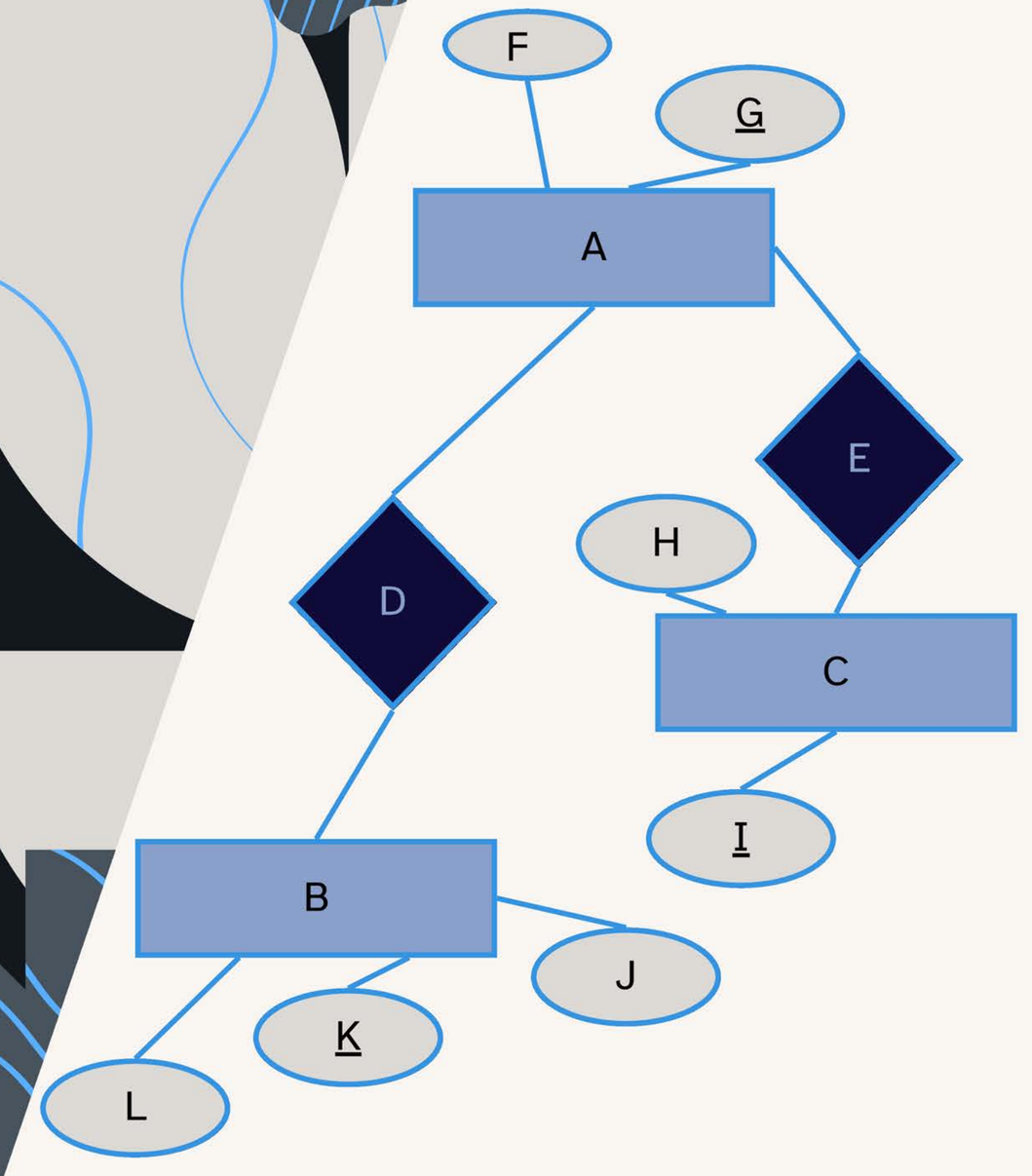
BRAC UNIVERSITY

Inspiring Excellence

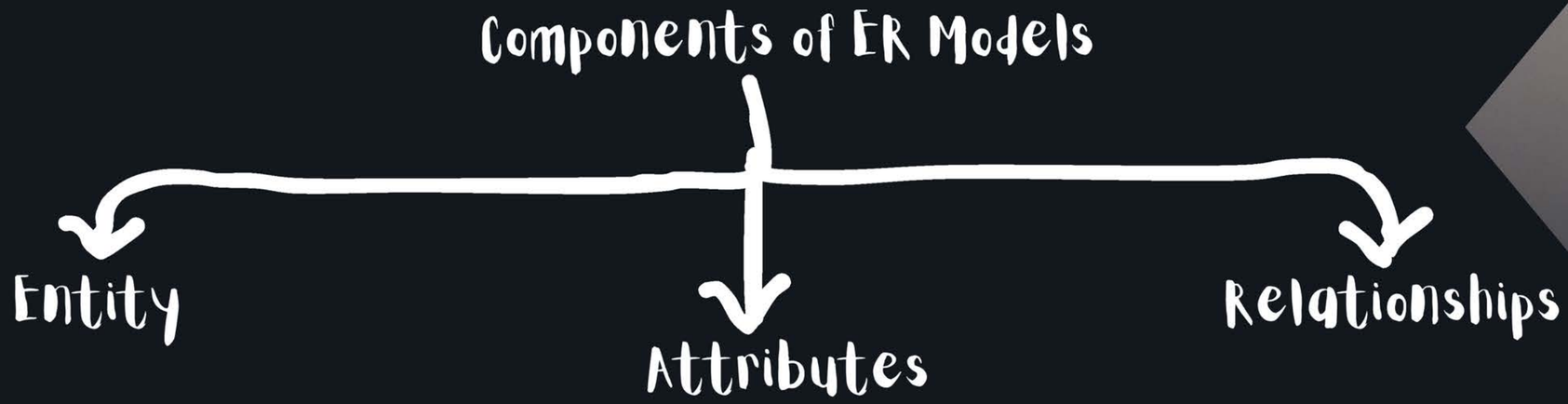
**CSE370: DATABASE SYSTEMS**

# Lecture 2: Data Modeling Using ER Model



# Introduction

Understanding **Entity-Relationship (ER)** models is crucial for effective database design. They provide a clear visual framework for mapping out data relationships and structures.



