name* and a *number*.
  - Every department is managed by one **Employee**, with the *start date* of the manager's assignment recorded.
  - A department may operate in multiple **locations**.
- **Projects**
  - Each department is responsible for one or more projects.
  - A project is defined by a *unique name* and *number*, and is associated with a single *location*.
- **Employees**
  - Employees are associated with a single department but may participate in multiple projects.
  - For each project assignment, we track the *number of hours per week* the employee contributes.
  - Each employee has a designated *direct supervisor*.
  - Employee details include *SSN, name, salary, sex,* and *birthdate*.
- **Dependents**
  - Employees may have zero or more dependents.
  - For each dependent, we record *name, sex, birthdate,* and the *relationship* to the employee.

Answers

EMPLOYEE
SSN
Name
First_name
Last_Name
Salary
Sex
Birthdate

DEPARTMENT
Name
Number
{Locations}

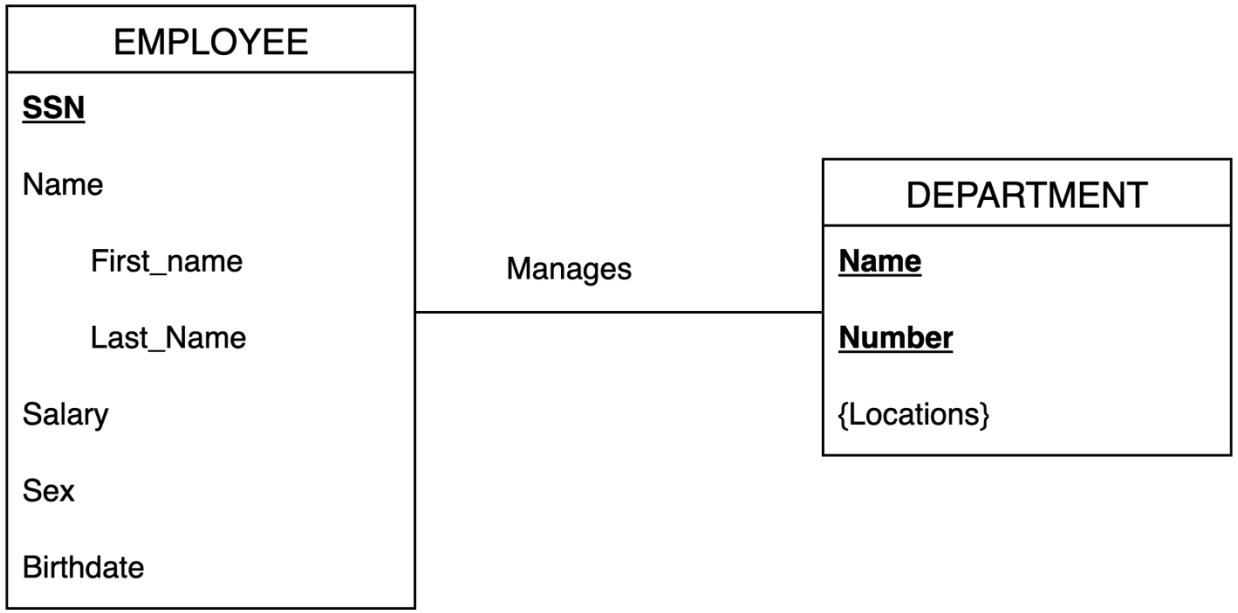
Maybe you have add manager as attribute for DEPARTMENT. These will be represented as relationships later.

DEPENDENT
Name
Sex
Birth_date
Relationship

PROJECT
Name
Number
Location

Relationships

- Captures various associations between related entities
- For example, EMPLOYEE Franklin Wong *manages* the Research DEPARTMENT

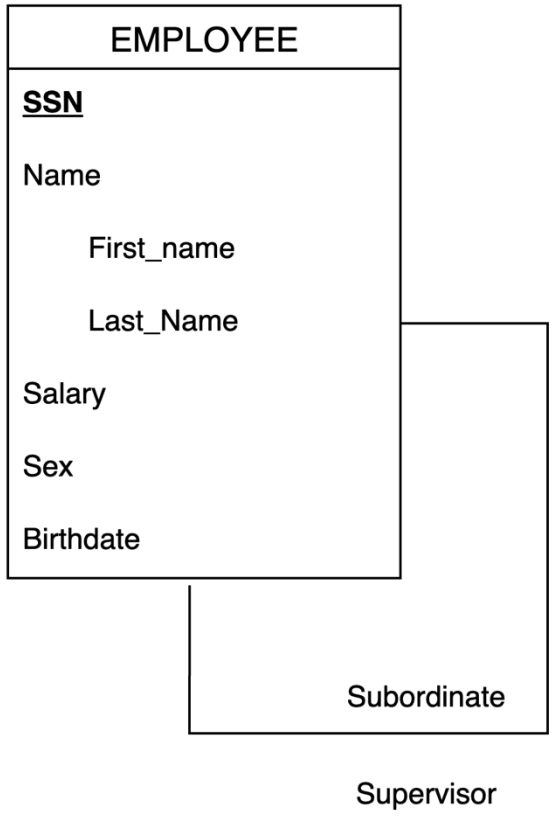


Entity set connected to the
participating entity set via straight lines

Degree of a Relationship

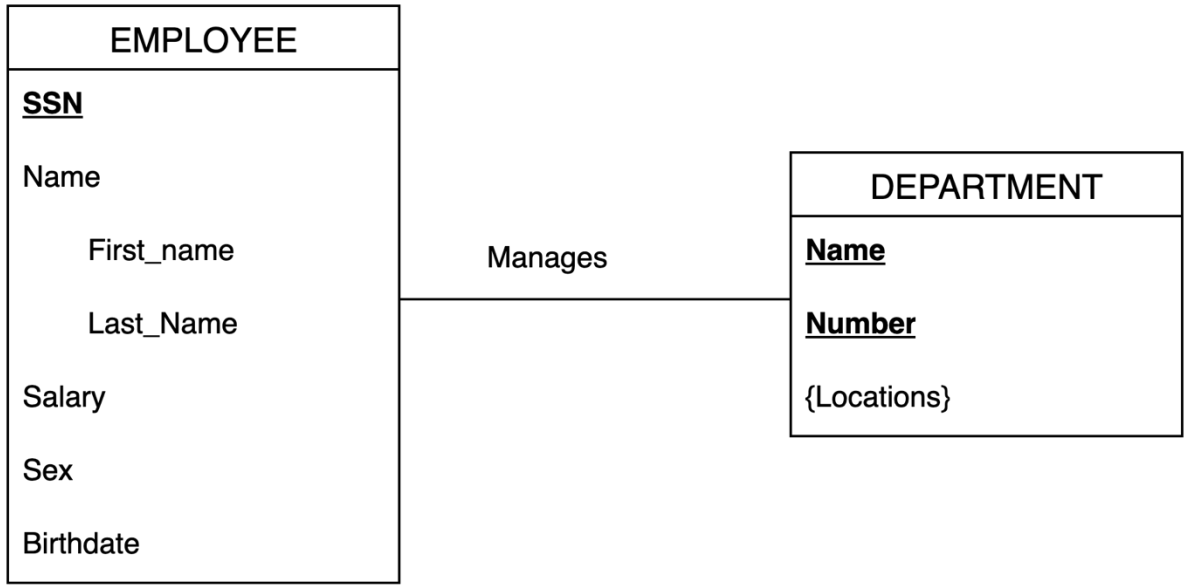
- Refers to number of entity types that participate in a relationship set.

Unary Relationship



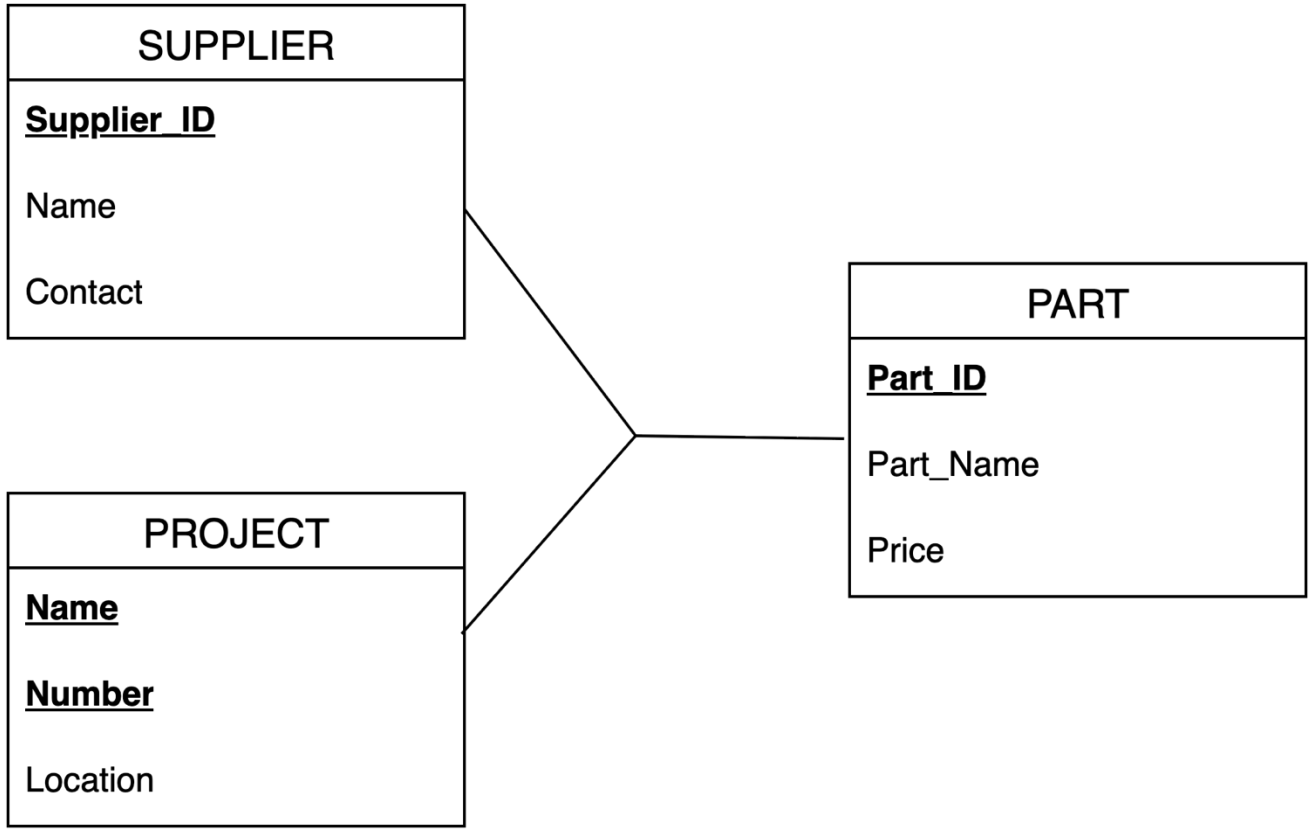
- An relationship type whose with the same participating entity type in distinct roles
- For example, in our company, Employee may supervise other Employees.

