DBMS: Lecture 3
(Entity Relationship (ER) Model)

By

Dr. Sumit Kumar Tetarave

Entity

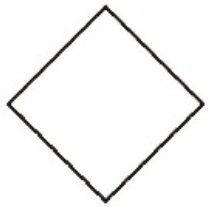
Weak Entity

Attribute

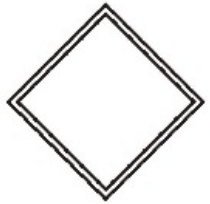
Key Attribute

Derived Attribute

- An **entity** is an object of concern used to represent the things (or, a concept) in the real world.
- It represents a class of things, not any one instance, e.g., “STUDENT” entity has instances of “Ramesh” and “Mohan”.
- A collection of a similar kind of entities is called an **Entity Set or entity type.**
- An entity type usually has an attribute whose values are distinct for each individual entity in the collection.
  - Such an attribute is called a **key attribute** and its values can be used to identify each entity uniquely.
  - The entity types containing a key attribute are called **strong entity** types or regular entity types.
- Entity types that do not contain any key attribute, and hence cannot be identified independently, are called **weak entity** types.

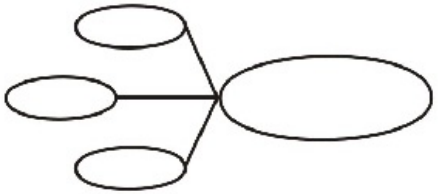


Relationship



Identifying Relationship

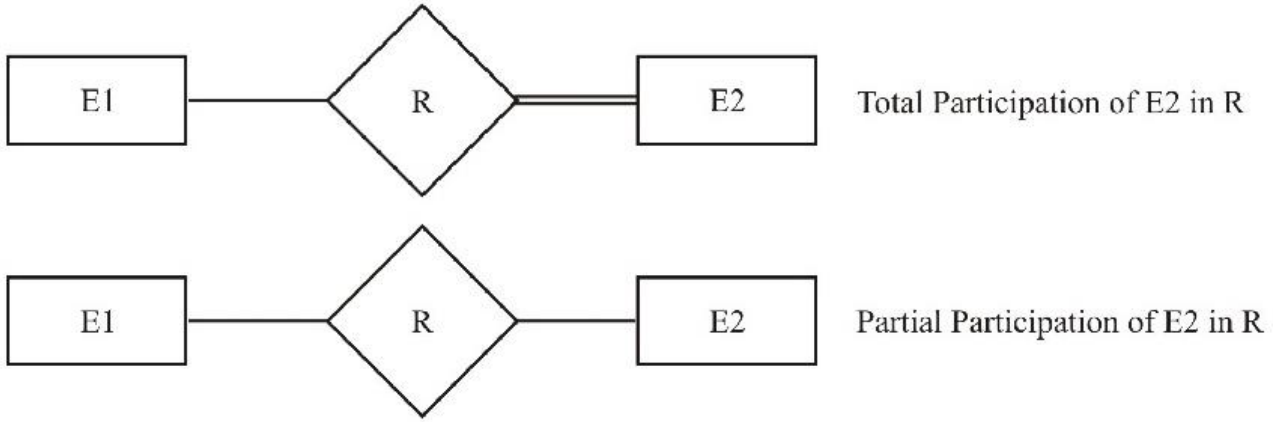
- An **attribute** is a property used to describe the specific feature of the entity.
- So to describe an entity entirely, a set of attributes is used.
- For example, a student entity may be described by the student’s name, age, address, course, etc.
- An entity will have a value for each of its attributes.
- For example for a particular student the following values can be assigned: RollNo: 1234 Name: Rakesh, Age: 18, Address: B-4, Mayapuri, Delhi, and Course: B.Sc. (H)



Composite Attribute

Types of attributes: Simple (First Name), Composite (Name), Single valued (Age), Multivalued (phone numbers**), Stored (**directly stored in the data base, eg., “Birth date” attribute of a person.**), Derived (**are not stored directly but can be derived from stored attributes, eg., total salary of a “person” can be calculated from “basic salary” **)**

More about Entities and Relationships

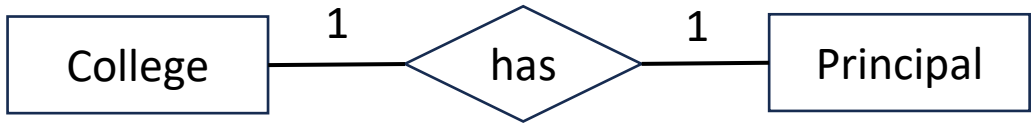


- **Recursive relationships**
  - When the same entity type participates more than once in a relationship type in different roles, the relationship types are called recursive relationships.
- **Participation constraints**
  - The participation Constraints specify whether the existence of an entity depends on its being related to another entity via the relationship type.
  - There are 2 types of participation constraints:
    - **Total:** When all the entities from an entity set participate in a relationship type, is called total participation,
      - For example, the participation of the entity set student in the relationship set must “opts” is said to be total because every student enrolled must opt for a course.
    - **Partial:** When it is not necessary for all the entities from an entity set to participate in a relationship type, it is called partial participation.
      - For example, the participation of the entity set student in “represents” is partial, since not every student in a class is a class representative.

Relationship Cardinality

- Cardinality specifies the number of instances of an entity associated with another entity participating in a relationship.
- Based on the cardinality binary relationship can be further classified into the following categories:
- **One-to-one**: An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A.

  Example: Relationship between college and principal.



  - One college can have at the most one principal and one principal can be assigned to only one college.
  - Similarly we can define the relationship between university and Vice Chancellor.

- **One-to-many**: An entity in A is associated with any number of entities in B.
  - An entity in B is associated with at the most one entity in A.

  Example: Relationship between department and faculty.

  - One department can appoint any number of faculty members but a faculty member is assigned to only one department.

Relationship Cardinality…

- **Many-to-one**: An entity in A is associated with at most one entity in B. An entity in B is associated with any number in A.
- **Example:** Relationship between course and instructor.