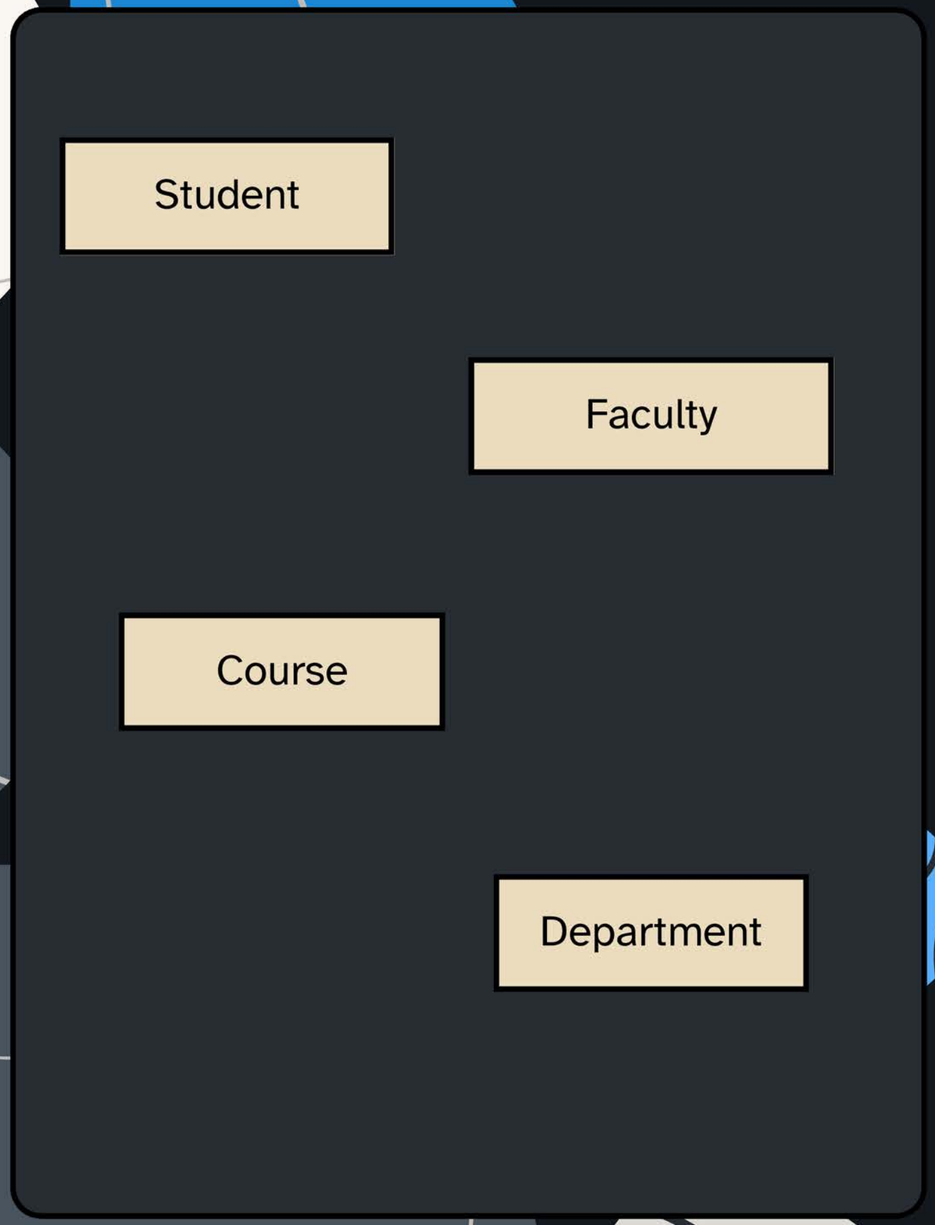

Note: For constructing the ER diagram, we will be using the Chen notation. There are many other notations, such as crow's foot notation or UML class diagram notation, used in different literature or design tools.

# The Mini World

The **mini world** concept is essential in ER modeling. It helps in identifying and defining the relevant entities and relationships required for database design.

**Mini world** is some part of the real world about which data is stored in a database. For example, if we want to create a database for train reservation in Bangladesh, then the train reservation system is the mini-world. Similarly, if we want to create a database for a university, then that University is the mini-world that we will represent in our database.

Note, the mini-world is not shown on the ER diagram.




