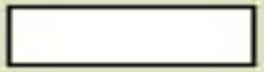
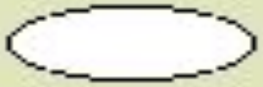
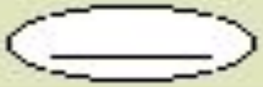
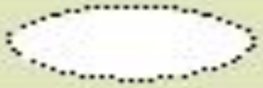
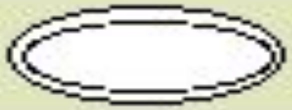
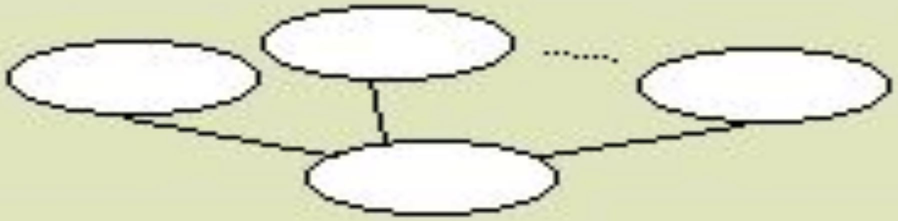
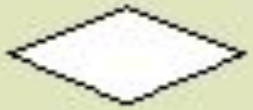
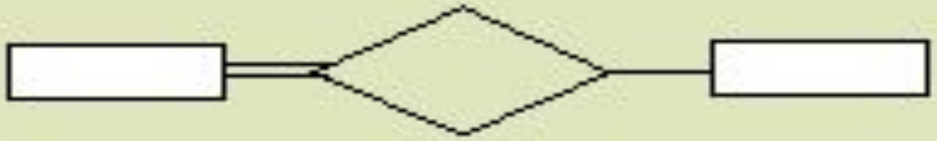
DATABASE SYSTEMS

WEEK 6 LECTURE 2

Notation	**Meaning**
	Entity type
	Attribute
	Key attribute
	Derived attribute
	Multivalued attribute
	Composite attribute
	Relationship type
	Total participation

Topics to Cover

- Attributes of relationships
- Inheritance
- Super type
- Subtypes
- Constraints
  - Completeness
  - Disjointness

Attributes of Relationship

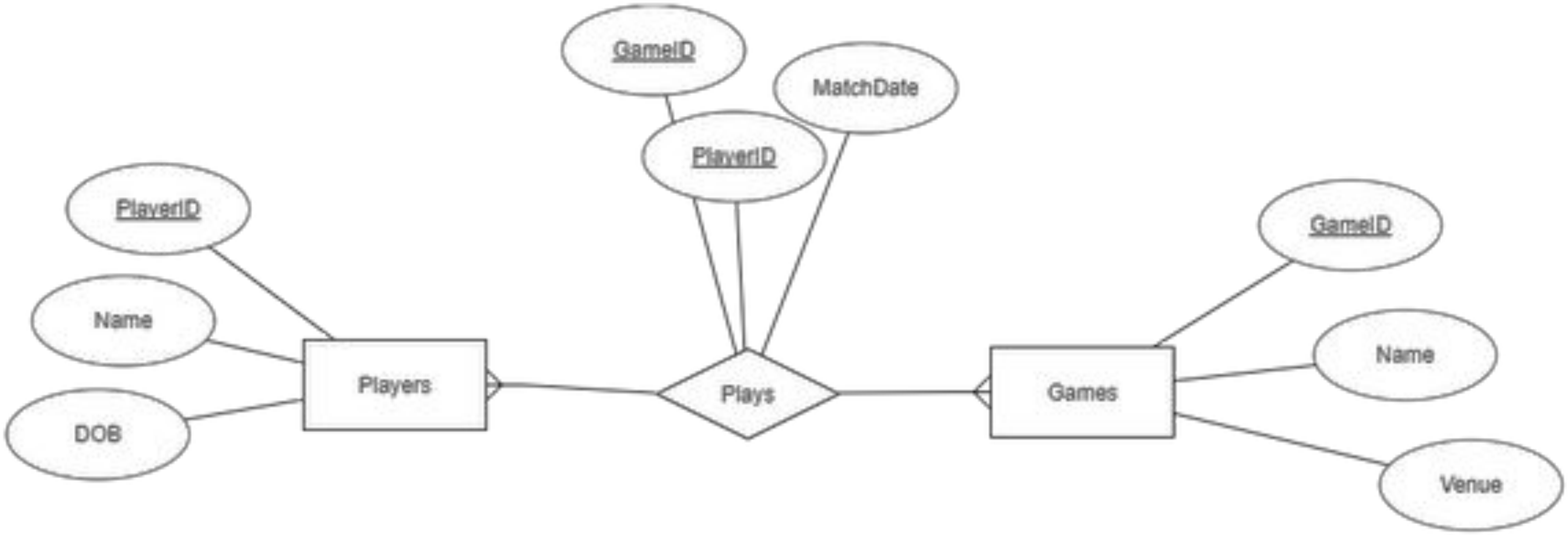
- If the relationship between two entities are many to many we will need an additional relationship table and the new table shall have at least two attributes (foreign keys) one from each table (their respective primary keys).
- However, we can add more attributes depending on the needs and depth of data to capture.