- An instructor can teach various courses but a course can be taught only by one instructor. Please note this is an assumption.
- **Many-to-many**: Entities in A and B are associated with any number of entities from each other.
- **Ex1:** Taught_by Relationship between course and faculty.

- One faculty member can be assigned to teach many courses and one course may be taught by many faculty members.

- **Ex2:** Relationship between book and author.

- One author can write many books and one book can be written by more than one authors.

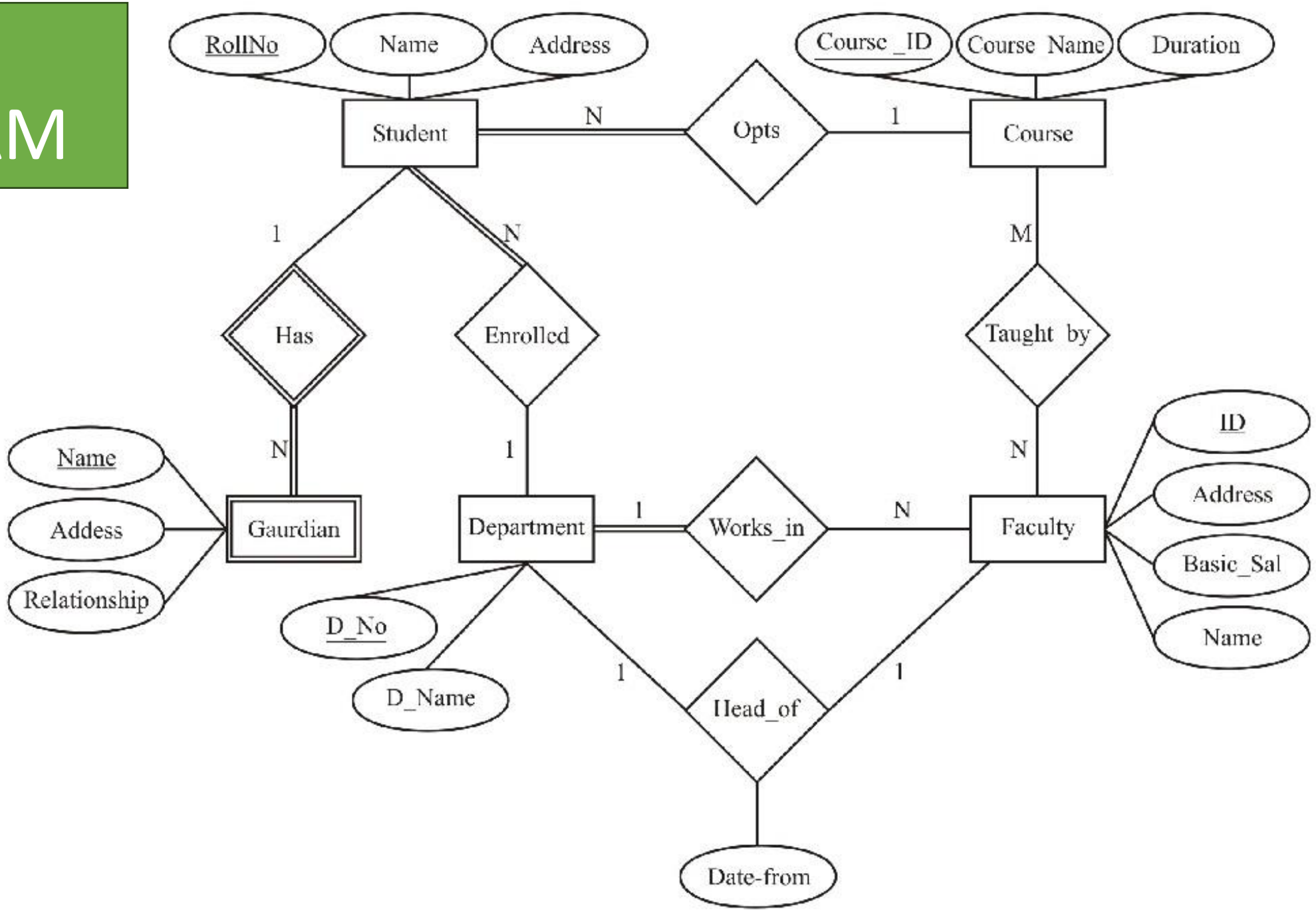
An Example Database Application

- College database keeps track of Students, Faculty, Departments and Courses organized by various departments.
- Following is the description of COLLEGE database:
  - College contains various departments like Computer Applications, Mechanical Engineering, etc.
    - Each department is assigned a unique id and name.
    - Some faculty members are also appointed to each department and one of them works as head of the department.
  - There are various courses conducted by each department.
    - Each course is assigned a unique id, name and duration.
  - Faculty information contains name, address, department, basic salary etc.
    - A faculty member is assigned to only one department but can teach various courses of other department also.
  - Student's information contain Roll no (unique), Name, Address etc.
    - A student can opt only for one course.
  - Parent (guardian) information is also kept along with each student.
    - We keep each guardian's name, age, sex and address.

Defining Relationship For College Database

- Using the concepts defined earlier, we have identified that strong entities in COLLEGE database are
  - STUDENT, FACULTY, COURSE and DEPARTMENT.
- This database also has one weak entity called GUARDIAN. We can specify the following relationships:
  1. **Head-of**, is a 1:1 relationship between FACULTY and DEPARTMENT.
     - Participation of the entity FACULTY is partial since not all the faculty members participate in this relationship,
     - while the participation from department side is total, since every department has one head.
  2. **Works_in**, is a 1:N relationship between DEPARTMENT and FACULTY.
     - Participation from both side is total.
  3. **Opts,** is a 1:N relationship between COURSE and STUDENT.
     - Participation from student side is total because we are assuming that each student enrolled opts for a course.
     - But the participation from the course side is partial, since there can be courses that no student has opted for.
  4. **Taught_by**, is a M: N relationship between FACULTY and COURSE,
     - as a faculty can teach many courses and a course can be taught by many faculty members.
  5. **Enrolled**, is a 1:N relationship between STUDENT and DEPARTMENT
     - as a student is allowed to enroll for only one department at a time.
  6. **Has**, is a 1:N relationship between STUDENT and GUARDIAN
     - as a student can have more than one local guardian and one local guardian is assumed to be related to one student only.
     - The weak entity Guardian has total participation in the relation “Has”.
- So now, let us make an **E-R diagram** for the college database.