Binary Relationship



- Relationship between two distinct entity types.
- A binary relationship is the most common type in ER/EER models.
- For example, EMPLOYEE *manages* the Research DEPARTMENT

Ternary Relationship



- A ternary relationship involves three different entity sets at the same time

Example:

- A supplier may supply a certain part **for a specific project**.
- If you model it as separate binaries (e.g., Supplier → Part, Supplier → Project, Part → Project), you lose information about the fact that *this supplier provided this part for this project*.

Exercise :

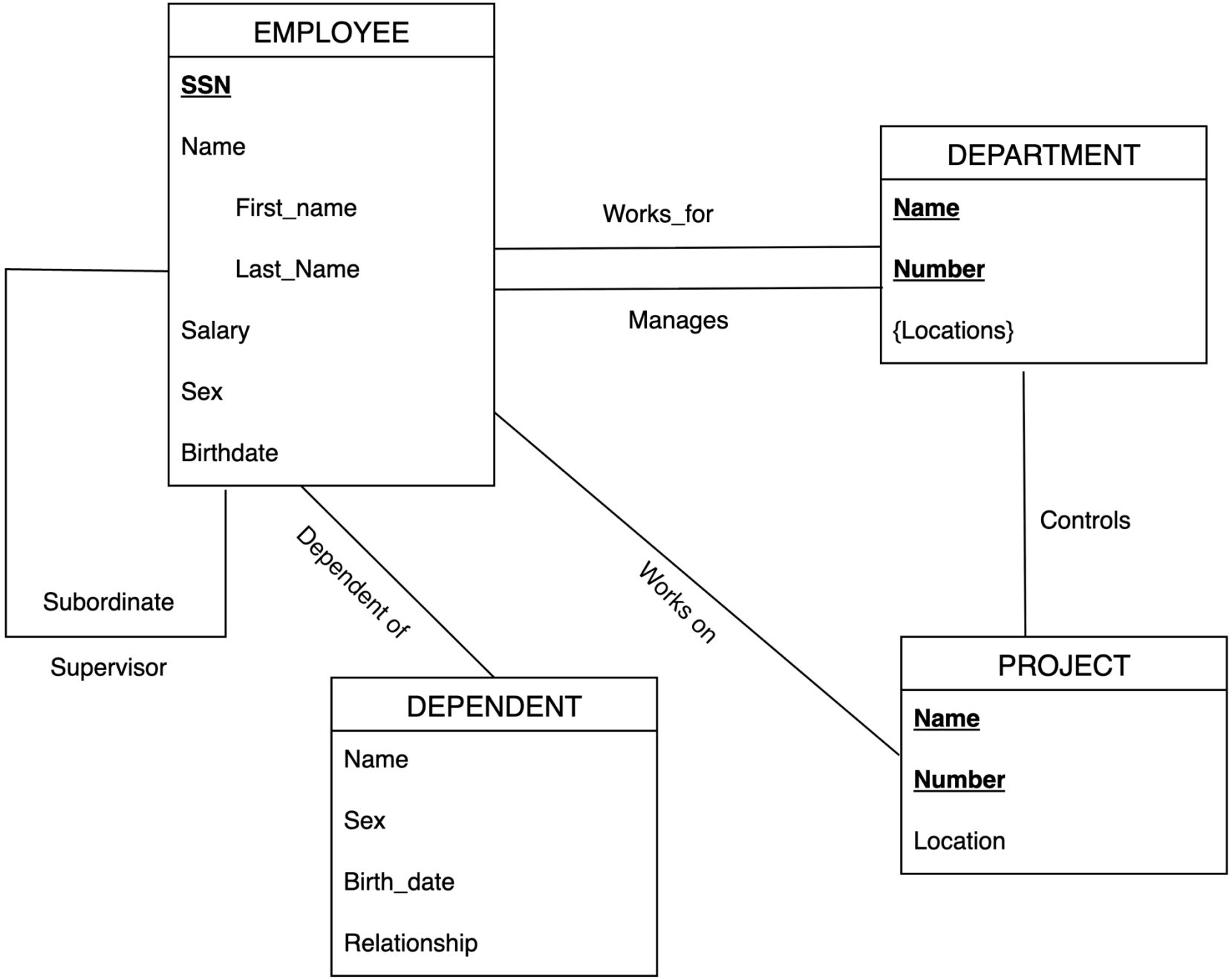
- Based on the requirements, Identify relationships between entity types in the company:
  - DEPARTMENT
  - PROJECT
  - EMPLOYEE
  - DEPENDENT

Requirements Gathering for EER Modeling (COMPANY Case)

The company's organizational structure and activities can be represented using an **Entity–Relationship (ER) model**. The key business elements and their relationships are as follows:

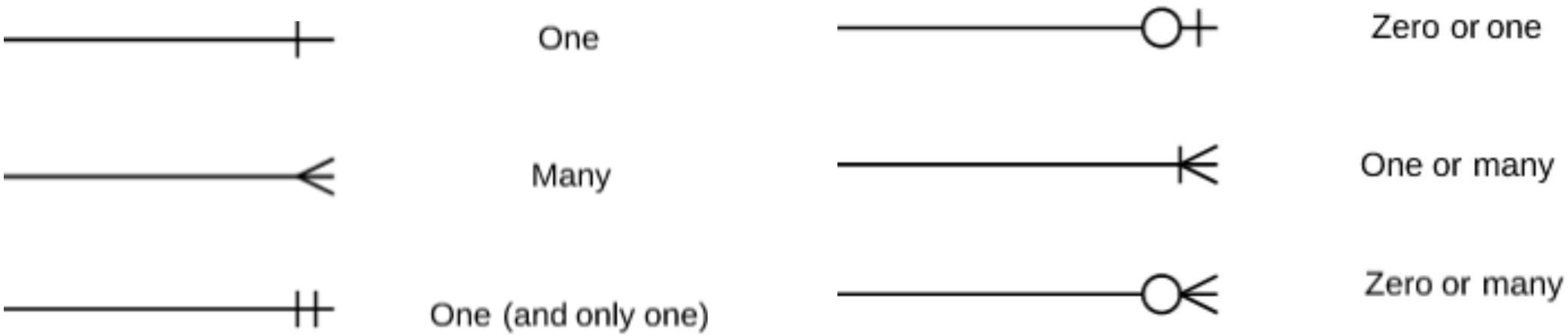
- **Departments**
  - Each department is identified by a *name* and a *number*.
  - Every department is managed by one **Employee**, with the *start date* of the manager's assignment recorded.
  - A department may operate in multiple **locations**.
- **Projects**
  - Each department is responsible for one or more projects.
  - A project is defined by a *unique name* and *number*, and is associated with a single *location*.
- **Employees**
  - Employees are associated with a single department but may participate in multiple projects.
  - For each project assignment, we track the *number of hours per week* the employee contributes.
  - Each employee has a designated *direct supervisor*.
  - Employee details include *SSN, name, salary, sex,* and *birthdate*.
- **Dependents**
  - Employees may have zero or more dependents.
  - For each dependent, we record *name, sex, birthdate,* and the *relationship* to the employee.

Answers


EMPLOYEE
SSN
Name
First_name
Last_Name
Salary
Sex
Birthdate
DEPARTMENT
Name
Number
{Locations}
Works_for
Manages
Controls
Subordinate
Supervisor
Dependent of
Works on
DEPENDENT
Name
Sex
Birth_date
Relationship
PROJECT
Name
Number
Location

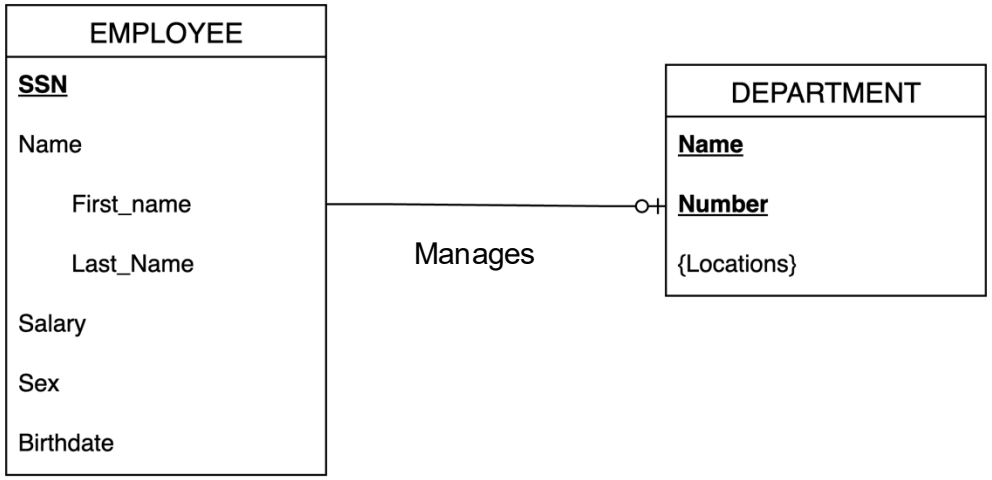
Cardinality

- Specifies the number of entity instances associated with another entity instance in the relationship