Attributes of Relationship

- Lets take an example with two entities **Players** and **Games**.
- **Scenario**: Many Players play many games
- The association between these two tables will be many to many. Table *Players* only contain player details (PlayerID, Name,….) and table *Games* contain only game details (GameID, Name, Venue,…).
- To know which player played which game and when, we construct a new table (Plays) with attributes from either table.

Attributes of Relationship

- Also note that ***PlayerID*** on *Plays* is a **foreign key** and references the ***Players*** table. ***GameID*** on *Plays* is a **foreign key** and references the *Games* table. But, the combination of both *GameID* and *PlayerID* forms a composite primary key in the *Plays* table.



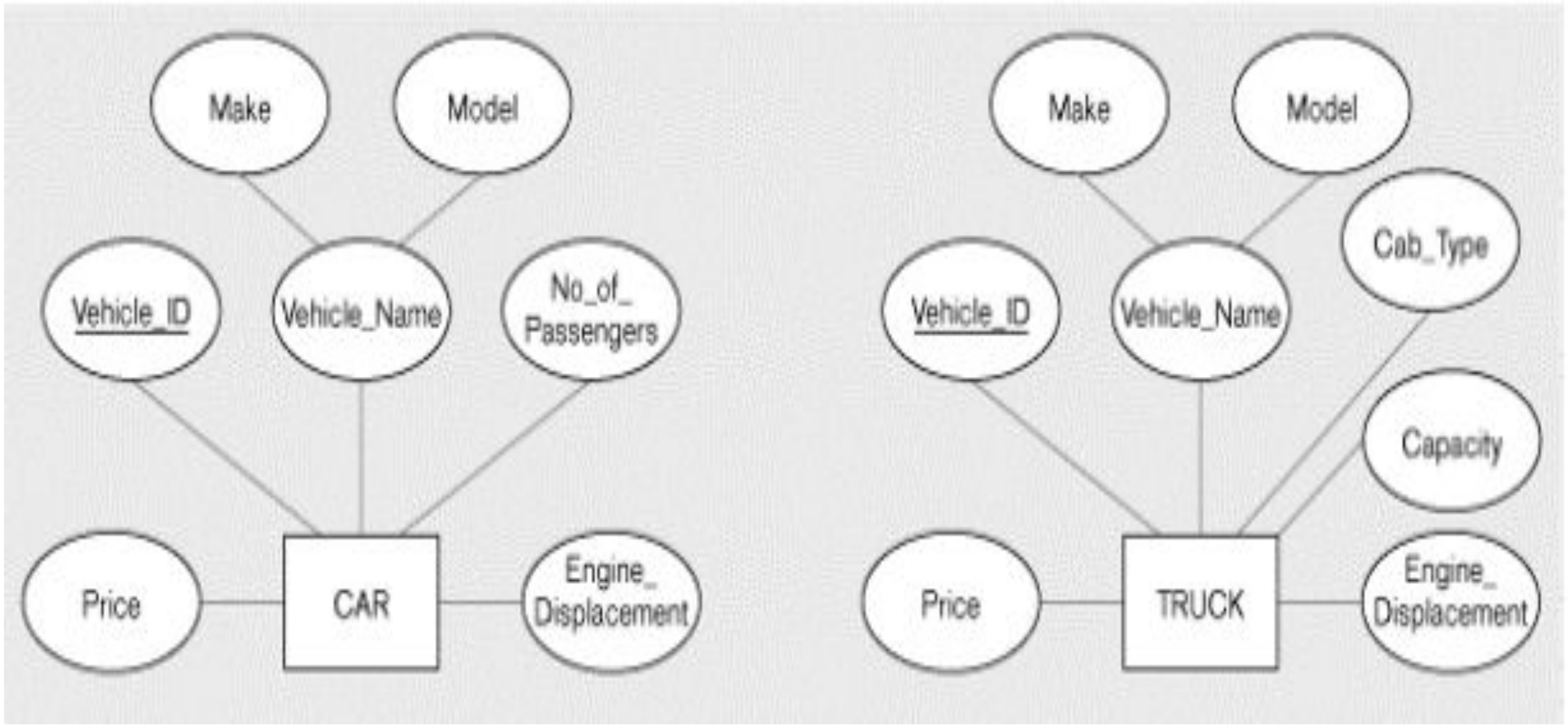
Inheritance

- Inheritance in database systems mean the transfer of properties of one entity to some derived entities, which have been derived from the same entities.
- The transfer of the characteristics of a class in object-oriented programming to other classes derived from it.
- For example, if “vegetable” is a class, the classes “potato” and “beetroot” can be derived from it, and each will inherit the properties of the “vegetable” class: name, growing season, and so on.

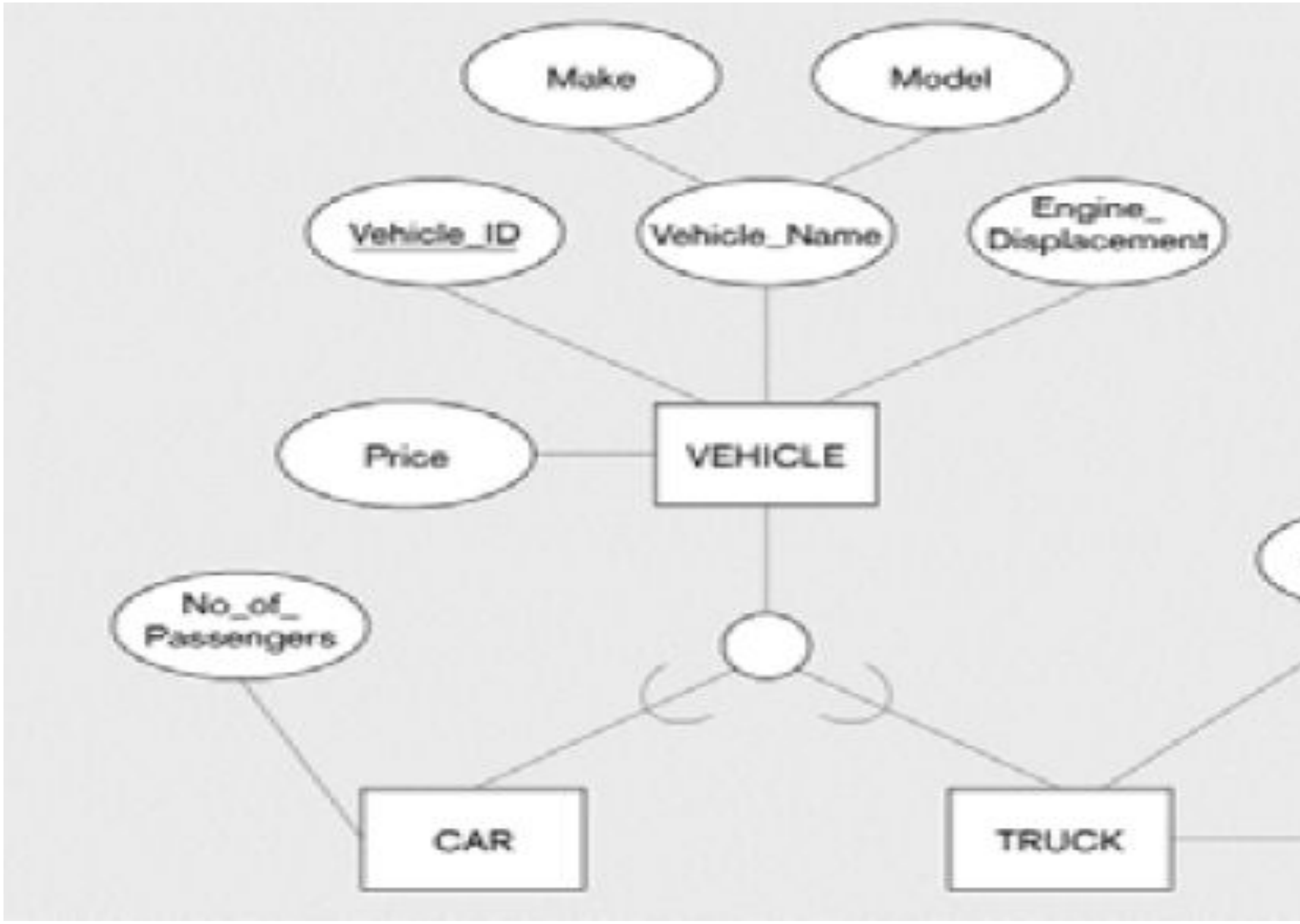
Ways to develop super/subtype relationships