E-R DIAGRAM


RollNo
Name
Address
Student
N
Opts
1
Course _ID
Course Name
Duration
Course
1
N
M
Has
Enrolled
Taught by
N
1
N
Name
Addess
Relationship
Gaurdian
Department
1
Works_in
N
Faculty
ID
Address
Basic_Sal
Name
D_No
D_Name
1
Head_of
1
Date-from

Conversion of ER Diagram to RELATIONAL DATABASE

- For every ER diagram we can construct a relational database which is a collection of tables. Following are the set of steps used for conversion of ER diagram to a relational database.

Conversion of entity sets:

- I) For each strong entity type E in the ER diagram, we create a relation R containing all the simple/key attributes of E.

STUDENT

ROLL NO: Primary Key	NAME	ADDRESS

FACULTY

ID: Primary Key	NAME	ADDRESS	BASIC_SAL

COURSE

COURSE_ID: PK	COURSE_NAME	DURATION

DEPARTMENT

D_NO: Primary Key	D_NAME

- II) For each weak entity type W in the E R Diagram, we create another relation R that contains all simple attributes of W.
  - If E is an owner entity of W then key attribute of E is also included in R.
  - This key attribute of R is set as a foreign key attribute of R.
  - Now the combination of primary key attribute of owner entity type and partial key of weak entity type will form the key of the weak entity type.
- *The* weak entity GUARDIAN, where the key field of student entity RollNo has been added.

ROLL NO NAME (Primary Key)		ADDRESS	RELATIONSHIP

Conversion of relationship sets- 1:1

D_NO: Primary Key	D_NAME

Binary Relationships:

- I) One-to-one relationship:
  - For each 1:1 relationship type R in the ER diagram involving two entities E1 and E2
    - we choose one of entities (say E1) preferably with total participation and
    - add primary key attribute of another entity E2 as a foreign key attribute in the table of entity (E1).
  - We will also include all the simple attributes of relationship type R in E1 if any.
- For example, the DEPARTMENT relationship has been extended to include head-Id and attribute of the relationship.
  - Note that we will keep information in this table of only current head and Date from which s/he is the head.
  - There is one Head_of 1:1 relationship between FACULTY and DEPARTMENT.
  - We choose DEPARTMENT entity having total participation and add primary key attribute ID of FACULTY entity as a foreign key in DEPARTMENT entity named as Head_ID.
- Now the DEPARTMENT table will be as follows:

D_NO	D_NAME	Head_ID	Date-from

Converting 1:1 relationship

Conversion of relationship sets- 1:n

- II) One-to-many relationship:
  - For each 1: n relationship type R involving two entities E1 and E2,
    - we identify the entity type (say E1) at the n-side of the relationship type R and
    - include primary key of the entity on the other side of the relation (say E2) as a foreign key attribute in the table of E1.
  - We include all simple attributes (or simple components of a composite attributes of R (if any) in the table of E1).
  - For example, the works_in relationship between the DEPARTMENT and FACULTY.
    - For this relationship choose the entity at N side, i.e., FACULTY and add primary key attribute of another entity DEPARTMENT, i.e., DNO as a foreign key attribute in FACULTY.

FACULTY

ID: Primary Key	NAME	ADDRESS	BASIC_SAL	D_NO

Converting 1:N relationship

Conversion of relationship sets- m:n

- III) Many-to-many relationship:
  - For each m:n relationship type R, we create a new table (say S) to represent R.
  - We also include the primary key attributes of both the participating entity types as a foreign key attribute in S.
  - Any simple attributes of the m:n relationship type (or simple components of a composite attribute) is also included as attributes of S.
  - For example, the m: n relationship taught-by between entities COURSE and FACULTY should be represented as a new table. The structure of the table will include primary key of COURSE and primary key of FACULTY entities.
  - A new table TAUGHT-BY will be created as: Primary key of TAUGHT-By table.

TAUGHT-BY

ID	COURSE_ID
{Primary key of FACULTY table}	{Primary key of COURSE table}

Converting M:N relationship

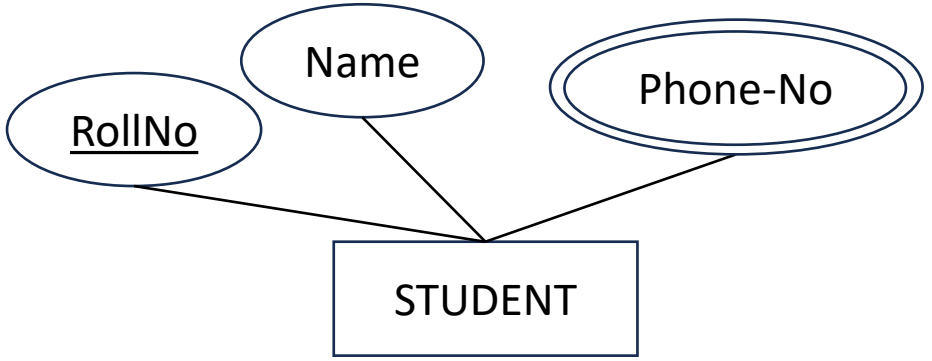
Conversion of relationship sets: n-ary

- **n-ary Relationship:**
  - For each n-ary relationship type R where n>2, we create a new table S to represent R.
  - We include as foreign key attributes in S the primary keys of the relations that represent the participating entity types.
  - We also include any simple attributes of the n-ary relationship type (or simple components of complete attributes) as attributes of S.
  - The primary key of S is usually a combination of all the foreign keys that reference the relations representing the participating entity types.
  - *TAUGHT-BY Table* is a special case of n-ary relationship: a binary relation.