# Entities and Entity Types

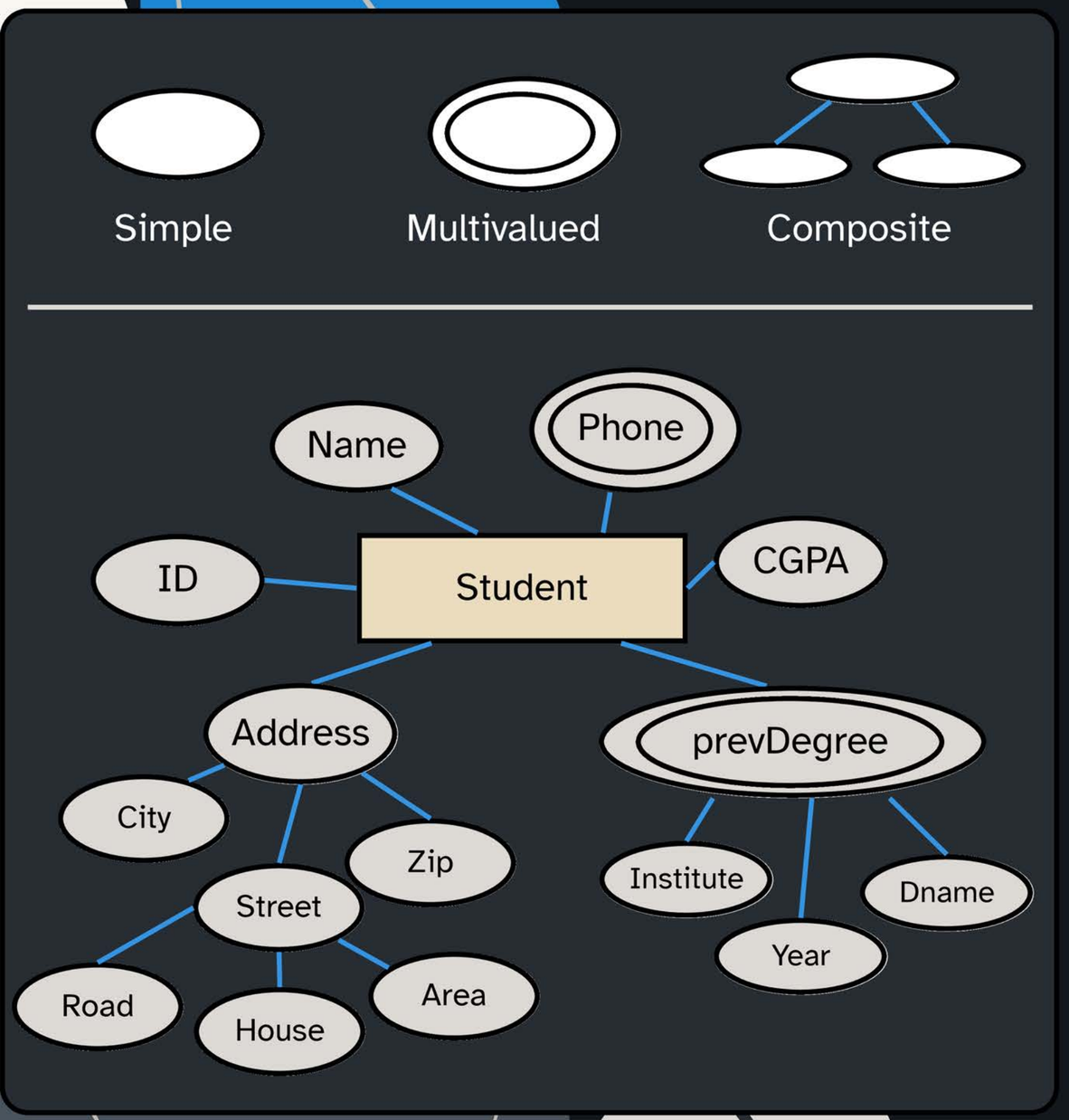
**Entities** are specific objects or things in the mini-world that are represented in the database.

For example, the STUDENT Sakib Chowdhury, the CSE DEPARTMENT or the course CSE370 are all entities in the University mini-world. Thus, if there are 5000 students enrolled in the university, then they are all 5000 individual entities.

All these 5000 students have similar type of information stored about them and they have the same role within the mini-world, so they can be grouped together into the Student entity type. So, when several entities are grouped together due to sharing the same properties and role then it is called the **Entity Type.**

Entity types are shown in ER diagram using a **“rectangle”** shape.

# Attributes (1)




**Attributes** are properties used to describe an entity. For example, a STUDENT may have attributes such as id, name, cgpa, email, etc. A specific entity will have a value for each of its attributes. There are 3 types of Attributes:

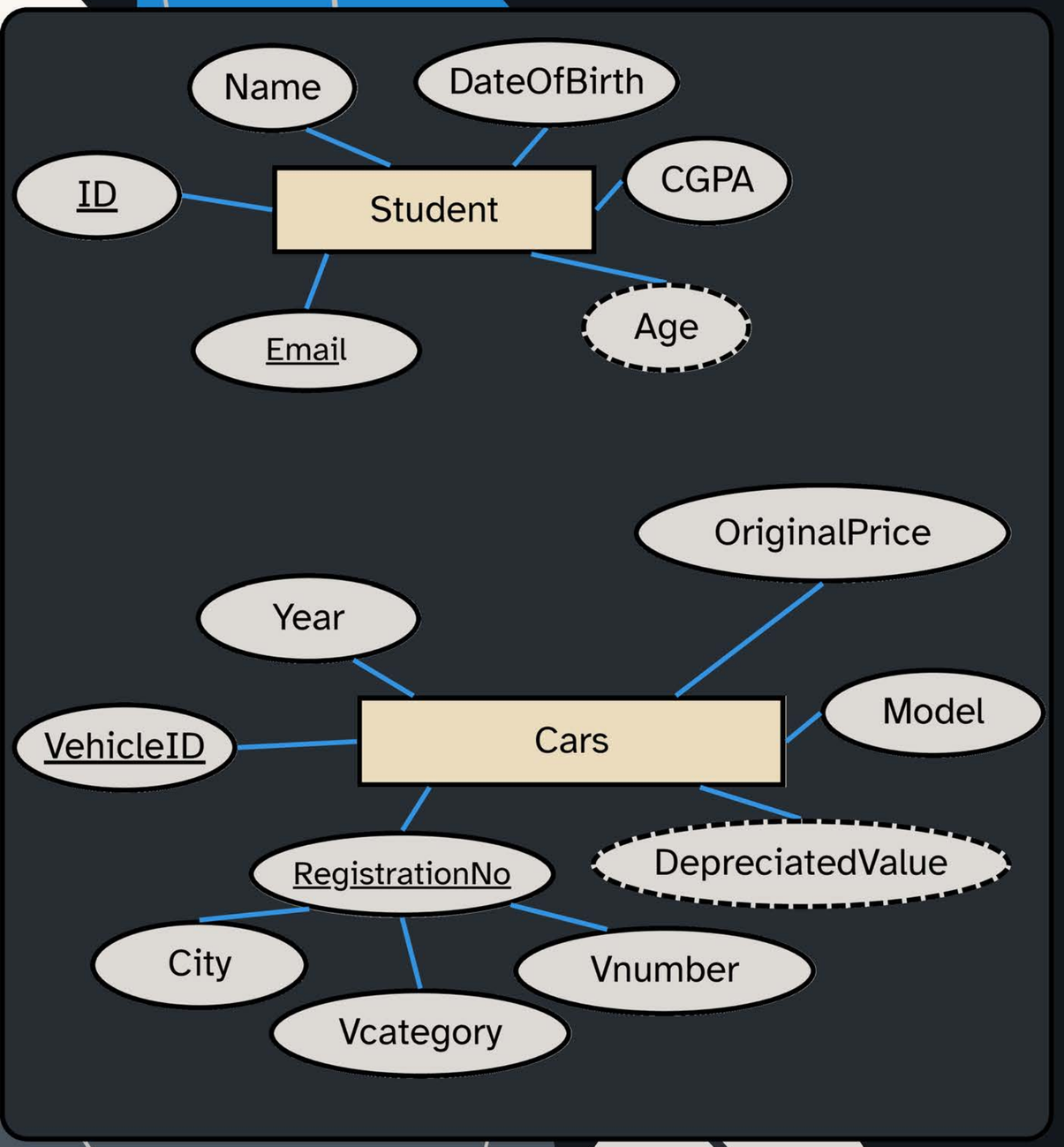
- **Simple:** Each entity has a single atomic value for the attribute. For example, STUDENT id, cgpa, EMPLOYEE salary. It is shown using an **“oval”** shape in the ER diagram.

- **Multivalued:** An entity may have multiple values for that attribute. For example, color of a CAR or email of a STUDENT. It is shown using a **“double oval”** shape in the ER.

- **Composite:** Each value of the attribute is composed of several components. For example, Address(Apt#, House#, Street, City, State, ZipCode, Country), or Name(FirstName MiddleName LastName). Some components may themselves be composite. **“Ovals”** are connected to other **“ovals”** in the ER.

An attribute can be composite-multivalued, for example, previous degrees of a STUDENT.

# Attributes (2)




**Key Attribute**

An attribute of an entity type for which each entity must have a unique value is called a key attribute of the entity type. For example, Student ID or email, Course code.