- ***Generalization:*** The process of defining a more general entity type from a set of more specialized entity types. BOTTOM-UP
- ***Specialization:*** The process of defining one or more subtypes of the supertype, and forming supertype/subtype relationships. TOP-DOWN

Ways to develop super/subtype relationships (specialization)



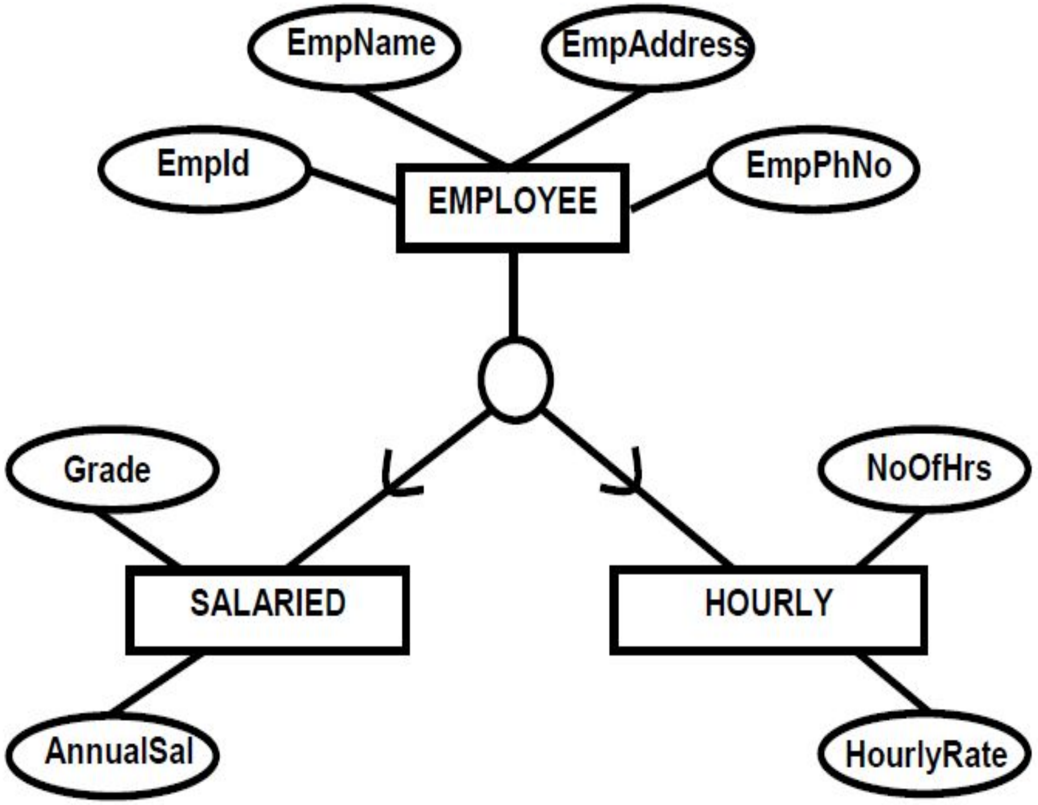
Ways to develop super/subtype relationships (specialization)



So we put the shared attributes in a supertype

Super types and Subtypes

- Subtypes hold all the properties of their corresponding super-types. Means all those subtypes which are connected to a specific super type will have all the properties of their super type.



Super types and Subtypes

- In the Figure we can see that the attributes which are specific to the subtype entities are not shown with the supertype entity.
- Only those attributes are shown on the supertype entity which are to be inherited to the subtypes and are common to all the subtype entities associated with this supertype.

