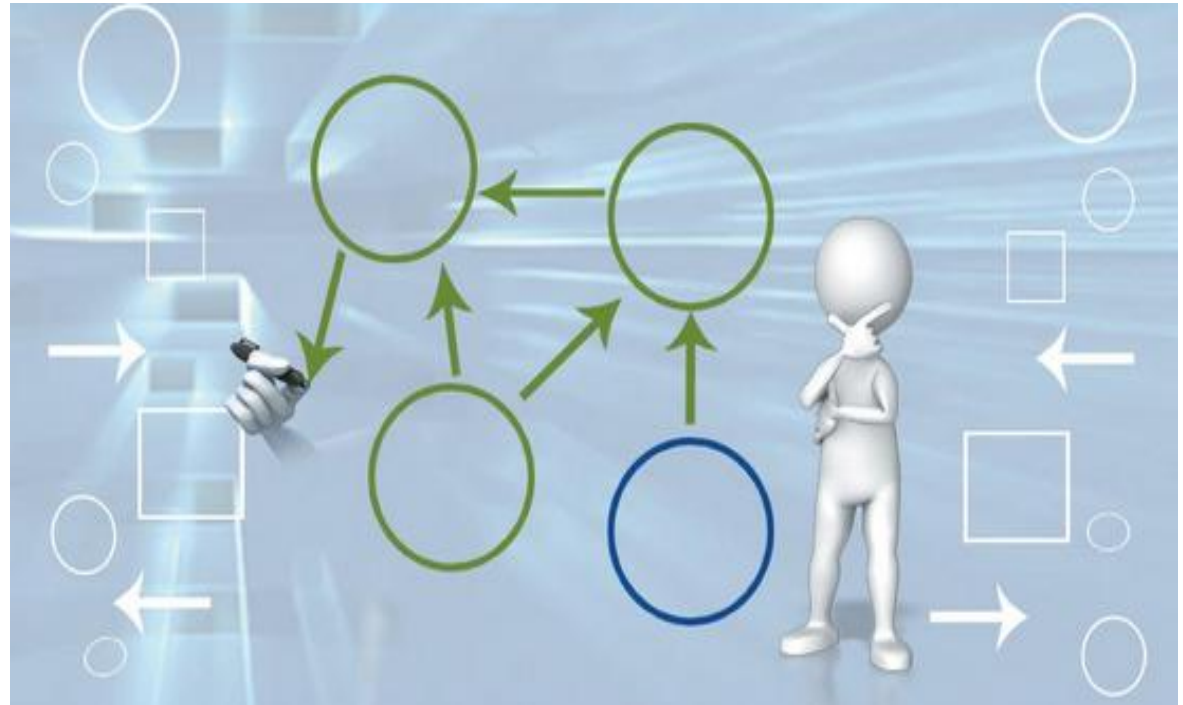
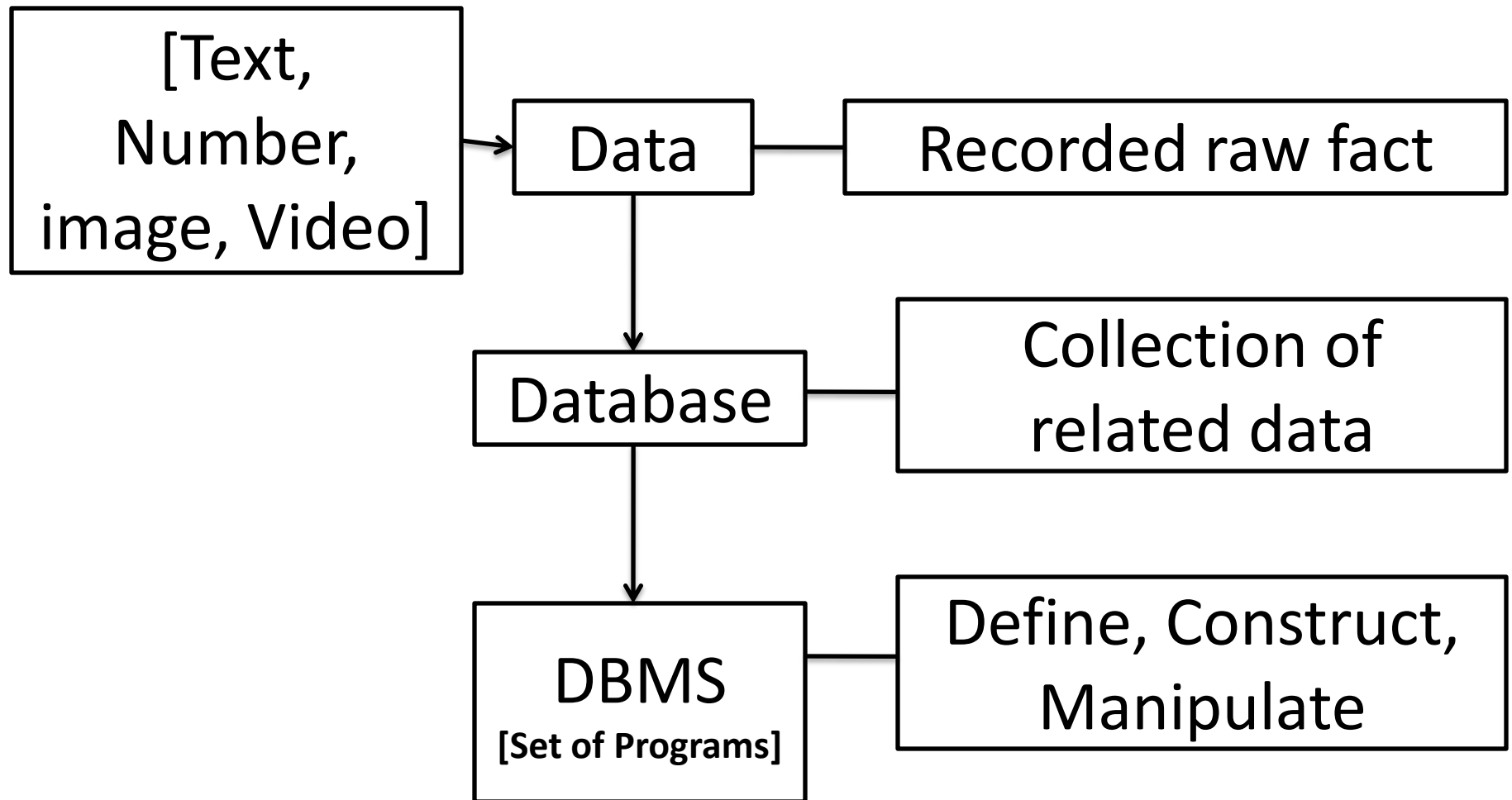


Topic – Entity Relationship Diagram(ERD)



Presented By –
M.M. Rakibul Hasan
Faculty, CSE, IUBAT University

Introduction





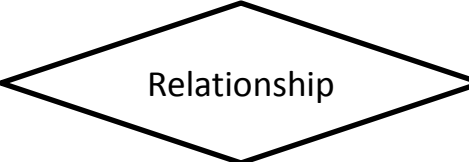
Why is ERD used?