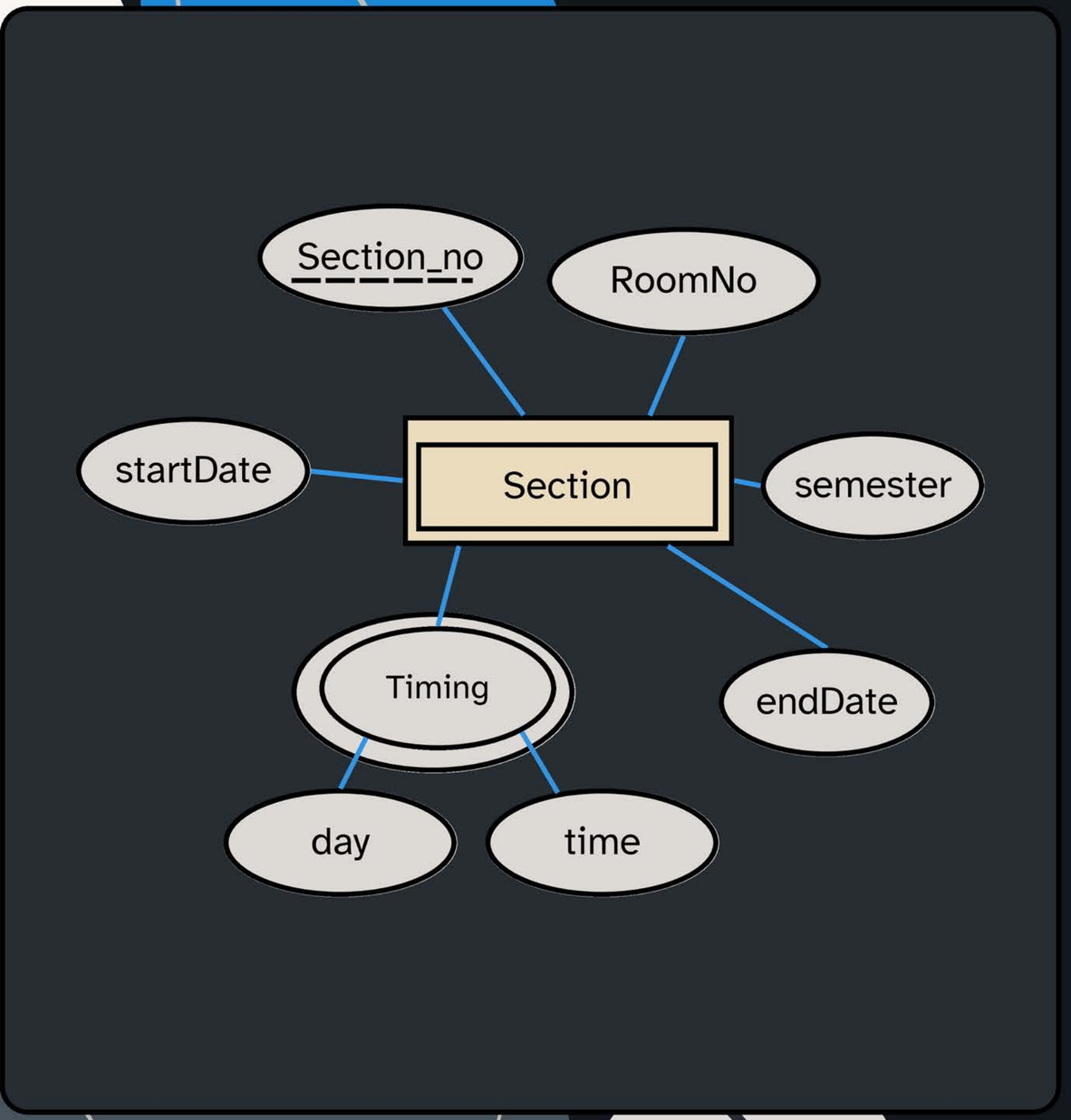
Key attributes are **“underlined”** in the ER diagram. An entity can have more than one key attribute. A key attribute should not be multivalued.

**Derived Attribute**

An attribute value that can be calculated/derived from other stored data, it is called a derived attribute. It means the value will not be stored in the database, but instead will be derived when needed in order to save space.

For example, “total bill” of an ORDER when the unit price and quantity is stored, “age” of a STUDENT when birthdate is stored. Derived attribute is shown using a **“dotted oval”** in ER diagram.