Super types and Subtypes

- Supertype / subtype Relationship: The use of supertype and subtype for the entities is very useful because it allows us to create hierarchy of the entities according to the attributes they have and we need not to write all the attributes again and again.
- We can group similar types of entities and the attributes associated with those entities at certain levels.

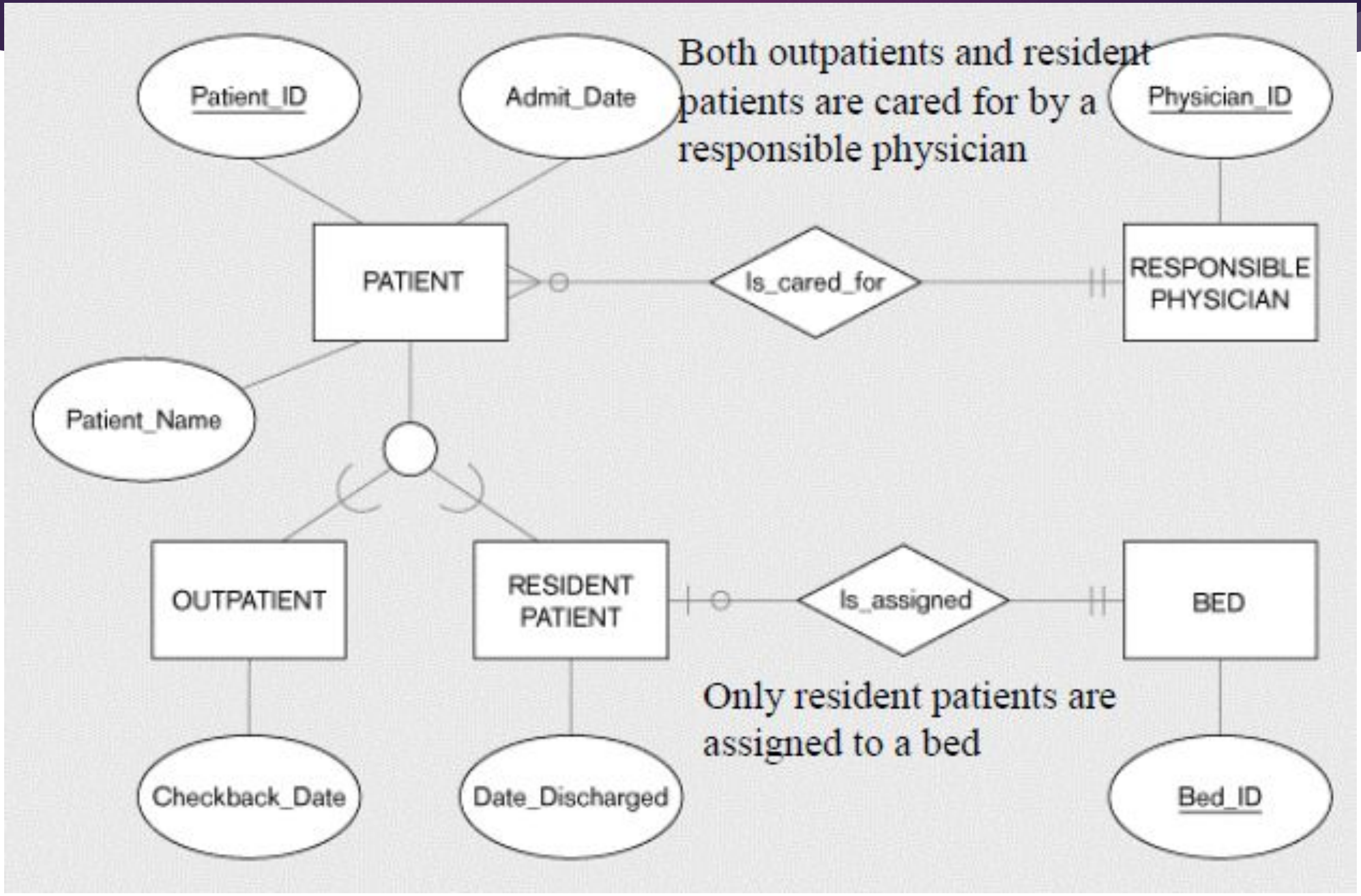
Super types and Subtypes

- The process of identifying supertype and creating different type of sub entities is supported by the general knowledge of the designer about the organization and also based of the attributes of the entities which are entities existing in the system.