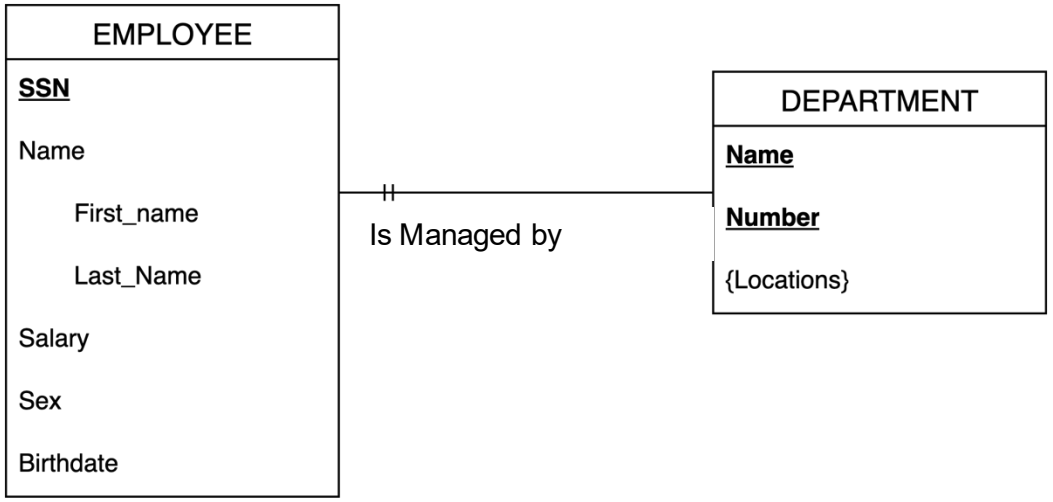


One-to-One Relationship:

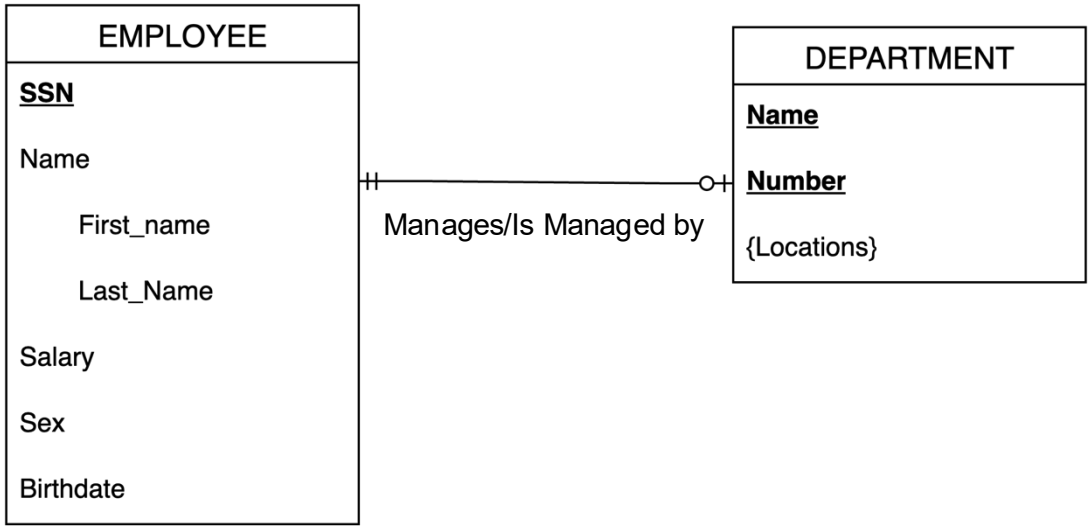
- Each instance of one entity maps to a single instance of the other.



Each employee can manage zero or one department



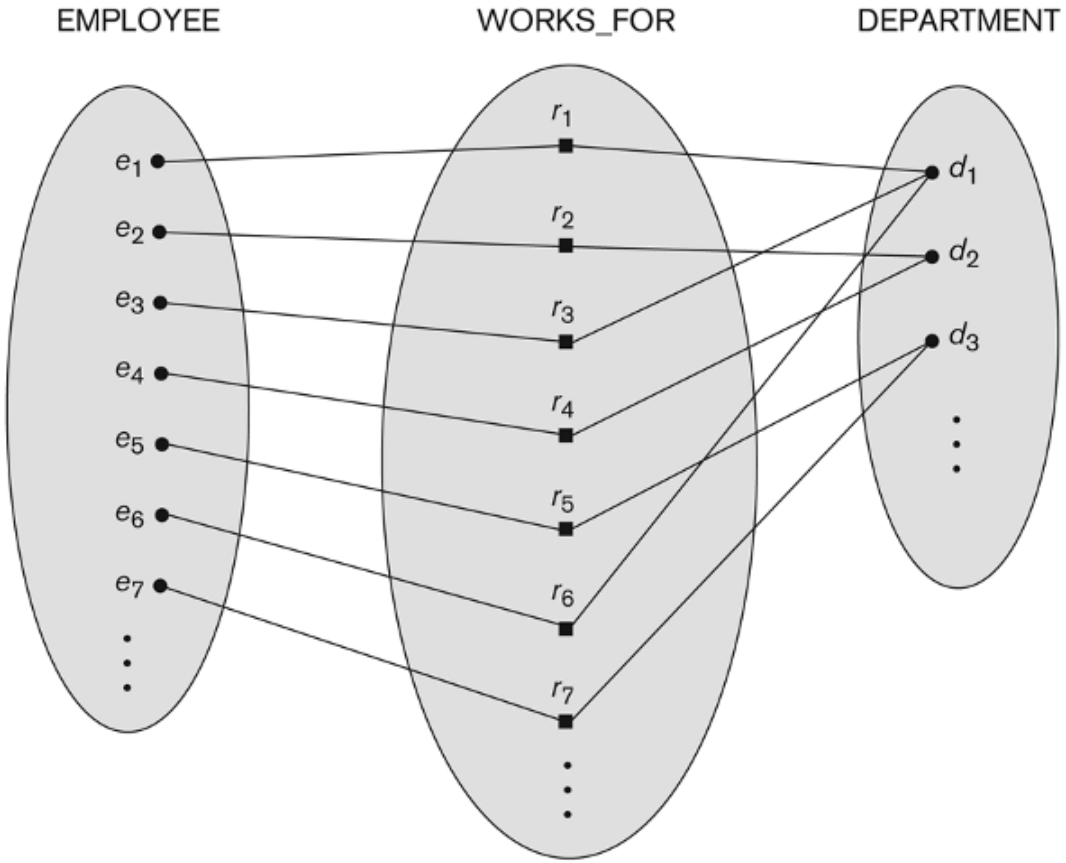
Each department is managed by exactly one manager

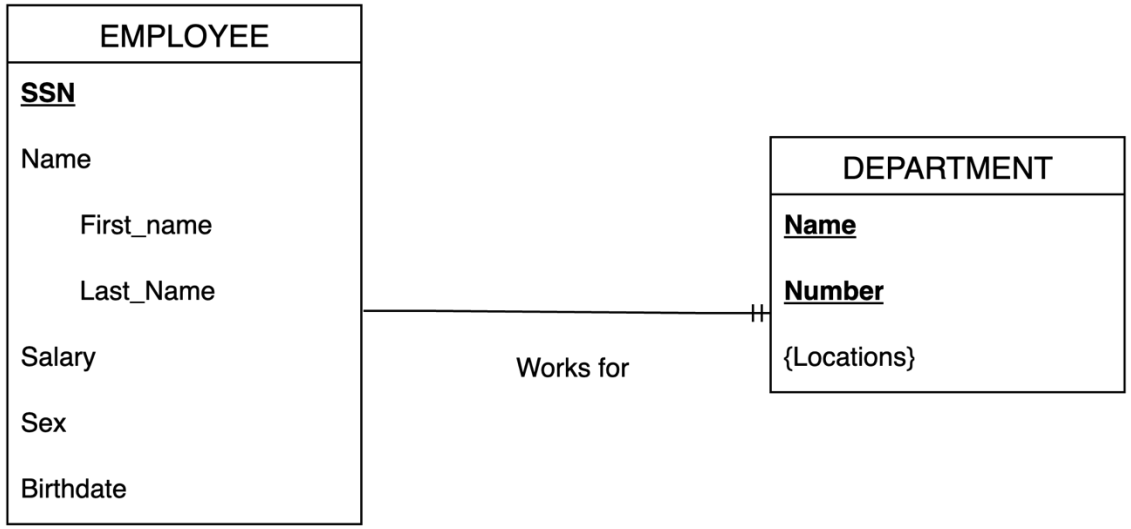


Each employee can manage zero or one department and Each department is managed by exactly one manager