- 1976 proposed by Peter Chen
- Entity Relationship Diagrams / ER Diagrams / ERDs are drawn to design a database schema for an intended software.
- ER diagram is widely used in database design.
- It is nothing but a conceptual design of the database, a blueprint for the its structure.

Main components in an ERD

ER diagram is the pictorial representation of real world objects, it involves various symbols and notation to draw the diagrams.

Main components are:

- Entity 
- Attribute 
- Relationship 

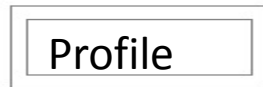
Entity

- **Entity**



- Entity can be any object, place, person or class. In a nutshell, it's a table in your database. It is represented using rectangle.
- Eg. Employee, Student, User.

- **Weak Entity**



- It is the same like an entity, the only difference being that it does not have a primary key, or its primary key is a composite of a column and a foreign key
 - You could say that weak entity depends on its parent entity for its existence
 - For e.g. A user has a profile, here profile will only exist if ever there is a user, hence profile is the weak entity.

N.B. A strong entity is represented by simple rectangle when a weak entity is represented by two rectangles.

Attributes

• Attributes

- An attribute is a column of an entity.
- It partly describes the entity in it's own way.
- An oval shape is used to represent the attributes. Name of the attribute is written inside the oval shape and is connected to its entity by a line.
- For example, name, salary, address are the attributes of an entity called Employee.
- **Types of Attribute:**
 - **Composite** vs Simple attribute
 - Single vs **Multivalued** attribute
 - Stored vs **Derived** attribute
 - Complex attribute
 - **Primary Key Attribute**



Attributes

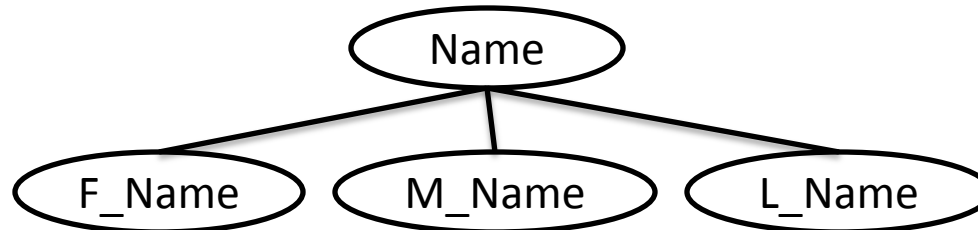
- **Simple Attributes**

- It is an attribute that has a single atomic value.
- eg. Gender of a Person.



- **Composite Attributes**

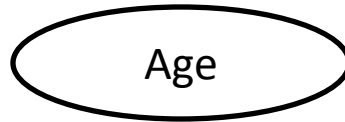
- A **composite attribute**, such as "Name", has multiple components such as "firstName", "MiddleName" and "lastName".
- A composite attribute is also represented by oval shape, but these attribute will be connected to its parent attribute forming a tree structure.



Attributes

- **Single Attributes**

- It is an attribute that generally contains only one value.
eg. Age of a Person.



- **Multivalued Attributes**

- It is an attribute that generally contains more than one value.
- Multivalued attributes are represented by double oval shape
- eg. Phone numbers of a user.



Attributes

- **Stored Attributes**

- It is an attribute that has some value.
- eg. Date of birth of a Person.



- **Derived Attributes**

- An attribute whose value is calculated (derived) from other attributes.
- This attribute is represented by dashed oval.



Attributes

- **Complex attributes**

- Collection Composite and Multivalued attribute.
- eg. Address of a Person.



- **Primary Key Attribute**

- A column which uniquely identifies a row in a table(entity) is called its primary key, its values have to be unique.
 - It could be a single column or a combination of two or more columns
 - The name of a key attribute is underscored.



Different Types of SQL Keys

1. Super Key

Super key is a set of one or more than one keys that can be used to identify a record uniquely in a table. **Example** : Primary key, Unique key, Alternate key are subset of Super Keys.

2. Candidate Key

A Candidate Key is a set of one or more fields/columns that can identify a record uniquely in a table. There can be multiple Candidate Keys in one table. Each Candidate Key can work as Primary Key.

Example: In below diagram ID, RollNo and EnrollNo are Candidate Keys since all these three fields can be work as Primary Key.

3. Primary Key

Primary key is a set of one or more fields/columns of a table that uniquely identify a record in database table. It can not accept null, duplicate values. Only one Candidate Key can be Primary Key.

4. Alternate key

A Alternate key is a key that can be work as a primary key. Basically it is a candidate key that currently is not primary key.

Example: In below diagram RollNo and EnrollNo becomes Alternate Keys when we define ID as Primary Key.

Different Types of SQL Keys

5. **Composite/Compound Key**

Composite Key is a combination of more than one fields/columns of a table. It can be a Candidate key, Primary key.

6. **Unique Key**

Unique key is a set of one or more fields/columns of a table that uniquely identify a record in database table. It is like Primary key but it can accept only one null value and it can not have duplicate values.