Relationships and Subtypes

- Relationships at the ***supertype*** level indicate that all subtypes will participate in the relationship
- The instances of a ***subtype*** may participate in a relationship unique to that subtype. In this situation, the relationship is shown at the subtype level.

Relationships and Subtypes



Both outpatients and resident patients are cared for by a responsible physician

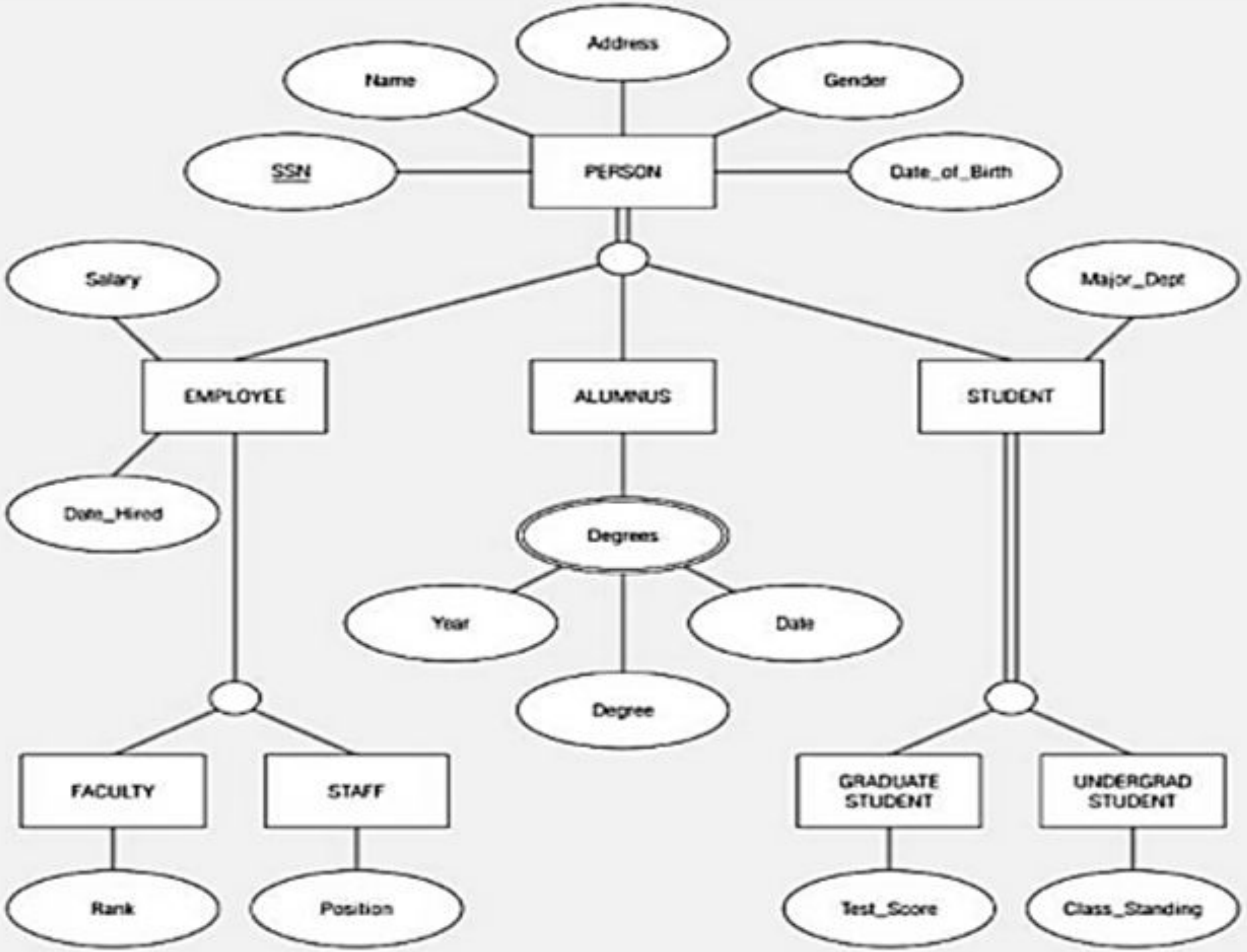
Only resident patients are assigned to a bed