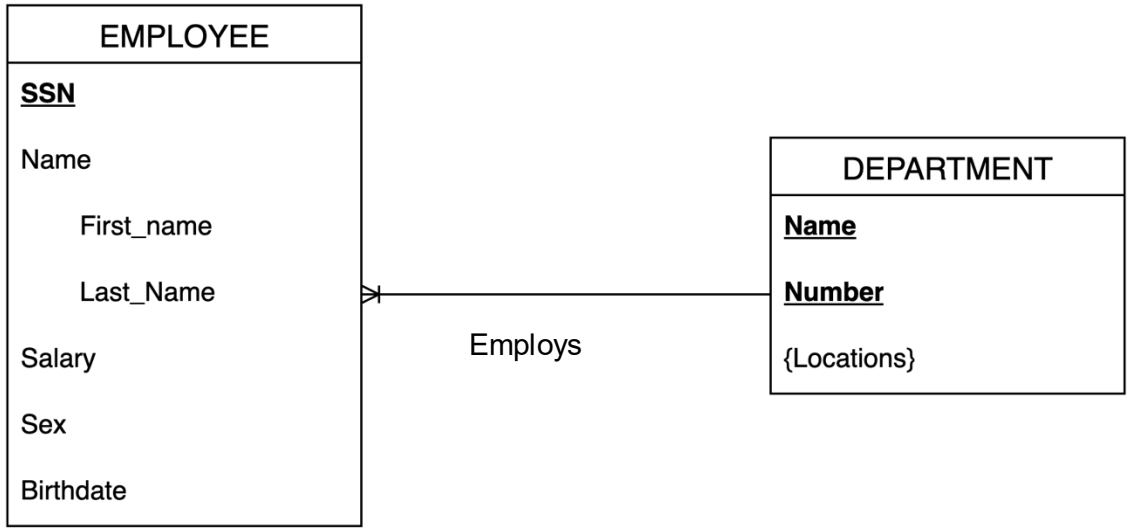
Many-to-One Relationship:

- Many instances of Entity B relate to one instance of Entity A, but each occurrence of Entity A is associated with only one occurrence of Entity B.
- For example, one DEPARTMENT can have many EMPLOYEES, but each employee works in only one department.

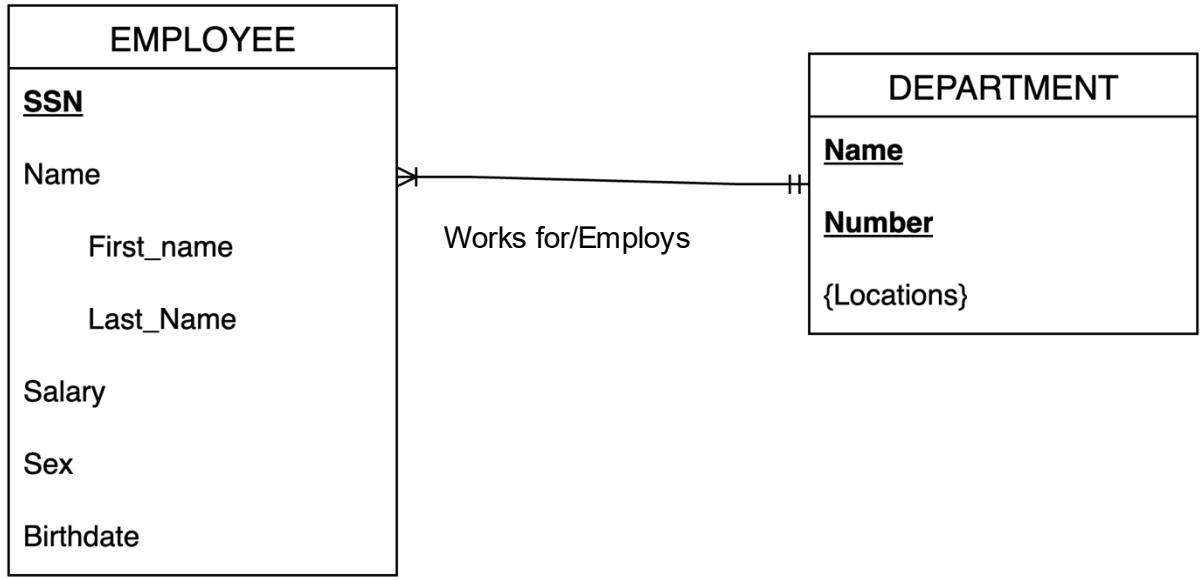




Each employee can works for exactly one department



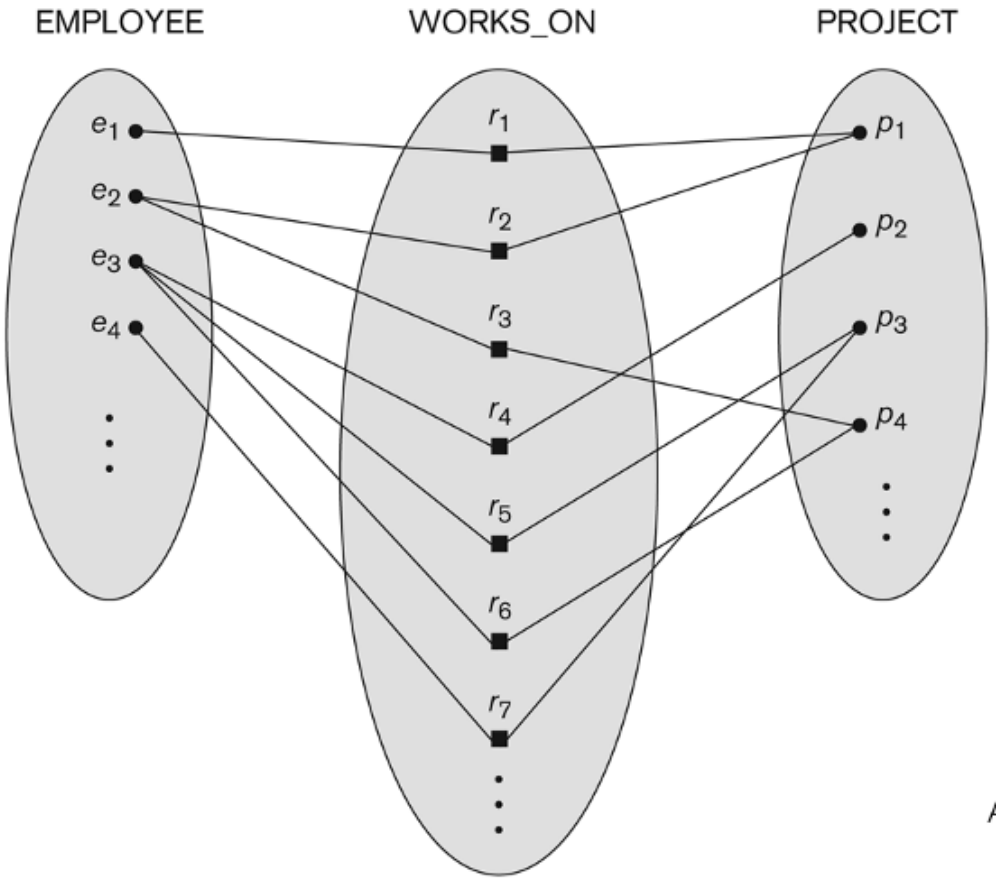
Each department can have one or many employees

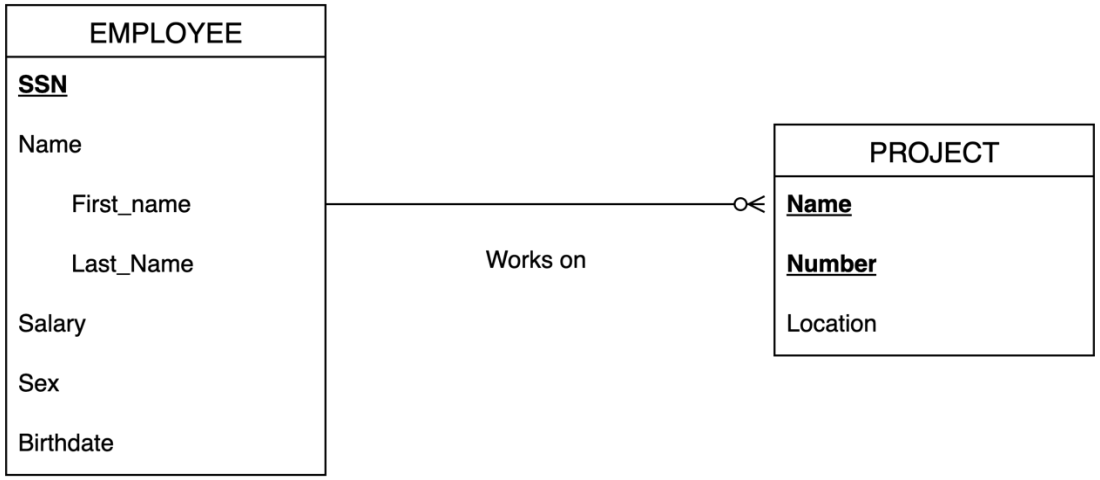


Each employee can works for exactly one department and each department can have one or many employees

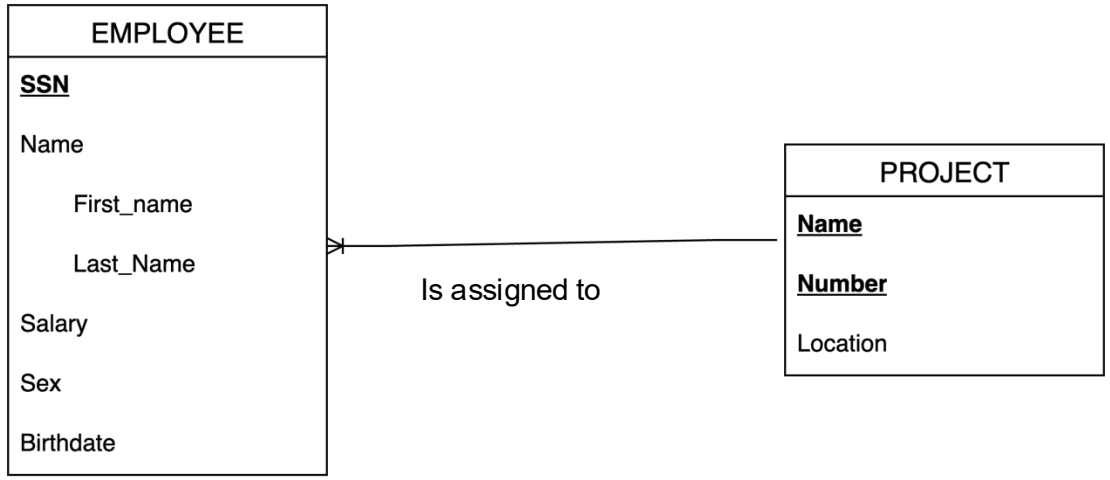
Many-to-Many Relationship:

- Each instance of A can relate to multiple instances of B, and vice versa
- For example, consider EMPLOYEE and PROJECT entities: an employee can work on multiple projects, and each project can have many employees.