TAUGHT-BY

ID	COURSE_ID
{Primary key of FACULTY table}	{Primary key of COURSE table}

Multivalued attributes:



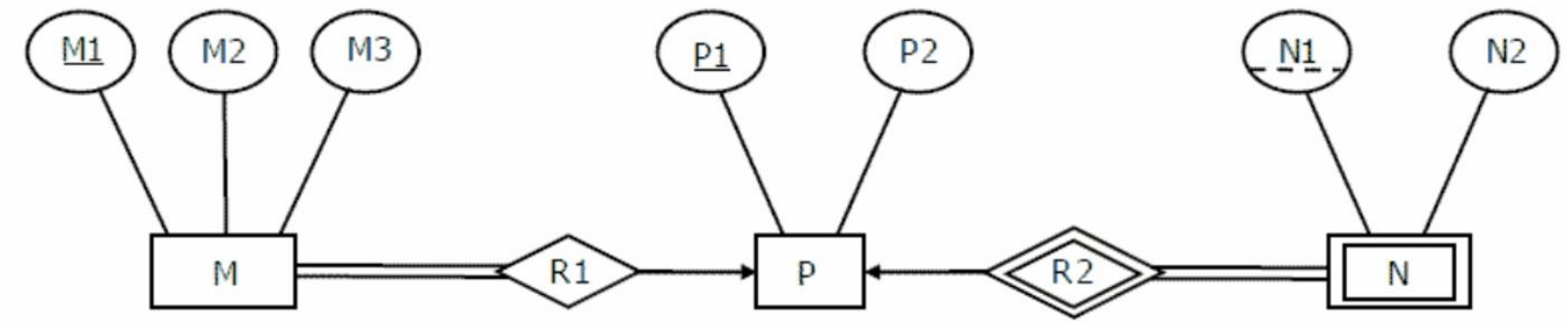
- For each multivalued attribute “A”,
  - We create a new relation R that includes
    - an attribute corresponding to plus
    - the primary key attribute k of the relation that represents the entity type or relationship type that has as an attribute.
  - The primary key of R is then combination of A and k.
  - For example, if a STUDENT entity has RollNo, Name and PhoneNumber where phone number is a multi-valued attribute then
    - We will create a table PHONE (**RollNo, Phone-No)** where primary key is the combination.
    - Please also note that then in the STUDENT table we need not have phoneNumber, instead it can be simply (Roll No, Name) only.

PHONE

RollNo	Phone-No

Example 1: The minimum number of tables are needed to represent the following ER model-

Answer: 3

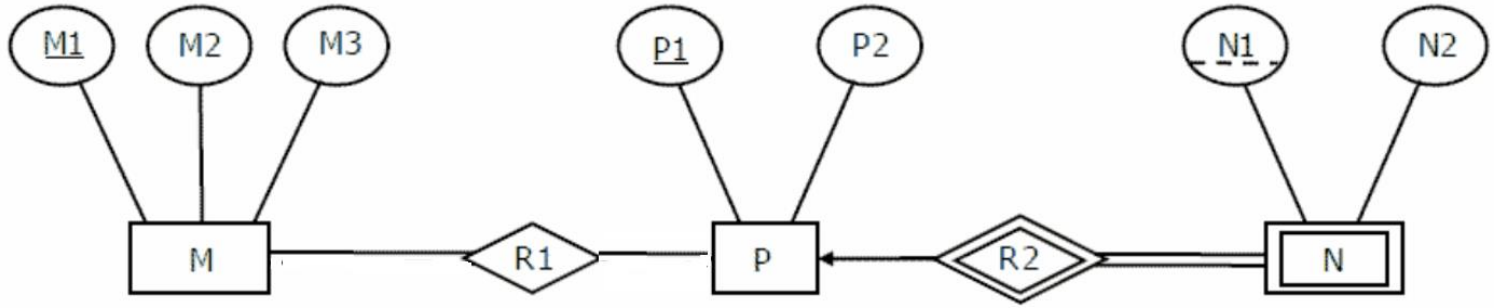


- M, P are strong entities hence they must be represented by separate tables.
- Many-to-one and one-to-many relationship sets that are total on the many-side
  - can be represented by adding an extra attribute to the "many" side,
  - containing the primary key of the "one" side.
  - This way no extra table will be needed for Relationship sets.
  - M table is modified to include primary key of P side(i.e. P1).
- N is weak entity, and is modified to include primary key of P (i.e, P1).

Therefore there would be minimum of 3 tables with schema as:

1. M (M1, M2, M3, P1)
2. P (P1, P2)
3. N (P1, N1, N2)

Example 2: The minimum number of tables are needed -

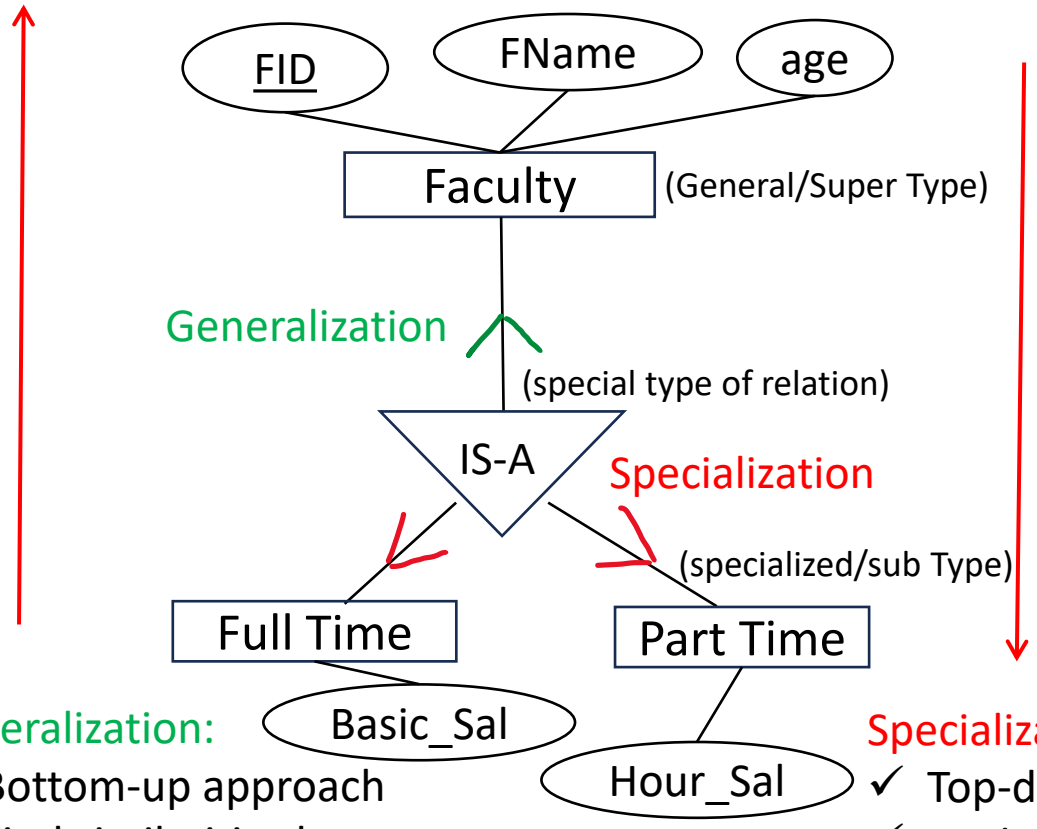


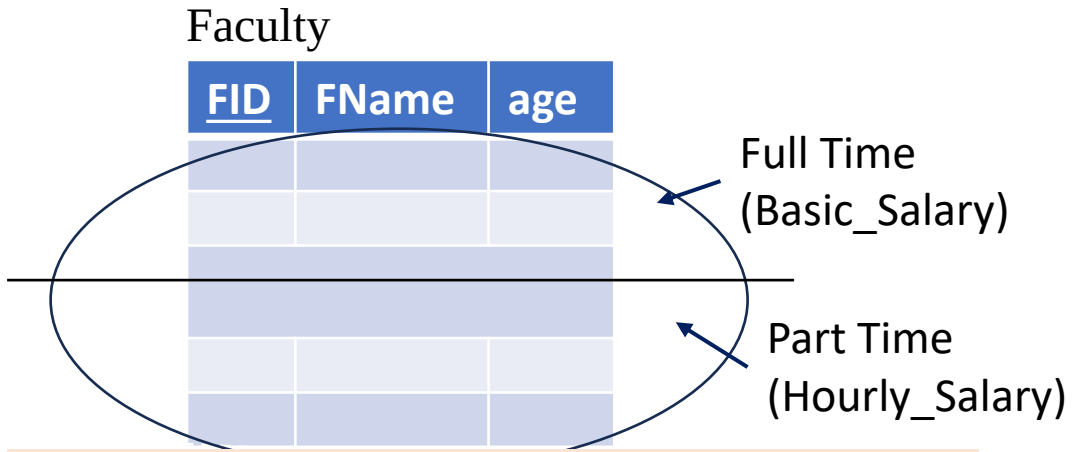
There would be minimum of 4 tables as :

1. M(M1, M2, M3)
2. P(P1, P2)
3. N(P1, N1, N2)
4. R1(M1, P1)

Extended E-R features: Generalization, Specialization and Aggregation

- Generalization/Specialization





Instances:
Assume that among all Faculty instances some of them are full time and others are part time

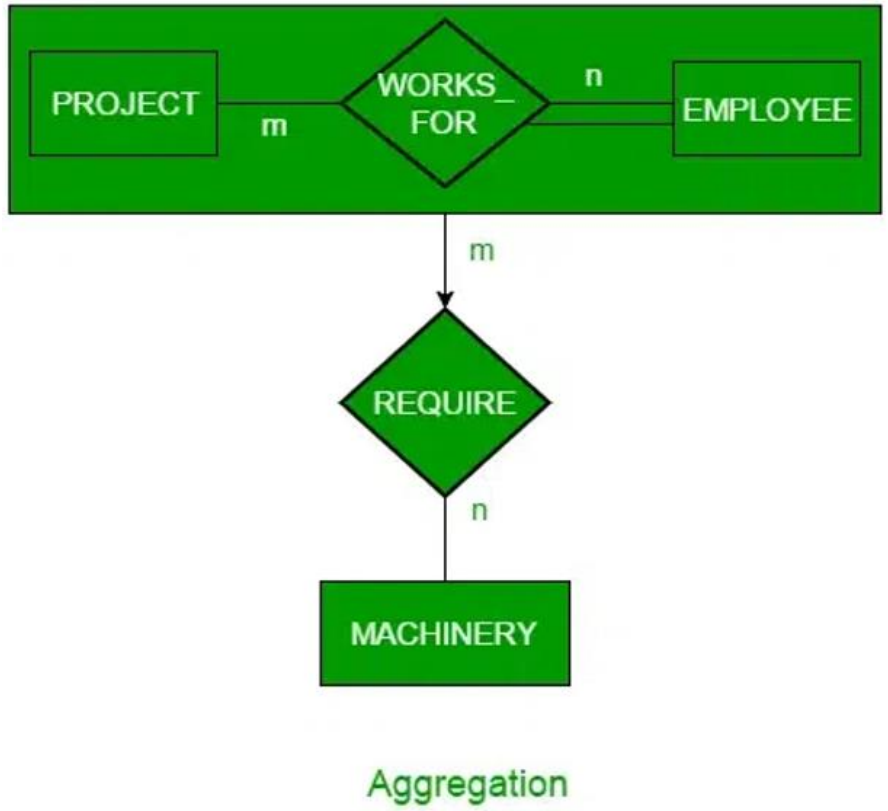
Generalization:
- ✓ Bottom-up approach
- ✓ Find similarities between instances of different entity type

Specialization:
- ✓ Top-down approach
- ✓ Find differences between instances of same entity type.