Specifying Constraints

- Once there has been established a super/sub entity relationship there are a number of constraints which can be specified for this relationship for specifying further restrictions on the relationship.
- Completeness Constraint
  - Total Completeness
  - Partial Completeness
- Disjointness Constraint
  - Disjointness rule
  - Overlap rule

Completeness Constraint

- There are two types of completeness constraints, partial completeness constraints and total completeness constraints.
- **Total Completeness:** Total Completeness constraint exist only if we have a super type and some subtypes associated with that supertype, and the following situation exists between the super type and subtype.
- All the instances of the supertype entity must be present in at one of the subtype entities, i.e.—there should be not instance of the supertype entity which does not belong to any of the subtype entity.


Address
Name
Gender
SSN
PERSON
Date_of_Birth
Salary
Major_Dept
EMPLOYEE
ALUMNUS
STUDENT
Date_Hired
Degrees
Year
Date
Degree
FACULTY
STAFF
GRADUATE STUDENT
UNDERGRAD STUDENT
Rank
Position
Test_Score
Class_Standing

Completeness Constraint

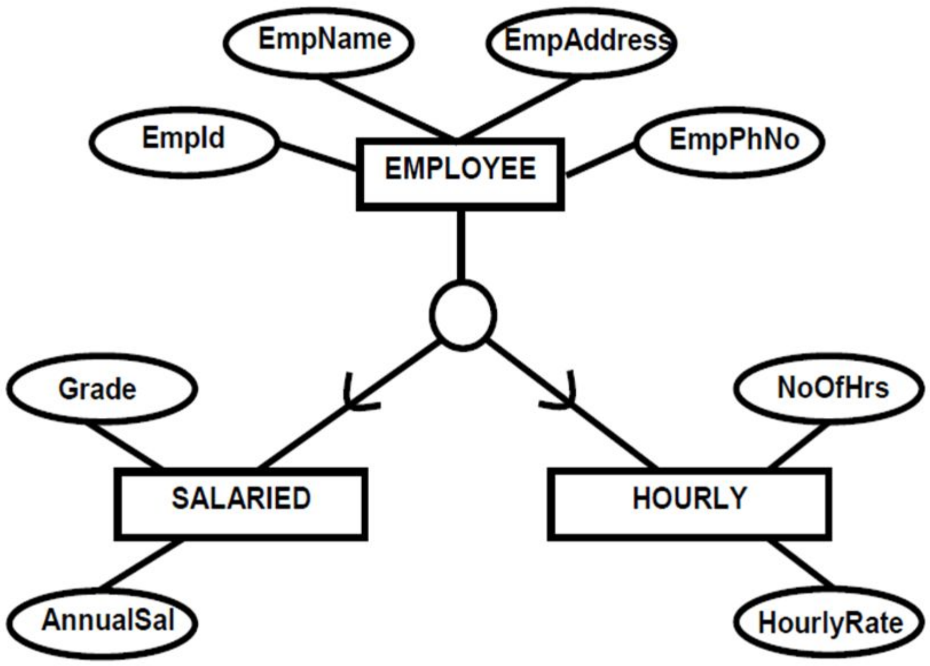
- **Total Completeness:** This is a specific situation when the supertype entities are very carefully analyzed for their associated subtype entities and no sub type entity is ignored when deriving sub entities from the supertype entity.

Completeness Constraint

- This type of completeness constraint exists when it is not necessary for any supertype entity to have its entire instance set to be associated with any of the subtype entity.
- This type of situation exists when we do not identify all subtype entities associated with a supertype entity, or ignore any subtype entity due to less importance of least usage in a specific scenario.

Disjointness Constraint

- **Disjoint constraint:** This constraint restricts the existence of one instance of any supertype entity to exactly one instance of any of the subtype entities.