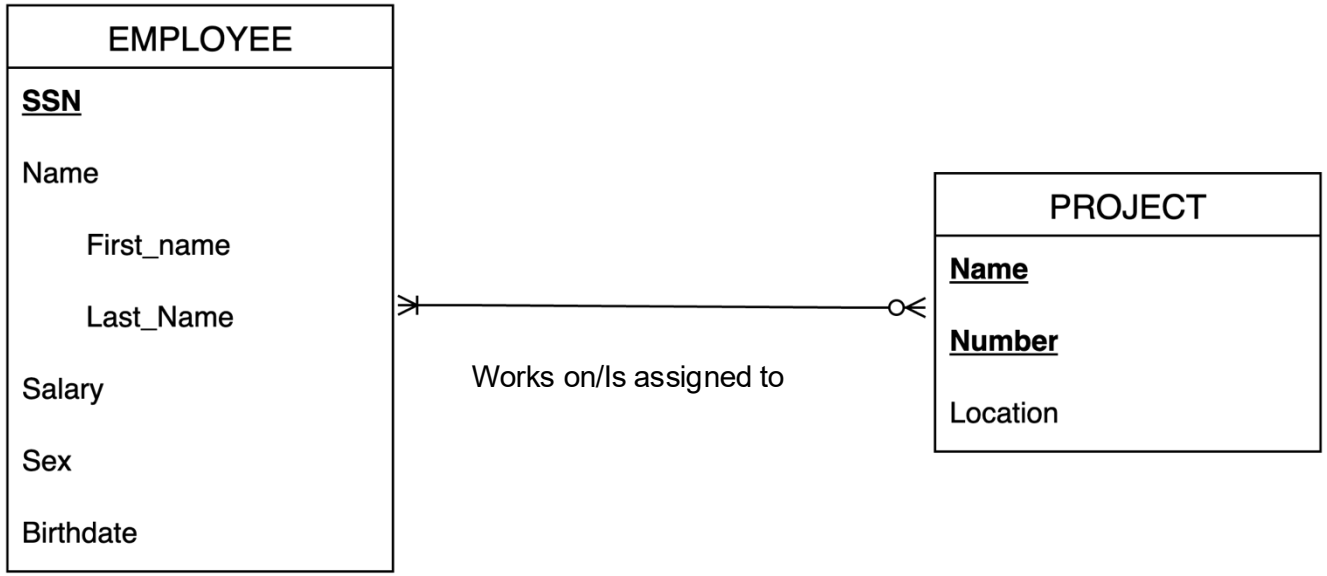




Each employee can works on zero or more projects



Each project can have one or many employees



Each employee can works on zero or more projects and each project can have one or many employees

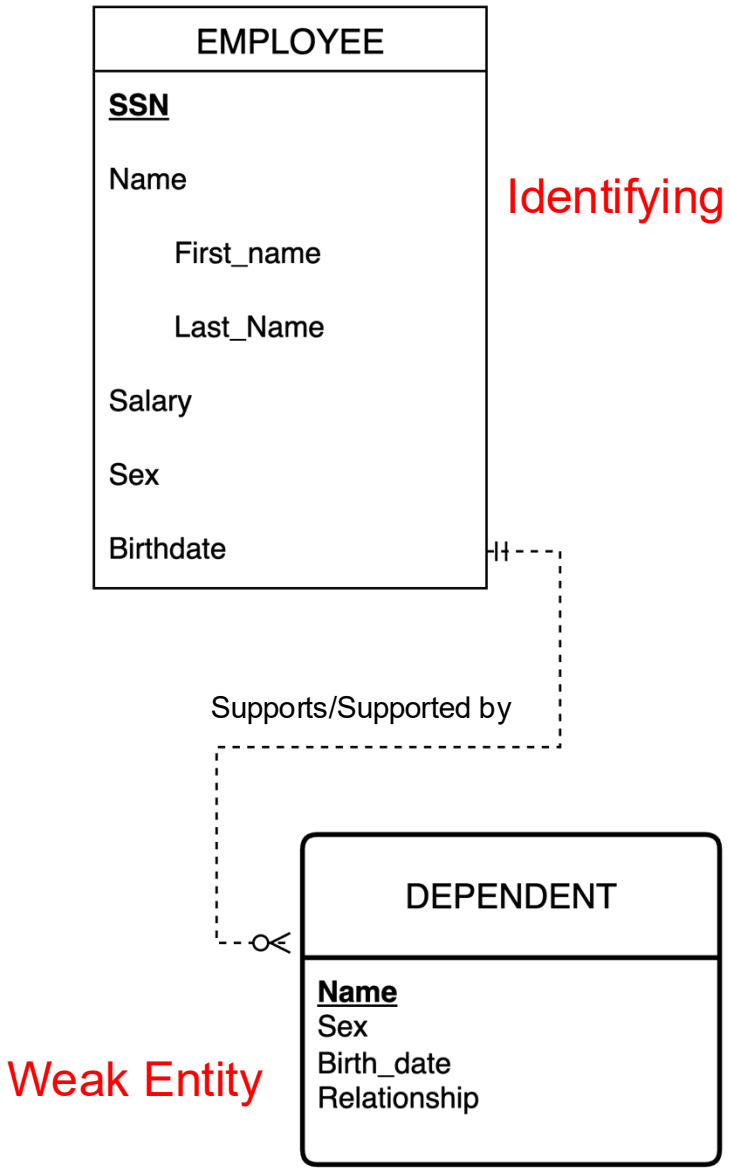
Weak Entity

- A weak entity is an entity that is existence-dependent on some other entity. By contrast, a regular entity (or “a strong entity”) is an entity which is not weak.
- The existence of a weak entity set depends on the existence of a identifying entity set
- Entities are identified by the combination of:
  - A partial key of the weak entity set
  - The particular entity they are related to in the identifying entity set

Weak Entity Set

E.g. An employee's dependents might be weak entities --- they can't exist (so far as the database is concerned) if the relevant employee does not exist.

- 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 relationship Supports



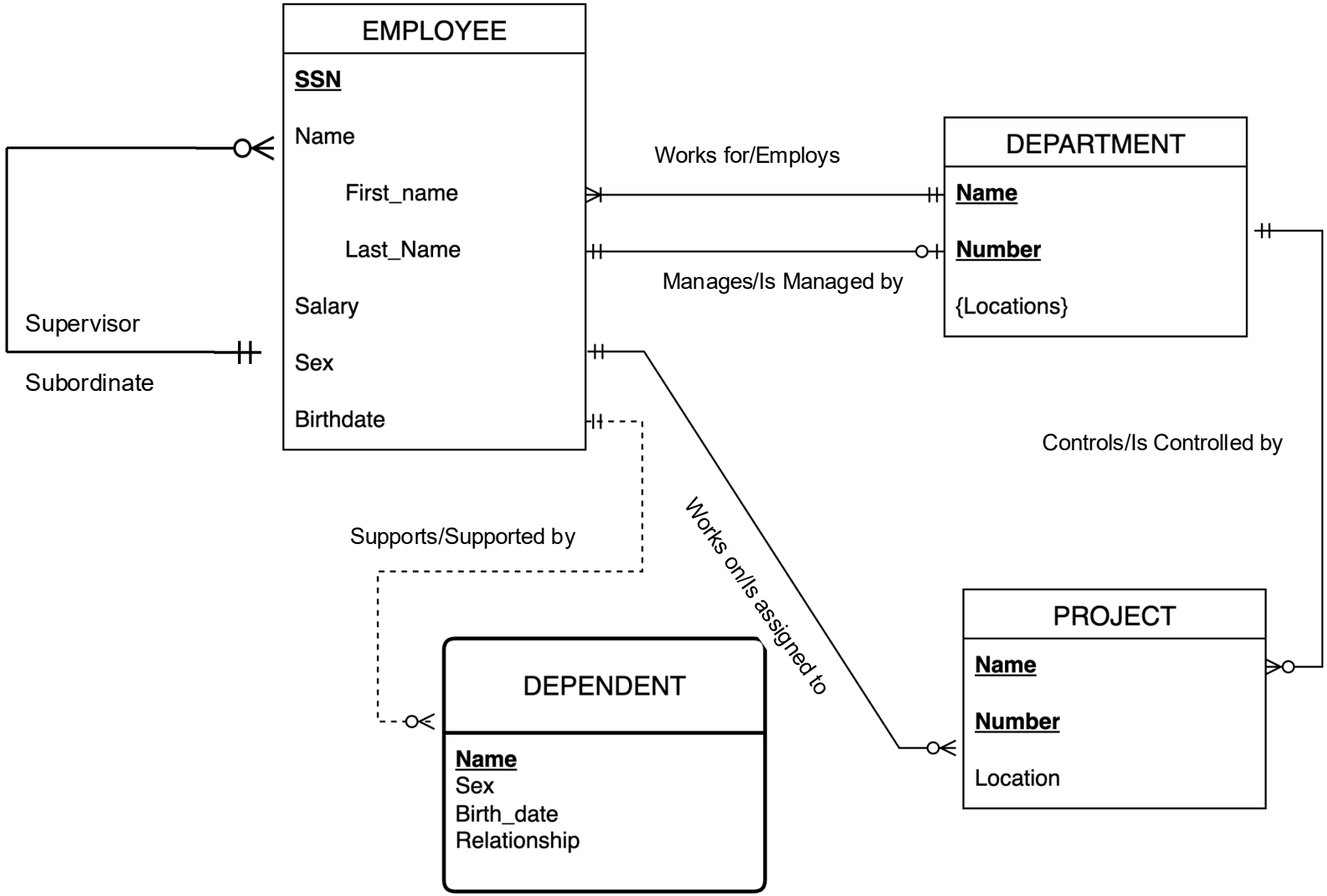
Exercise :