Attribute Inheritance:
Full Time entity has Basic_Sal attribute as well as Inheritance the attributes of super type, i.e., FID, Fname and age.

Aggregation

- It is an abstraction through which relationship is treated as high-level entity type.
- For Example,
  - An Employee working on a project may require some machinery.
  - So, REQUIRE relationship is needed between the relationship WORKS_FOR and entity MACHINERY.
  - But, we can not make a relationship between two relations.
  - Therefore, entities EMPLOYEE and PROJECT is aggregated into a single entity to make a relationship with MACHINERY.

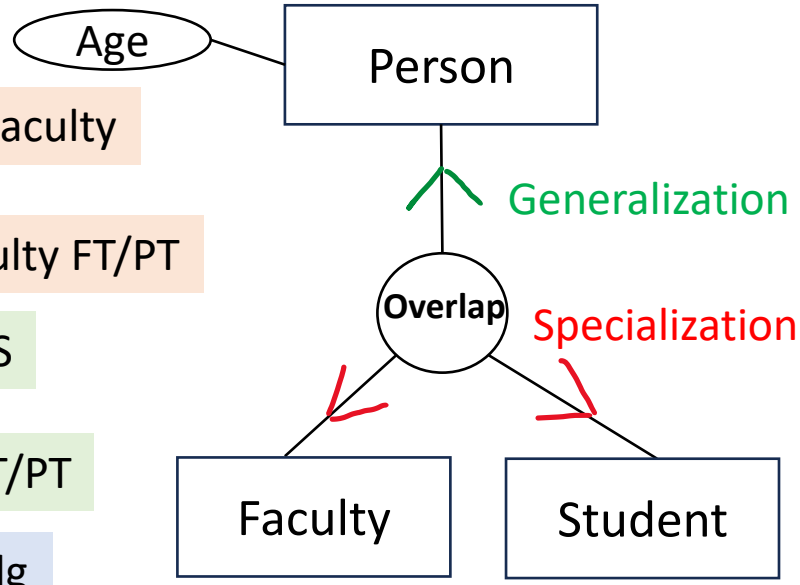


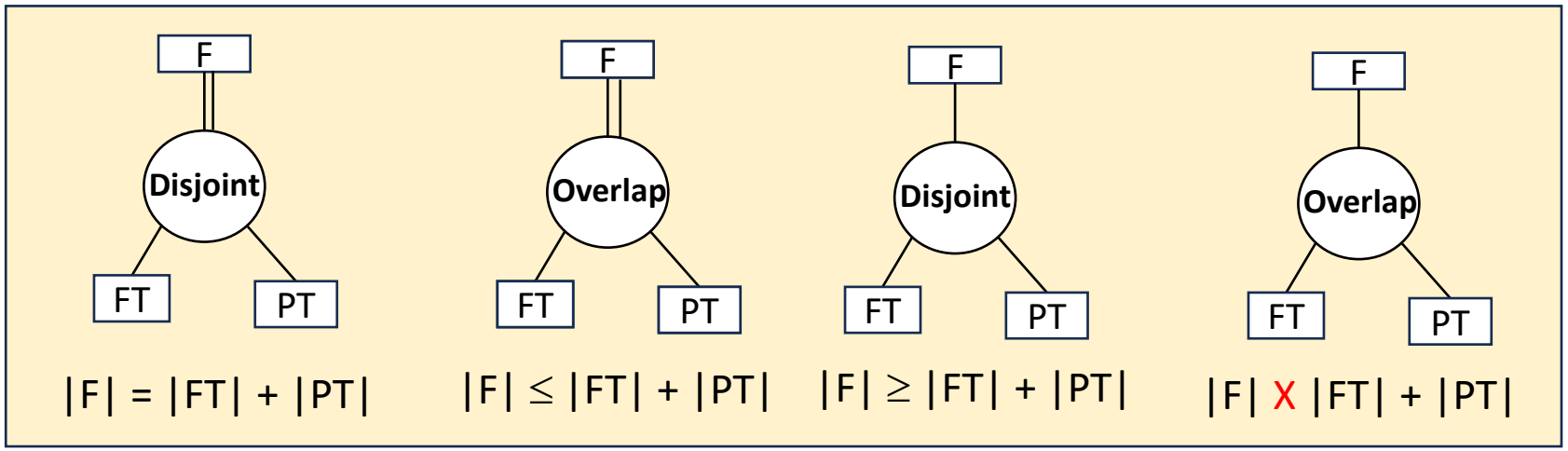
Aggregation

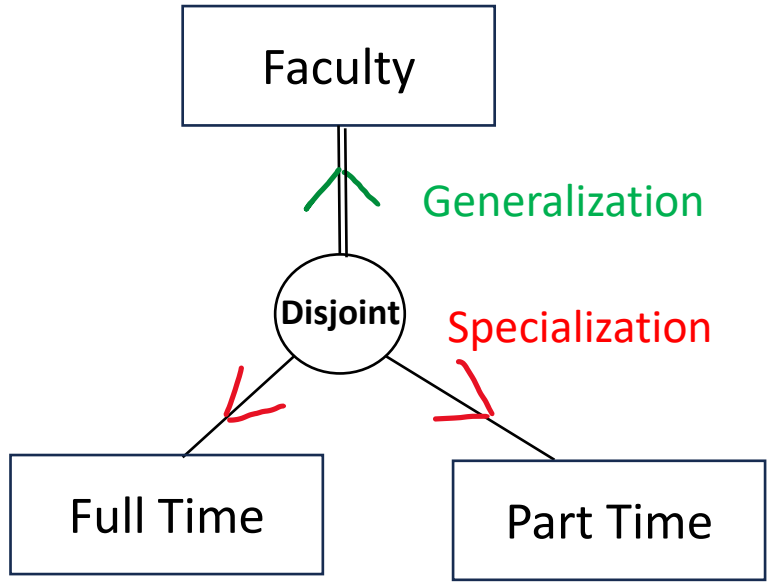
Constraints on Generalization and Specialization