7. **Foreign Key**

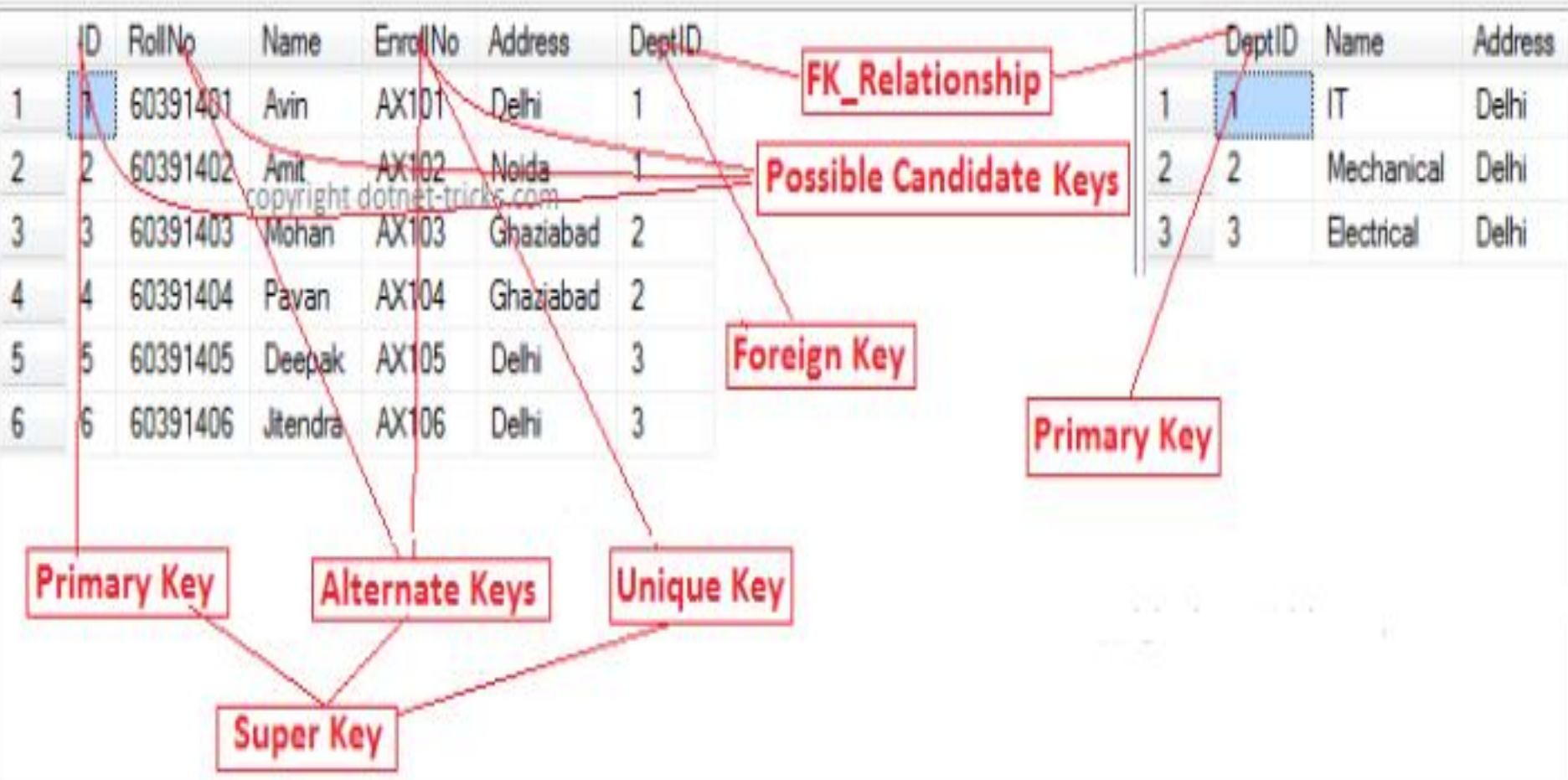
Foreign Key is a field in database table that is Primary key in another table. It can accept multiple null, duplicate values.

Example : We can have a DeptID column in the Employee table which is pointing to DeptID column in a department table where it a primary key.

Different Types of SQL Keys

Student Table

Department Table



Relationships

- **Relationships**

- A relationship describes relations between entities. Relationship is represented using diamonds.
- In practice there can be a relationship between two tables if and only if they have something in common, i.e a common column.

Relationships

- **Strong relationship**

- A relationship where entity is existence-independent of other entities, and PK of Child doesn't contain PK component of Parent Entity. A strong relationship is represented by a single diamond.



- **Weak (identifying) relationship**

- A relationship where Child entity is existence-dependent on parent, and PK of Child Entity contains PK component of Parent Entity. This relationship is represented by a double diamond.

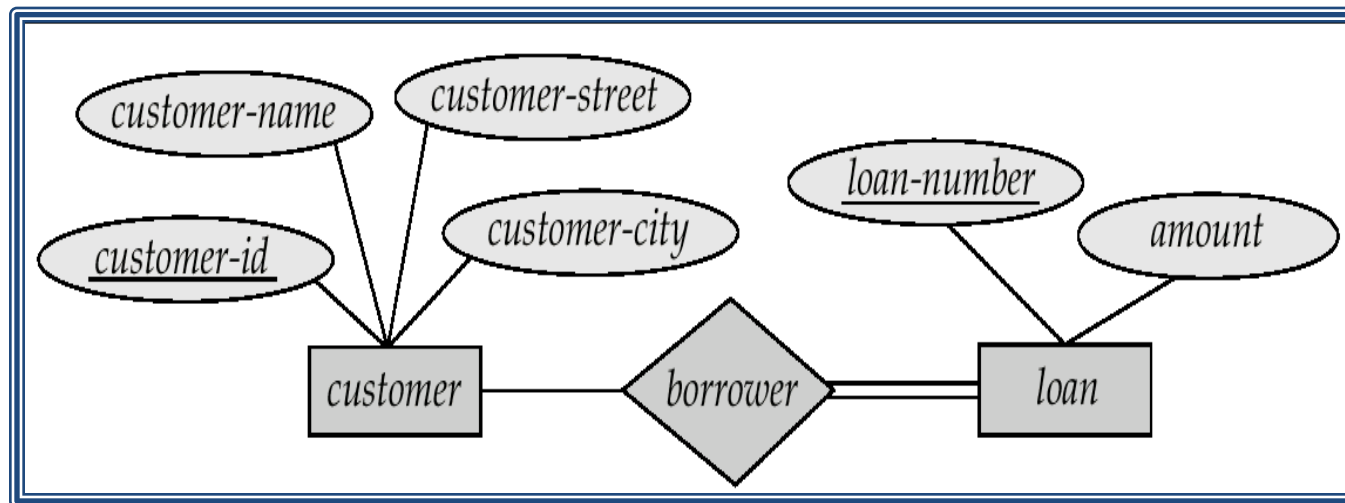


Participation Constraints

- **Total vs Partial Participation**

When we require all entities to participate in the relationship (total participation), we use double lines to specify. On the other hand Partial participations are shown as single line.