- Based on the requirements, use cardinality notations to identify relationships between entity types in the company:
  - DEPARTMENT
  - PROJECT
  - EMPLOYEE
  - DEPENDENT

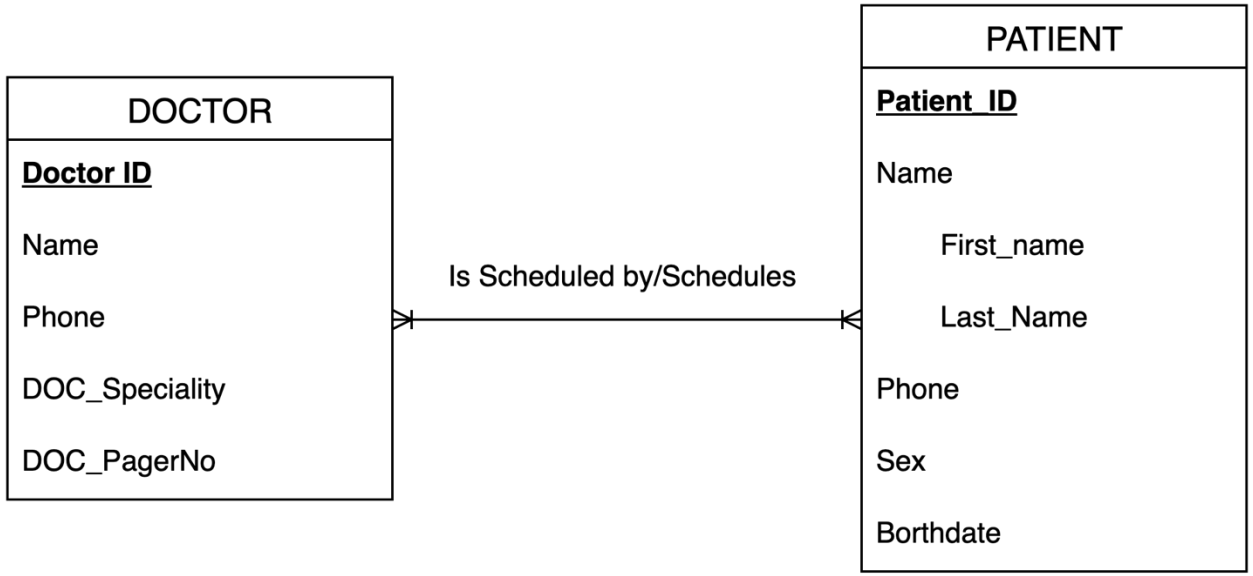
Requirements Gathering for EER Modeling (COMPANY Case)

The company's organizational structure and activities can be represented using an **Entity–Relationship (ER) model**. The key business elements and their relationships are as follows:

- **Departments**
  - Each department is identified by a *name* and a *number*.
  - Every department is managed by one **Employee**, with the *start date* of the manager's assignment recorded.
  - A department may operate in multiple **locations**.
- **Projects**
  - Each department is responsible for one or more projects.
  - A project is defined by a *unique name* and *number*, and is associated with a single *location*.
- **Employees**
  - Employees are associated with a single department but may participate in multiple projects.
  - For each project assignment, we track the *number of hours per week* the employee contributes.
  - Each employee has a designated *direct supervisor*.
  - Employee details include *SSN, name, salary, sex,* and *birthdate*.
- **Dependents**
  - Employees may have zero or more dependents.
  - For each dependent, we record *name, sex, birthdate,* and the *relationship* to the employee.


EMPLOYEE
SSN
Name
First_name
Last_Name
Salary
Sex
Birthdate
DEPARTMENT
Name
Number
{Locations}
Works for/Employs
Manages/Is Managed by
Supervisor
Subordinate
Controls/Is Controlled by
Supports/Supported by
Works on/Is assigned to
DEPENDENT
Name
Sex
Birth_date
Relationship
PROJECT
Name
Number
Location

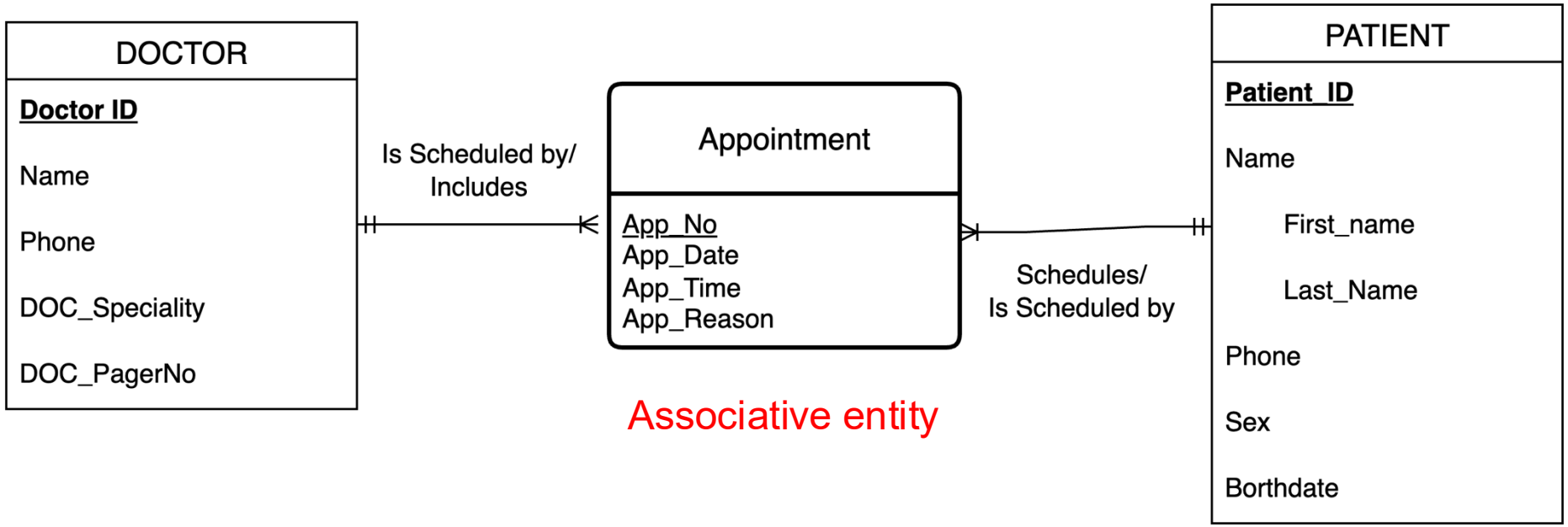
Attributes Associated with Relationship



•Doctor and Patient have a many-to-many relationship:

- A doctor can have many patients.
- A patient can see many doctors.

If we just connect Doctor ↔ Patient directly, we miss important details about the appointment.