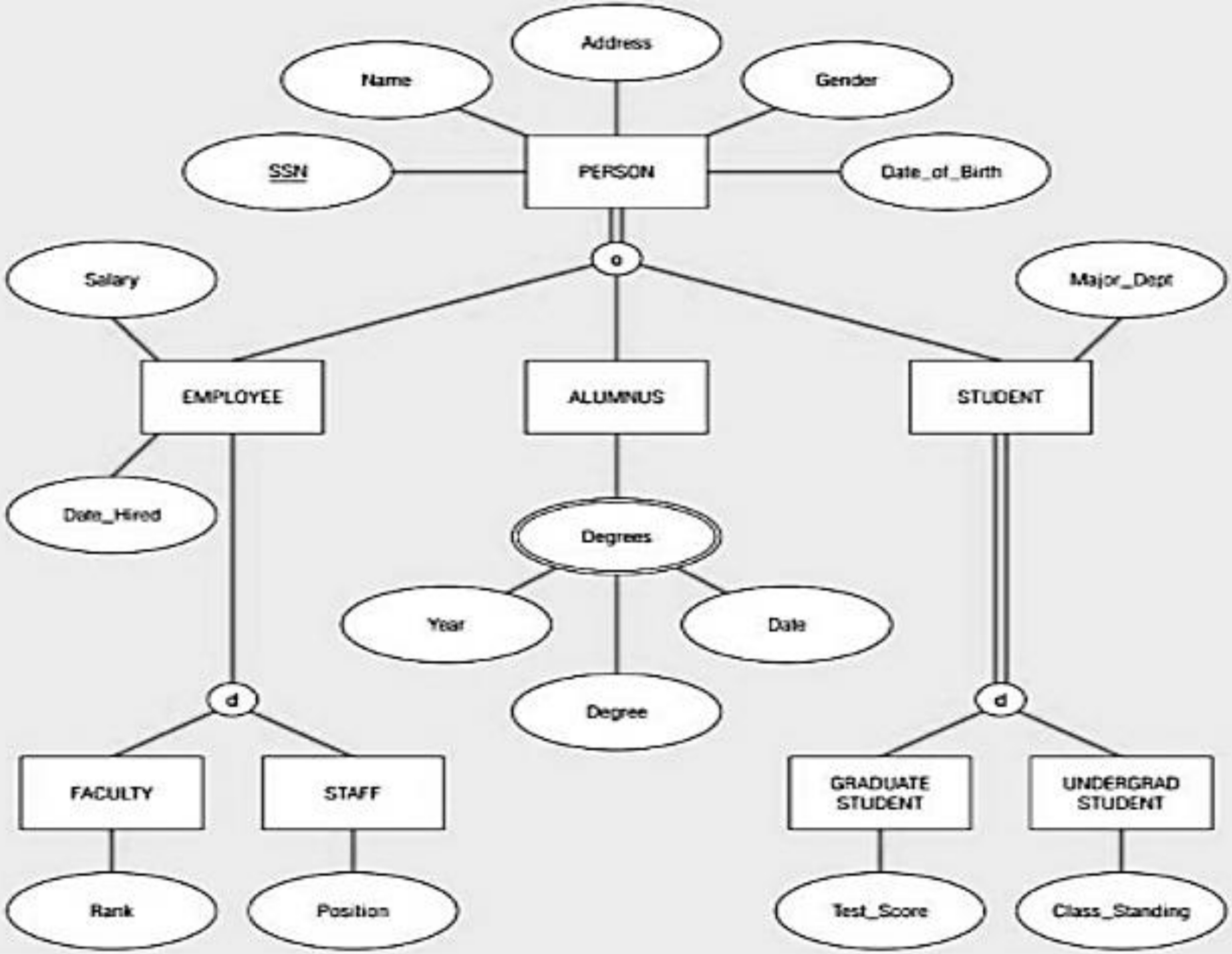


Disjointness Constraint

- **Disjoint constraint:** From the example, it is seen that there can be two types of employees, one which are fixed salary employees and the others are hourly paid employees.
- Now the disjoint rule tells that at a certain type an employee will be either hourly paid employee or salaried employee, he can not be placed in both the categories in parallel.

Disjointness Constraint

- **Overlap Rule:** This rule is in contrast with the disjoint rule, and tells that for one instance of any supertype entity there can be multiple instances existences of the of the instance for more then one subtype entities.
- Again taking the same example of the employee in an organization we can say that one employee who is working in an organization can be allowed to work for the company at hourly rates also once he has completed his duty as a salaried employee.
- In such a situation the employee instance record for this employee will be stored in both the sub entity types.


Address
Name
Gender
SSN
PERSON
Date_of_Birth
o
Salary
Major_Dept
EMPLOYEE
ALUMNUS
STUDENT
Date_Hired
Degrees
Year
Date
Degree
d
d
FACULTY
STAFF
GRADUATE STUDENT
UNDERGRAD STUDENT
Rank
Position
Test_Score
Class_Standing