Every loan has to have at least one customer

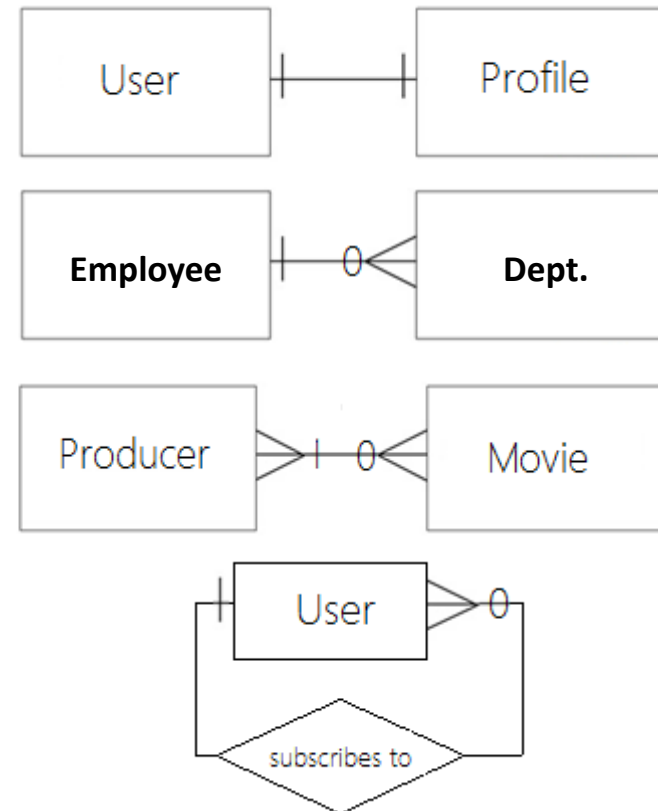


Cardinality of Relationship

Name

- One-to-one
- One-to-many
- Many-to-many
- Many-to-many (recursive)

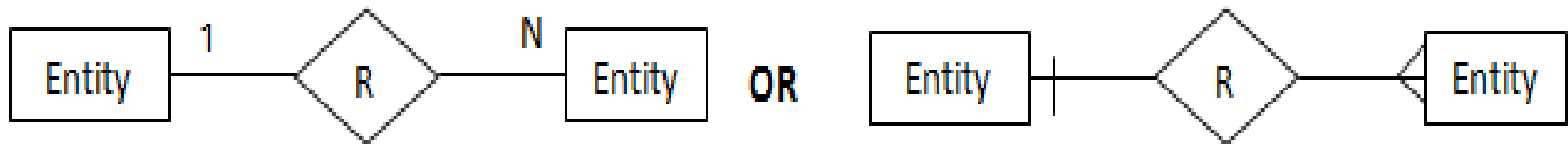
Example



Cardinality of Relationship

- **One-to-Many relation**

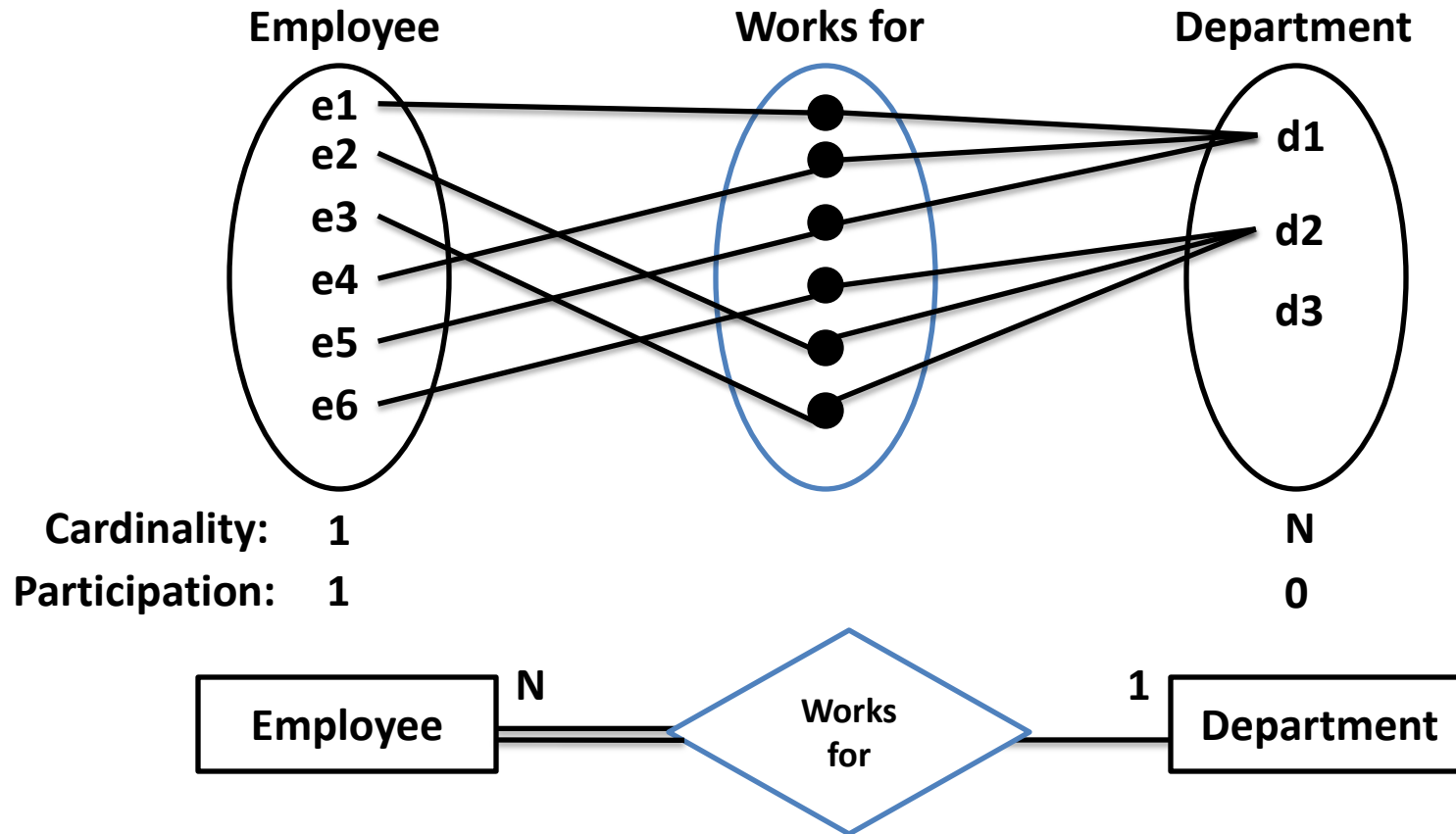
- A one-to-many relationship is represented by adding '1' near the entity at left hand side of relation and 'N' is written near the entity at right side. Other type of notation will have dash at LHS of relation and three arrow kind of lines at the RHS of relation as shown below.