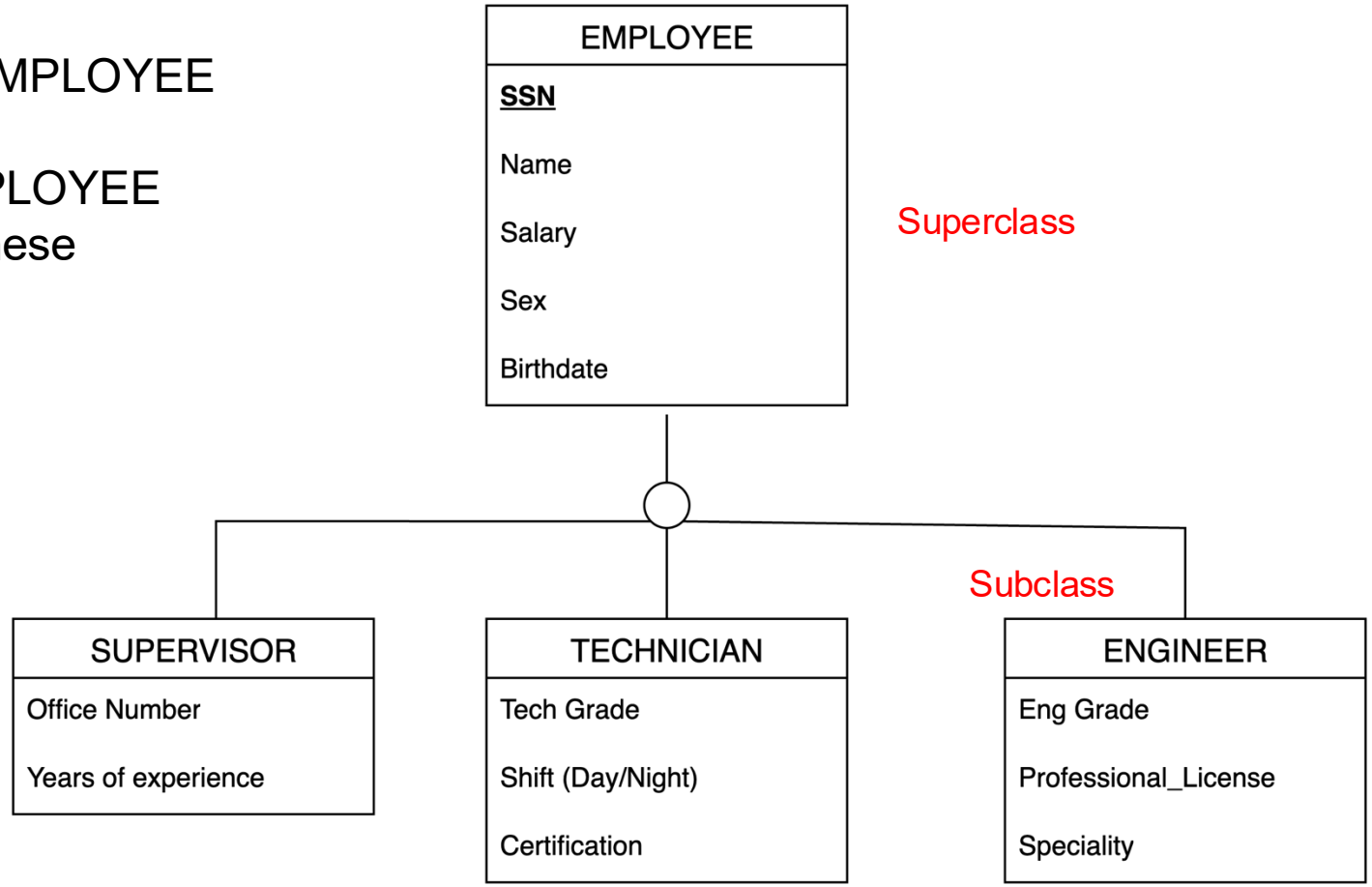


Associative entity

Subclasses and Superclasses

- An entity type may have additional meaningful subgroupings of its entities
  - ***Example:*** EMPLOYEE may be further grouped into: SECRETARY, ENGINEER, TECHNICIAN, … based on the EMPLOYEE's Job

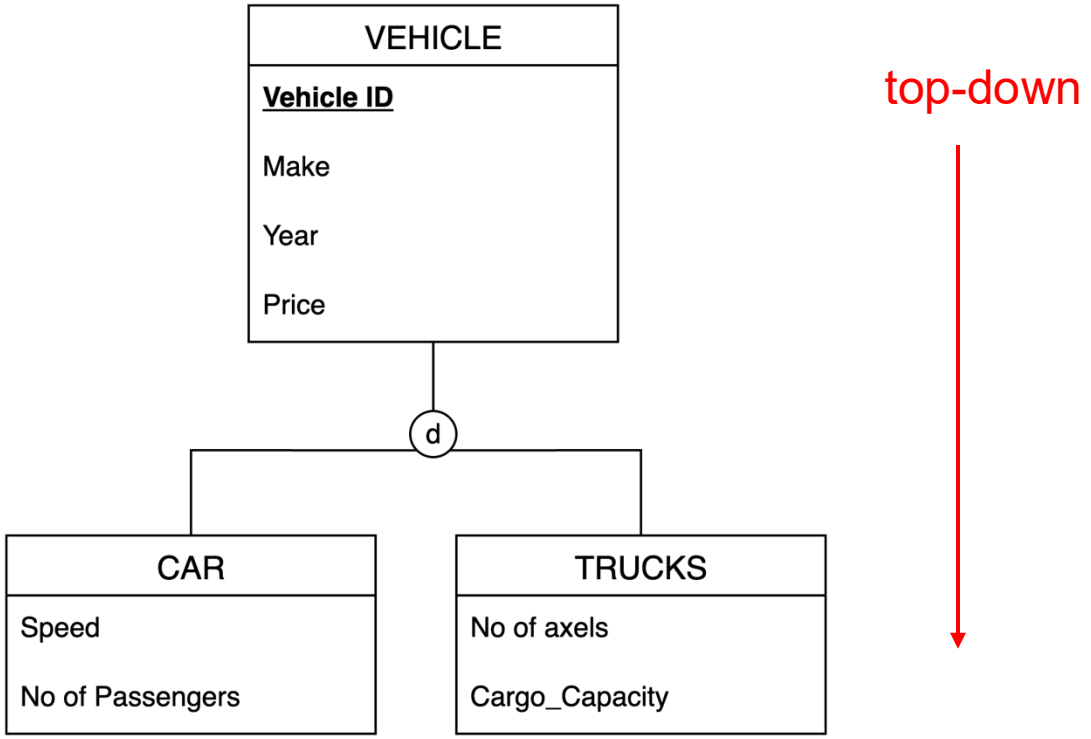
- Each of these subgroupings is a subset of EMPLOYEE entities
- Each sub group is called a **subclass** of EMPLOYEE
- EMPLOYEE is the **superclass** for each of these subclasses
- These are also called IS-A relationships
  - ENGINEER IS-A EMPLOYEE,
  - TECHNICIAN IS-A EMPLOYEE, ….



Notes on Subclasses and Superclasses

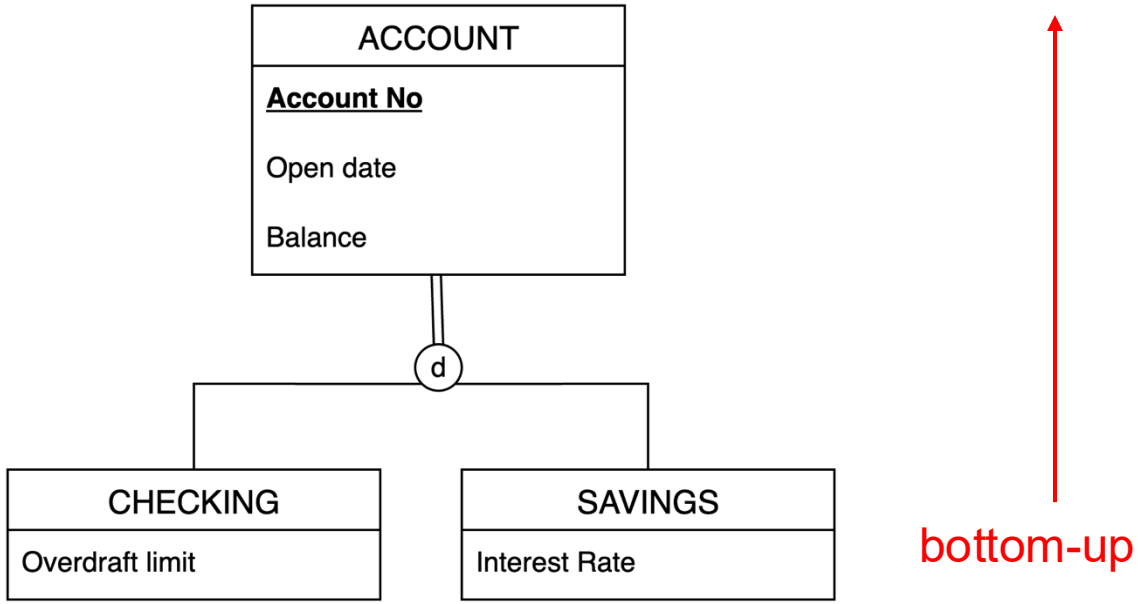
- An entity cannot exist in the database merely by being a member of a subclass; it must also be a member of the superclass.
- An entity that is member of a subclass inherits
  - All attributes of the entity as a member of the superclass
  - All relationships of the entity as a member of the superclass
    - *Example*: In the previous slide, SUPERVISOR (as well as TECHNICIAN and ENGINEER) inherit the attributes Name, SSN, …, from EMPLOYEE
- It is not necessary that every entity in a superclass be a member of some subclass

Specialization



- Start with a broad superclass and define more specific subclasses by adding distinguishing properties.
- Subclasses inherit all attributes/relationships of the superclass and may add their own.
  - *Example :* a VEHICLE can be specialized into CAR or TRUCK. Common attributes like Vehicle ID, Price, etc. become part of a VEHICLE and specialized attributes like Speed, No of axels become part of a specialized entity.

Generalization



- Start with several similar subclasses and form a broader superclass by extracting their common features.
- Move shared attributes/relationships up to the new superclass.
  - *Example :*CHECKING and SAVING can be generalized to a entity called ACCOUNT. Common attributes like Acc No and Open date, etc become part of a ACCOUNT