- Three Types:

1. Membership Constraints
   1. Attribute Constraint — If Age>30 then Faculty
   2. User Constraint — Make random Faculty FT/PT
2. Disjointness Constraints
   1. Disjoint Constraint — Person can be F/S
   2. Overlapping Constraint — No Faculty FT/PT
3. Completeness Constraints
   1. Total G/S — PT/FT Faculty from own clg
   2. Partial G/S — PT Faculty from other clg



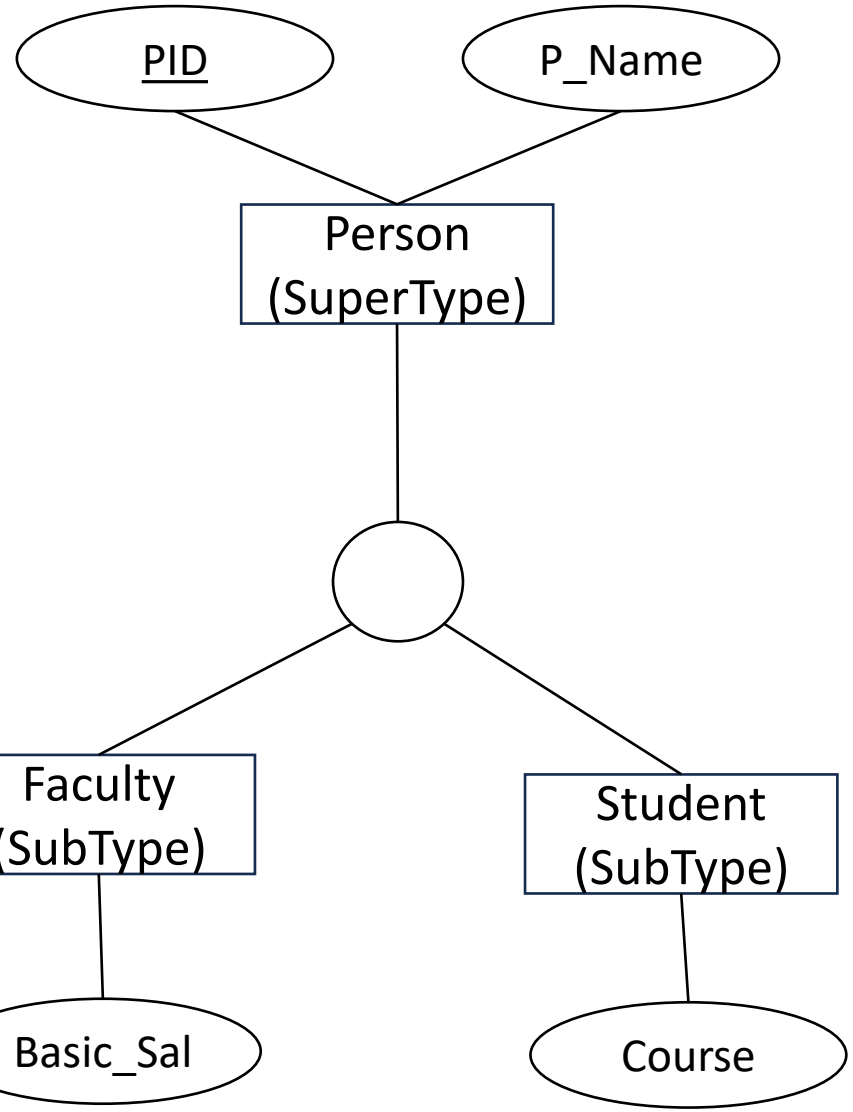




Converting Generalisation / Specialisation hierarchy to tables:

- Three ways to convert:
  1. Create individual tables for all

     It fulfils all constraints



Person

PID	P_Name
10	A

Foreign Key
- Apply constraint
a. Not null
b. Not Duplicate

Teacher

PID	Basic_Sal
10	25000
10	26000

Student

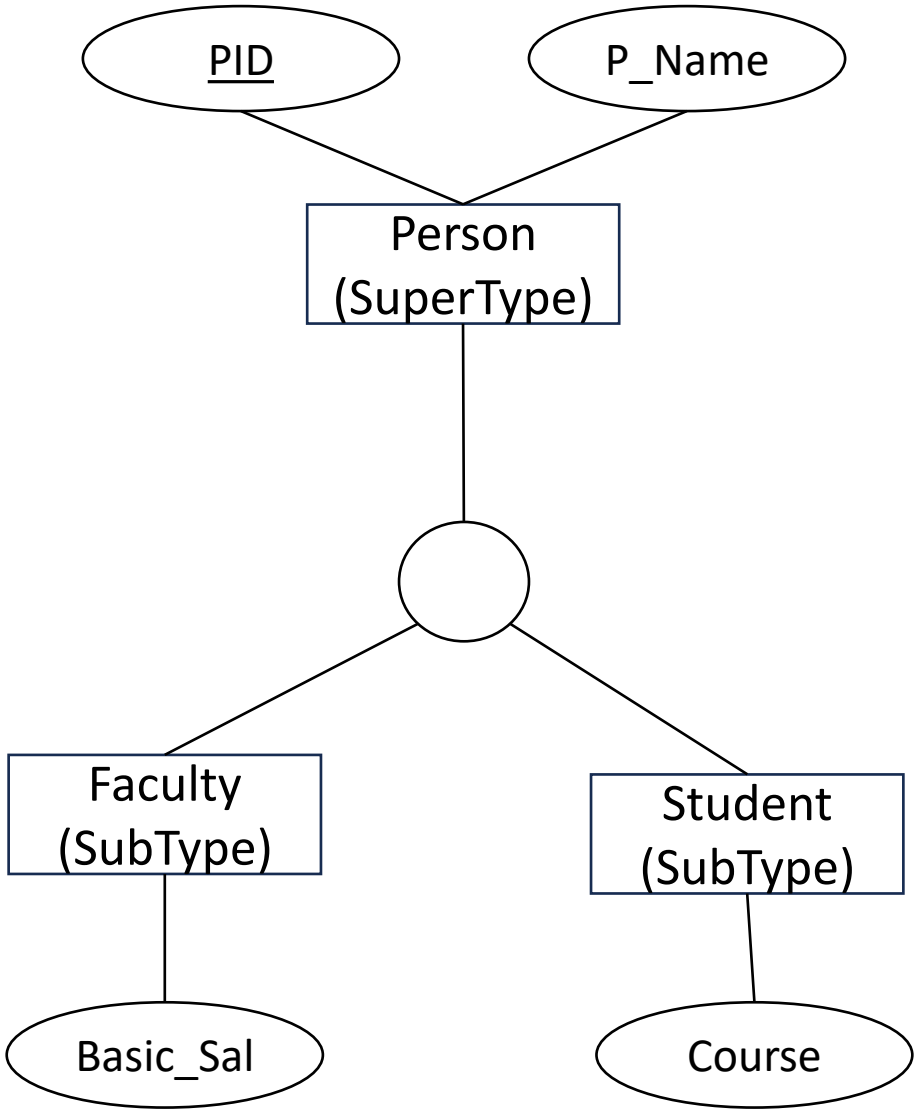
PID	Course

Converting Generalisation / Specialisation hierarchy to tables:

- Three ways to convert:
  2. Create only Sub Type tables

     It reduces the table, uses attribute inheritance

Issue: If person is not Faculty or Student, then we can not insert here.



Teacher

PID	P_Name	Basic_Sal

Student

PID	P_Name	Course

Converting Generalisation / Specialisation hierarchy to tables:

- Three ways to convert:
  3. Create only one tables

     Issue: To identify the person

Employee: New Table

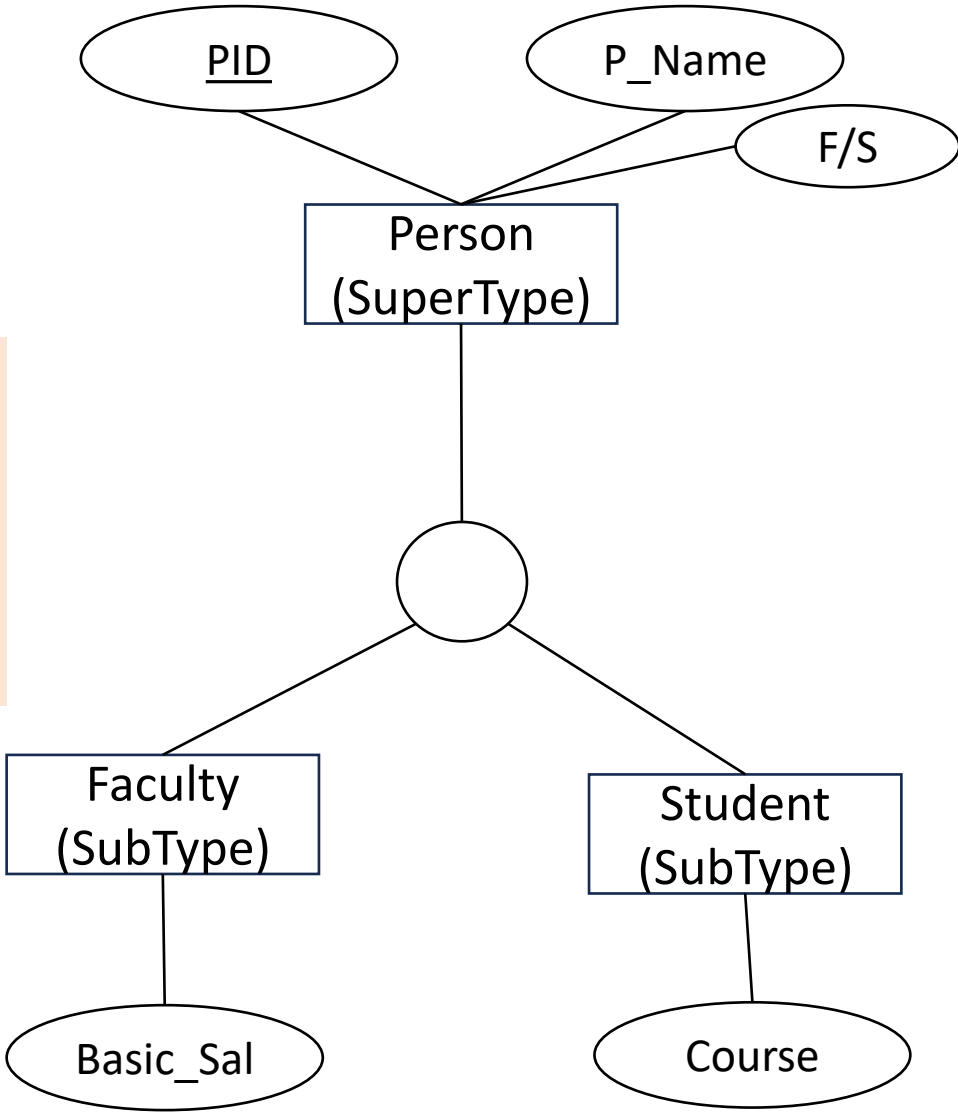
PID	P_Name	Basic_Sal	Course	F/S
10	A	35000	-	
12	B	-	MCA	
13	C	-	-	

SubType Discriminator-
It may be simple (in disjoint constraint) or composite (in overlap constraint)

Issues:

1. No F and No S
2. F but No S, do not know Salary
3. No F but S do not know course
4. Both but do not know salary/course

Sol: To add a special attribute, F/S



Assignment #1 Total marks= 10 Submission by 21/09/2023

- Design an ER model and convert it into tables of a dataset.
- You may choose any kind of organization as per your interest like Indian Railway, Hospital, Shop, etc.