Cardinality of Relationship

- **One-to-Many relation**

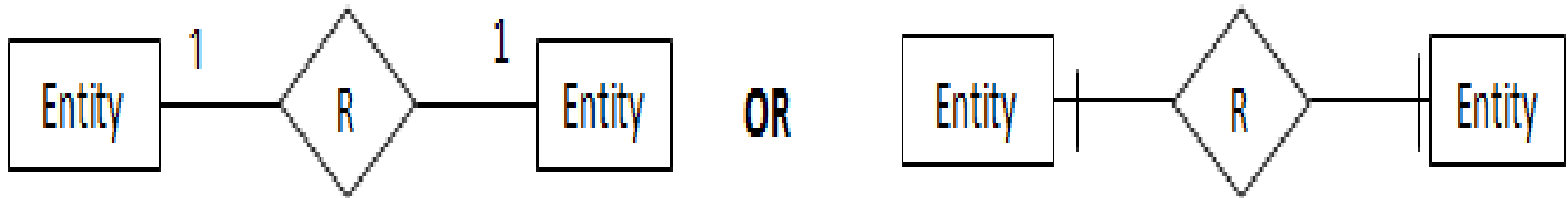
User Requirement- Every employee works for a department and a department can have many employees. New department need not have any employee.



Cardinality of Relationship

- **One-to-one relation**

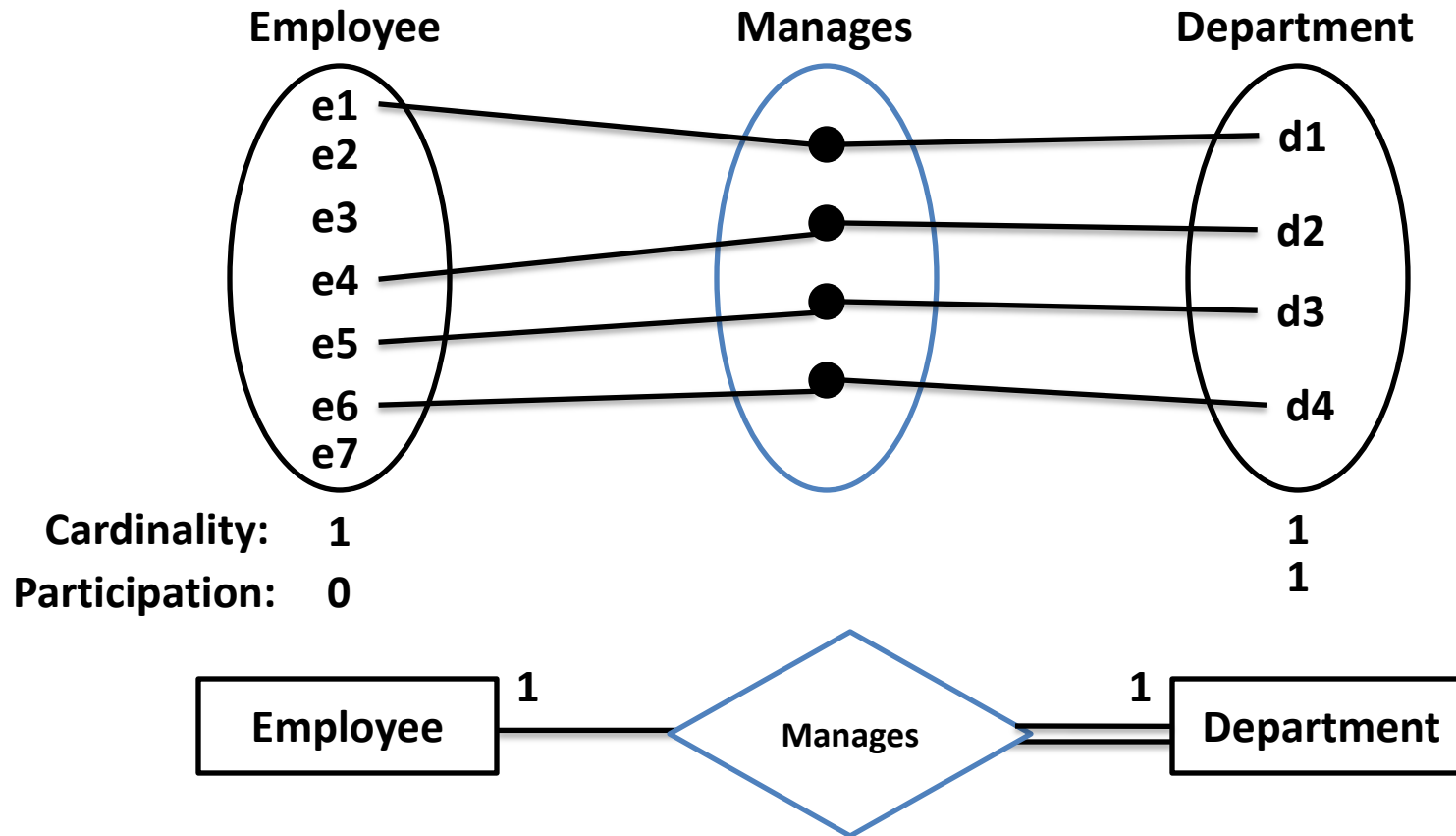
- A one-to-one relationship is represented by adding '1' near the entities on the line joining the relation. In another type of notation one dash is added to the relationship line at both ends.