Exercise

Circle the correct answer

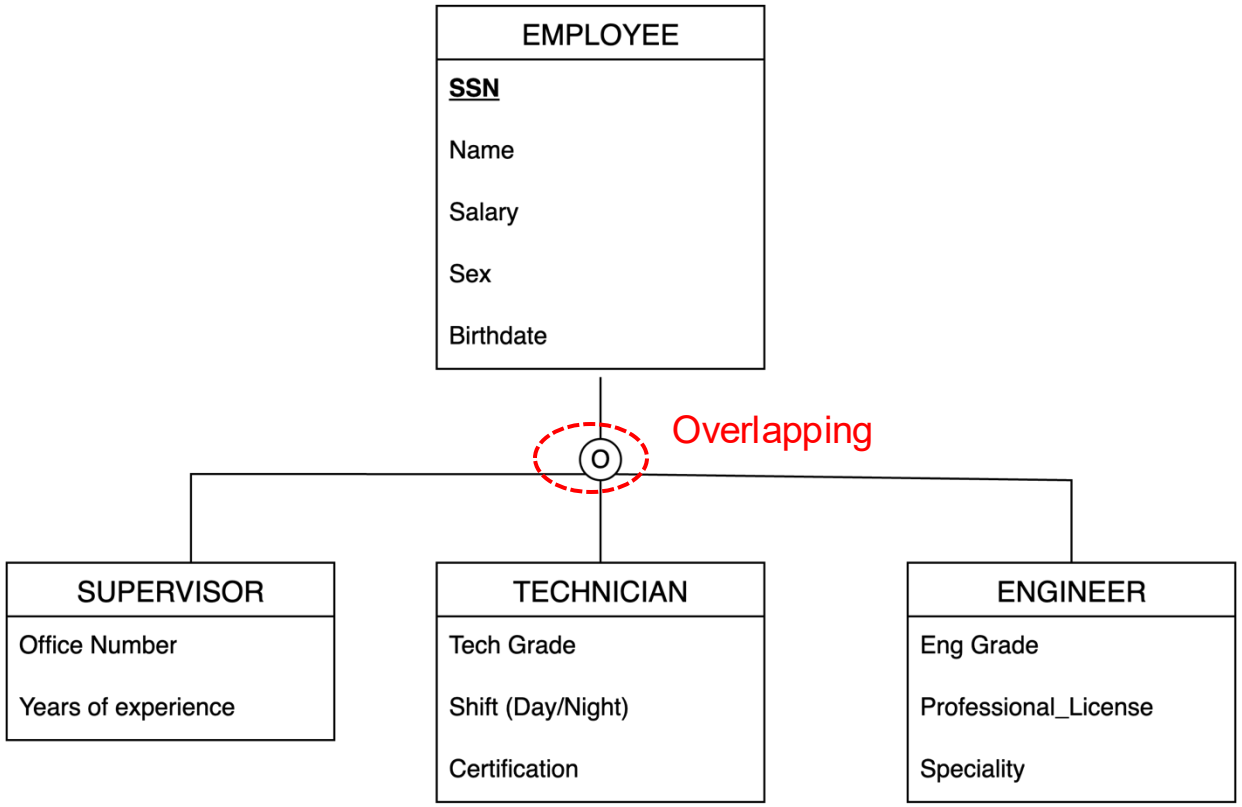
a. You already have three classes — CREDIT_CARD_PAYMENT, BANK_TRANSFER_PAYMENT, MOBILE_WALLET_PAYMENT — all with common attributes (Amount, Timestamp, Currency). You propose a common PAYMENT class to hold the shared parts.

- This step is: Generalization/ Specialization

b. You start with a broad USER class on a marketplace platform and define two more specific subclasses — SELLER (PayoutAccount, StoreName) and BUYER (DefaultShippingAddress) — that add attributes/roles.

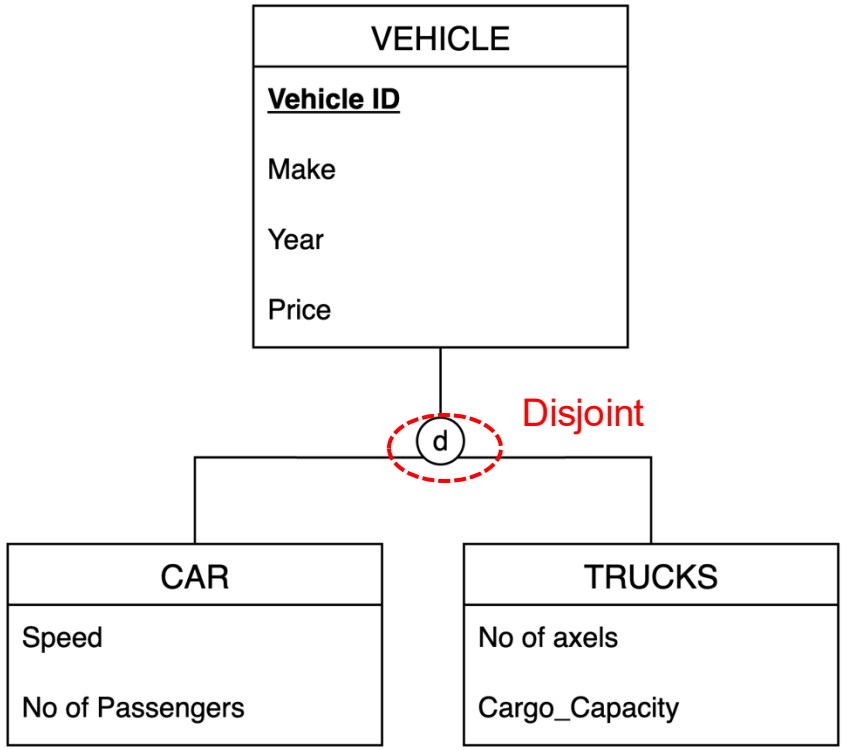
- This step is: Generalization / Specialization

Overlapping



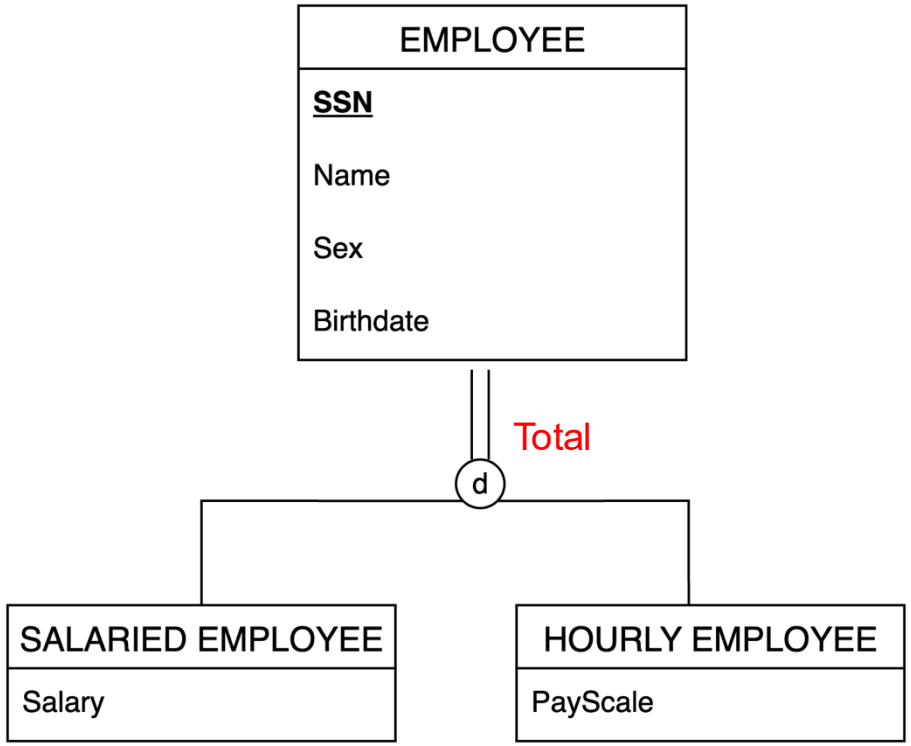
- A superclass entity may belong to multiple subclasses simultaneously.
  - *Example:* An employee can be both a supervisor and an engineer.

Disjoint



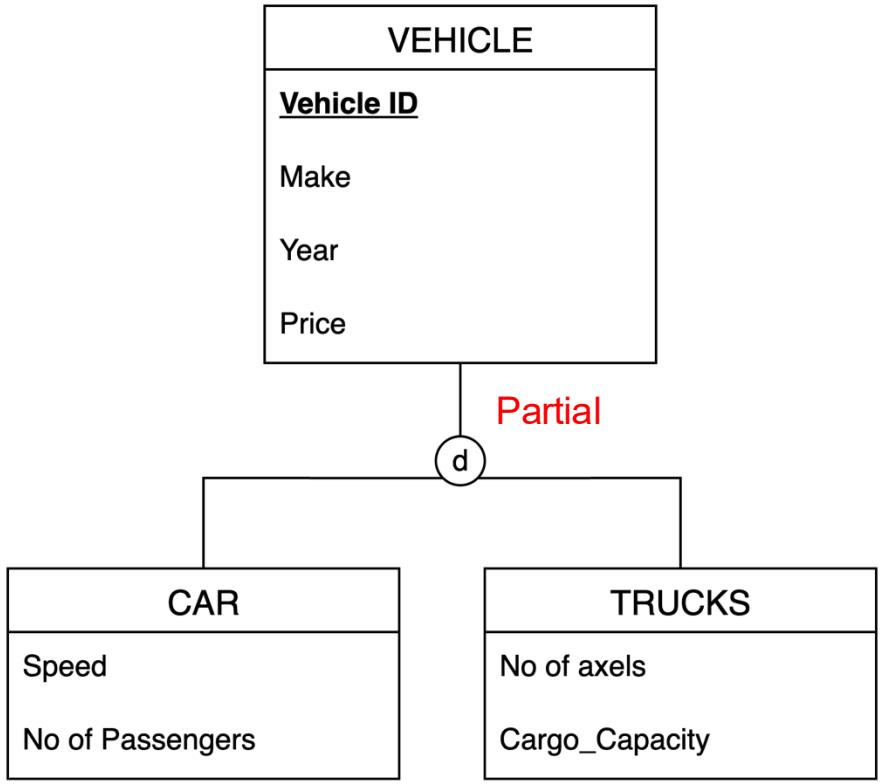
- A superclass entity can belong to at most one subclass.
  - *Example:* A vehicle cannot be both a Car and TRUCK simultaneously

Total Relationships



- Every entity in the superclass must belong to at least one subclass.
  - *Example:* all EMPLOYEES are either HOURLY_EMPLOYEES or SALARIED_EMPLOYEES.

Partial Relationships



- Some entities may not belong to any subclass.
  - *Example:* not every VEHICLE is a CAR, or a TRUCK.

Exercise

1. Every EXHIBIT in a museum is either PERMANENT or TEMPORARY; an exhibit cannot be both.
   a. Circle exactly one choice in each pair
      i. Disjoint /Overlapping
      ii. Partial / Total
   b. Draw a EER diagram with correct notation. Add at least 2 attributes to EXHIBIT and at least 1 attribute to each subclass

2. A CONFERENCE_PARTICIPANT can be a SPEAKER and/or a REVIEWER; many participants are neither.
   a. Circle exactly one choice in each pair
      i. Disjoint /Overlapping
      ii. Partial / Total
   b. Draw a EER diagram with correct notation. Add at least 2 attributes to EXHIBIT and at least 1 attribute to each subclass