Cardinality of Relationship

- One-to-one relation

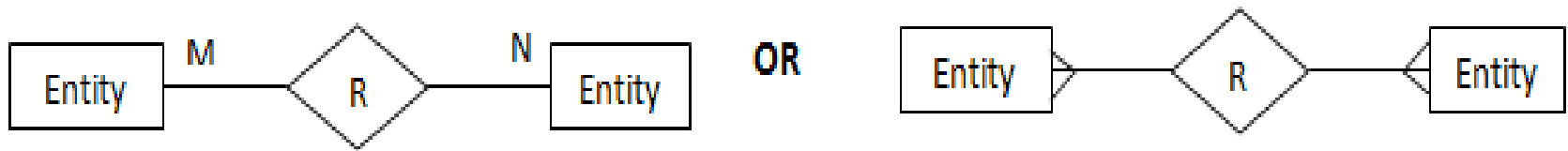
User Requirement- Every department should have a manager and only one employee manages a department. And an employee can manage only one department.



Cardinality of Relationship

- **Many-to-Many relation**

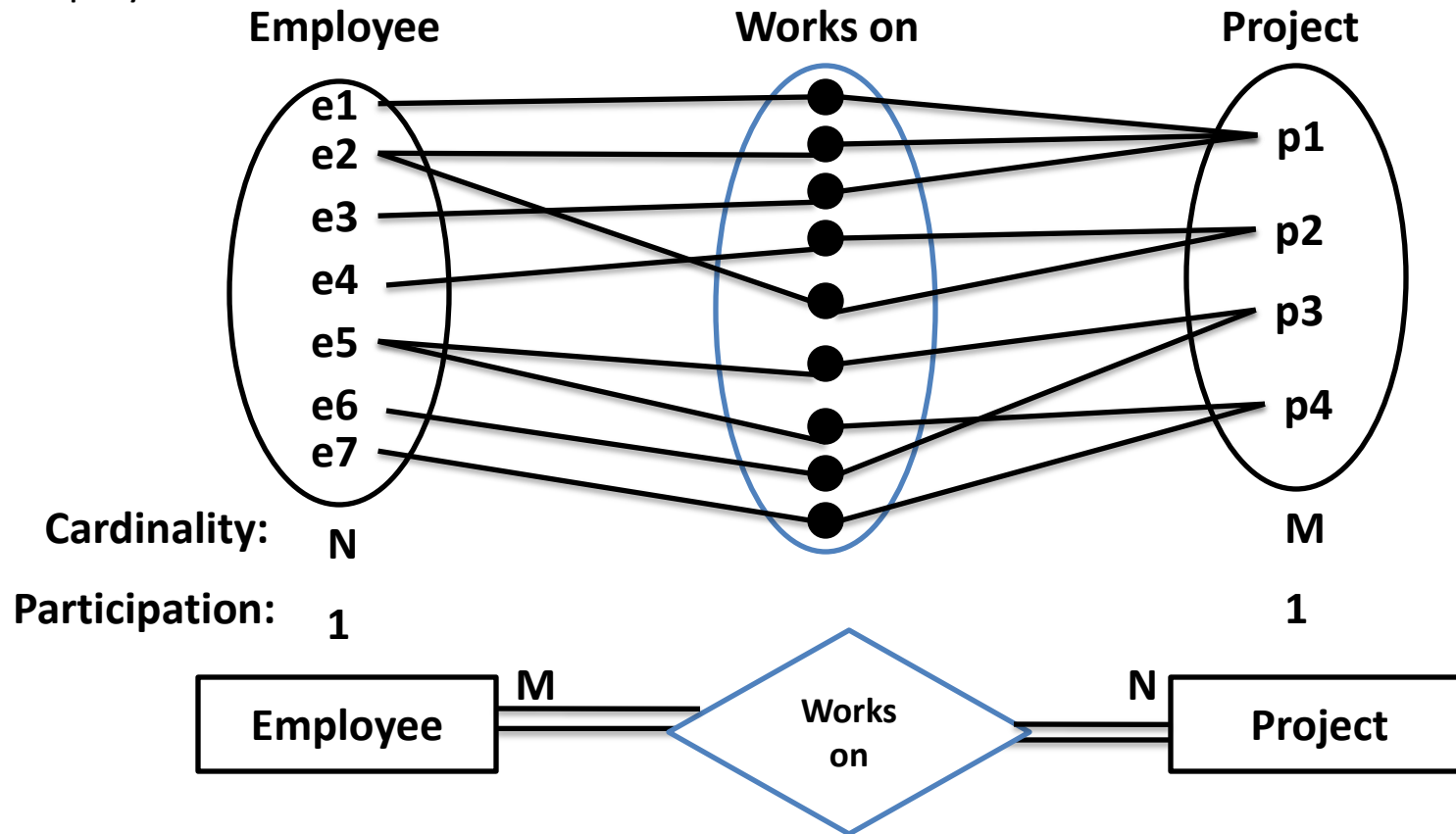
- A one-to-many relationship is represented by adding 'M' near the entity at left hand side of relation and 'N' is written near the entity at right side. Other type of notation will have three arrow kinds of lines at both sides of relation as shown below.



Cardinality of Relationship

- **One-to-one relation**

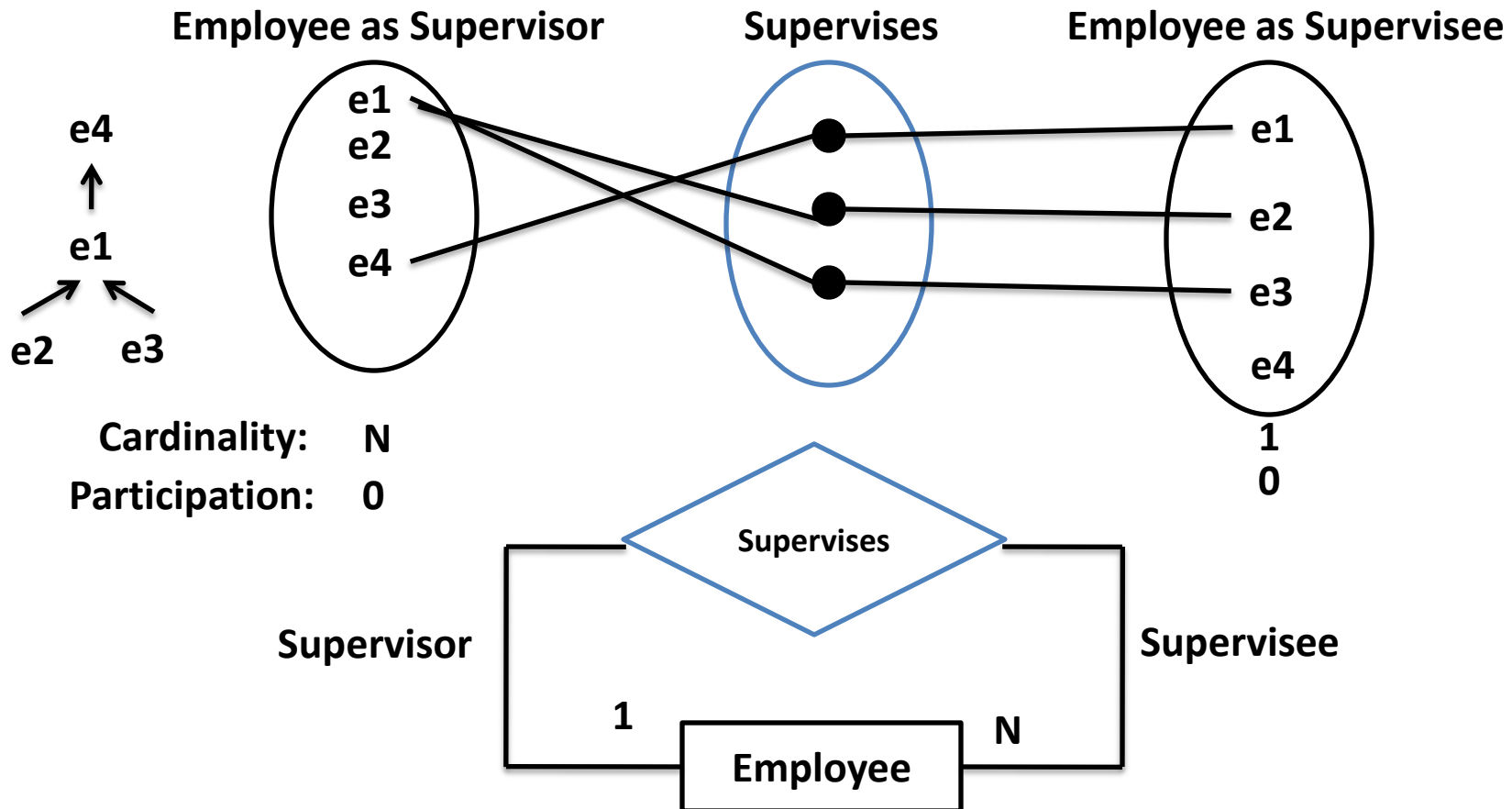
User Requirement- Every employee is suppose to work on at lest one project and he can works on many projects. Every project is suppose to have many employees but at lest one employee .



Cardinality of Relationship

- Many-to-many (recursive)

User Requirement- Every employee works under a supervisor except CEO.



Ground-rules

- A given entity should appear only once in one entity relationship diagram.
- Don't draw foreign key if it is already represented by a primary in the other table.
- Never connect a relation with another relation.
- Do not show relationships between two entities if there exist no common columns.

Complete ER diagram

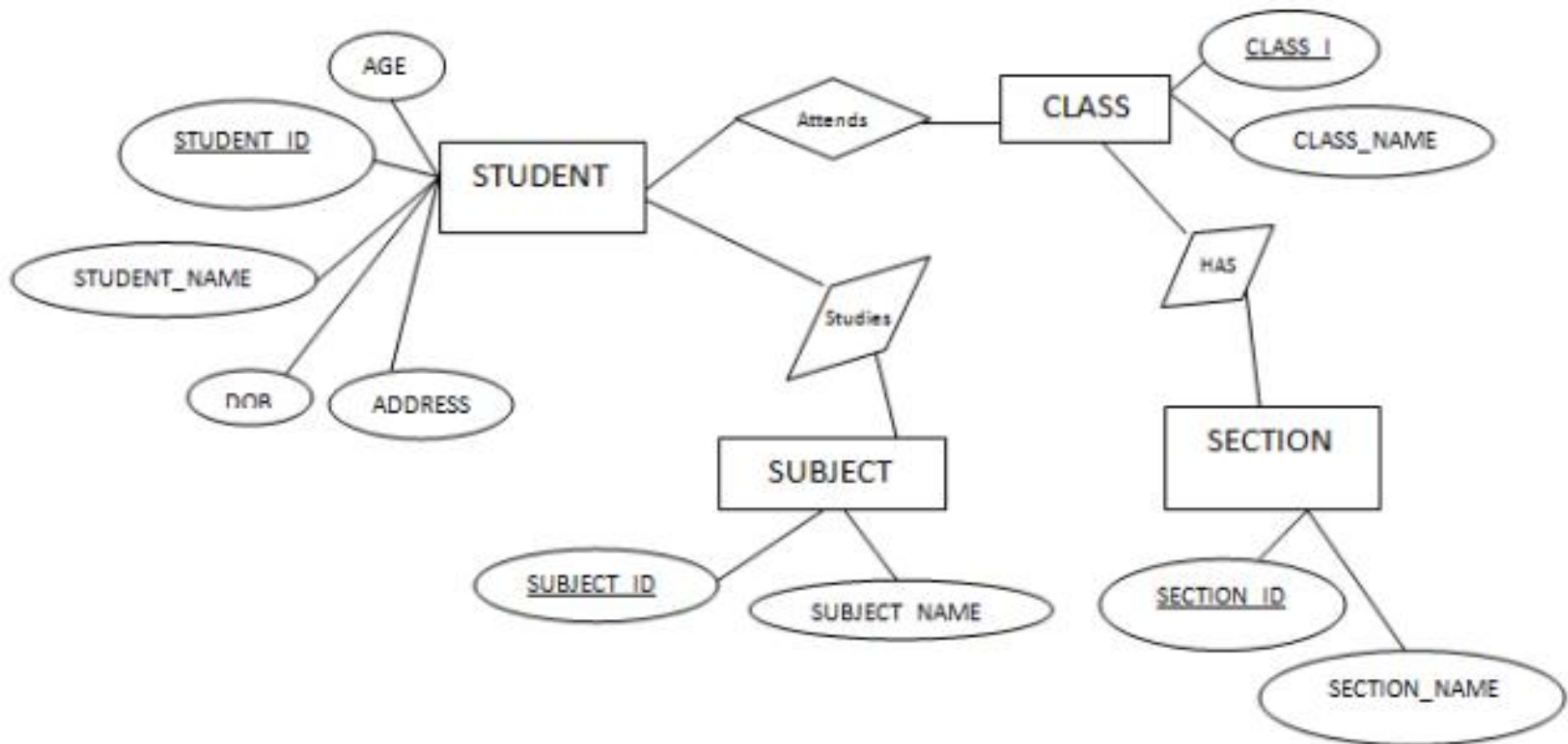
- Student attends class. Each class is divided into one or more sections.
- Each class will have its own specified subjects. Students have to attend all the subjects of the class that he attends.
- Now let us identify what are the entities?
 - **STUDENT, CLASS, SECTION, SUBJECT** are the entities.

- We can list the attributes as below:

STUDENT	CLASS	SECTION	SUBJECT
<u>STUDENT_ID</u>	<u>CLASS_ID</u>	<u>SECTION_ID</u>	<u>SUBJECT_ID</u>
STUDENT_NAME	CLASS_NAME	SECTION_NAME	SUBJECT_NAME
ADDRESS			
DOB			
AGE			
CLASS_ID			
SECTION_ID			

- What are the relationships we have? '
 - **Attends', 'has section', 'have subjects' and 'studies subjects'**

Complete ER diagram

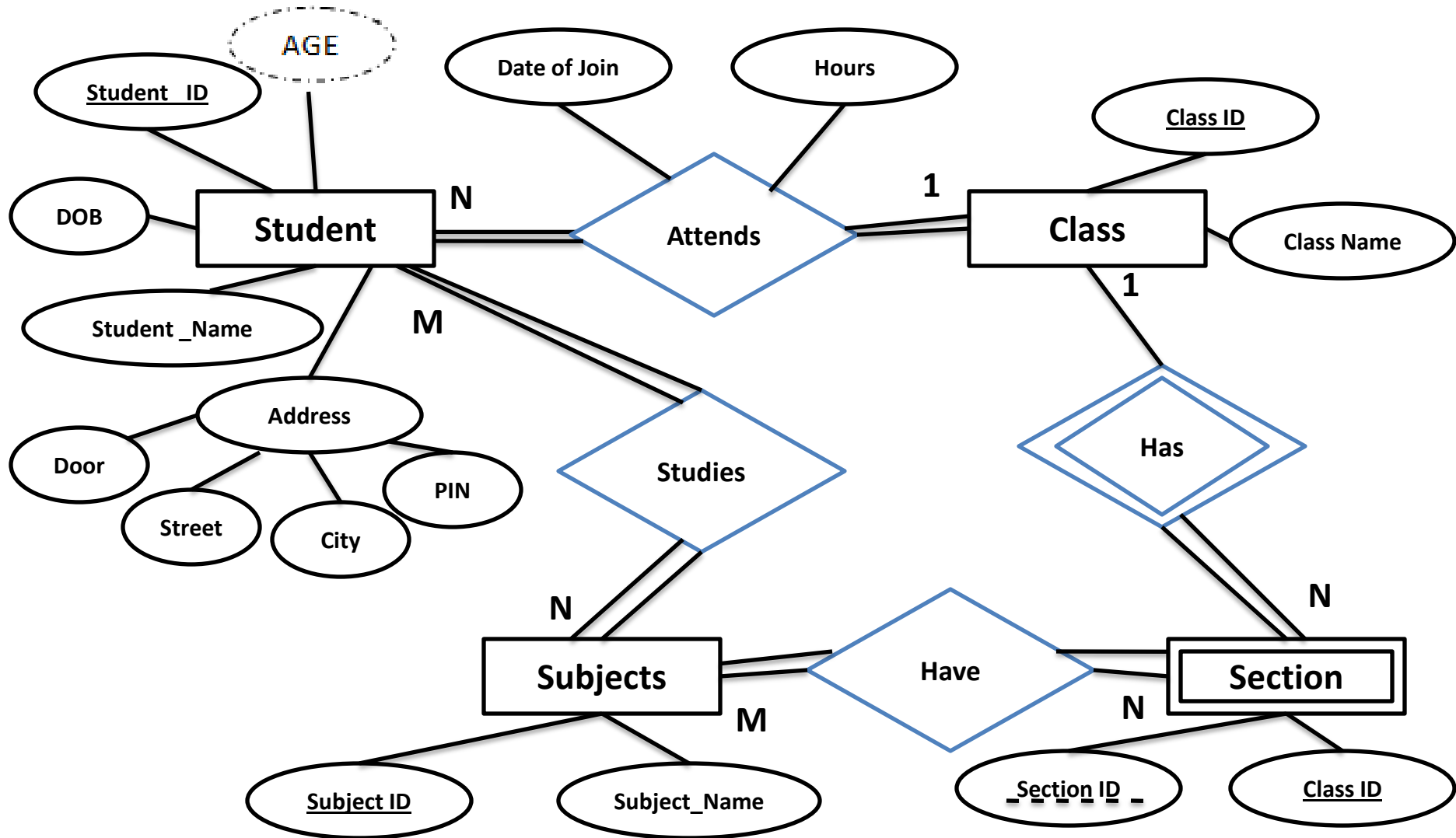


Complete ER diagram

Observe the diagram carefully. Are we missing anything on the diagram? Is it inferring correct requirement? What are our observations?

- Age attribute can be derived from DOB. Hence we have to draw dashed oval.
- Address is a composite attribute. We have to draw its sub attributes too. So that we will be very clear about his address details.
- If we see the SECTION entity, by section id, will we be able get the section that student attends? There is no relation mentioned between Student and Section. But Section is mapped only with Class. What do we understand from this? Section is a weak entity. Hence we have to represent it properly.
- If we look at 'attends' relationship between STUDENT and CLASS, we can have 'Joining Date' and 'Total Number of Hours' attributes. But it is an attribute of relation. We have to show them in the diagram.

Complete ER diagram



Thank You