# Weak Entity

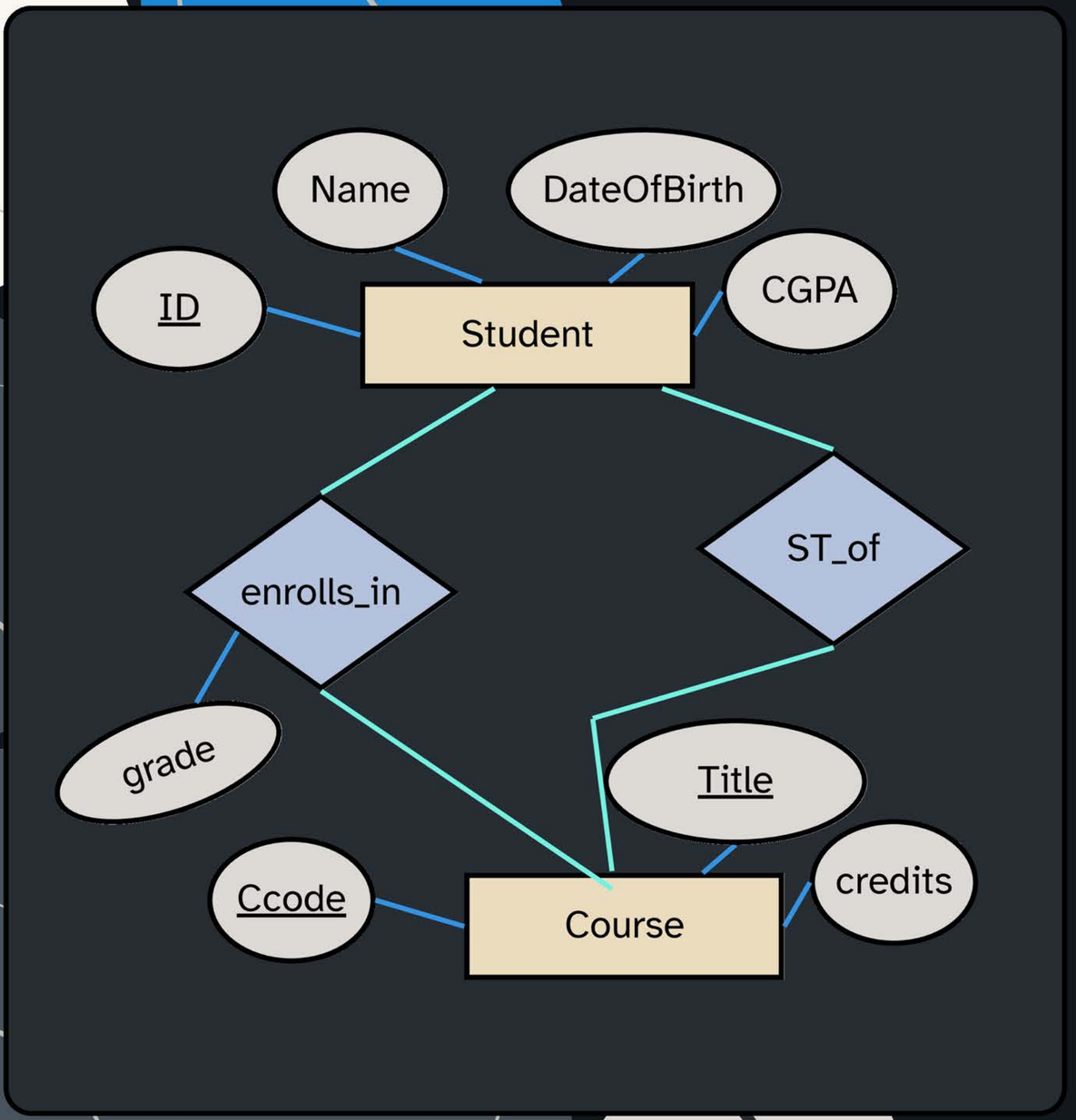



- Sometimes an entity may not have any unique (key) attributes. In such cases the individual entities cannot be uniquely identified using its own attributes. Such entities belong to **weak entity** types. The example of SECTIONS in a university mini-world (on the left) illustrates a weak entity type.

  Weak entity types are shown using a **“double rectangle”** in the ER diagram and such an entity type must not have any key attributes.

- **Partial Key**
  A weak entity may not have a key attribute, but it may have an attribute that is “part” of a unique key/value. Such an attribute is called a **partial key** and is shown using a **“dotted underline”**. On the left Section Number is not unique as many sections (of different courses) will have the same number. But the section number is part of the key that will be used to identify a particular section. (More on this later.....)

# Relationships

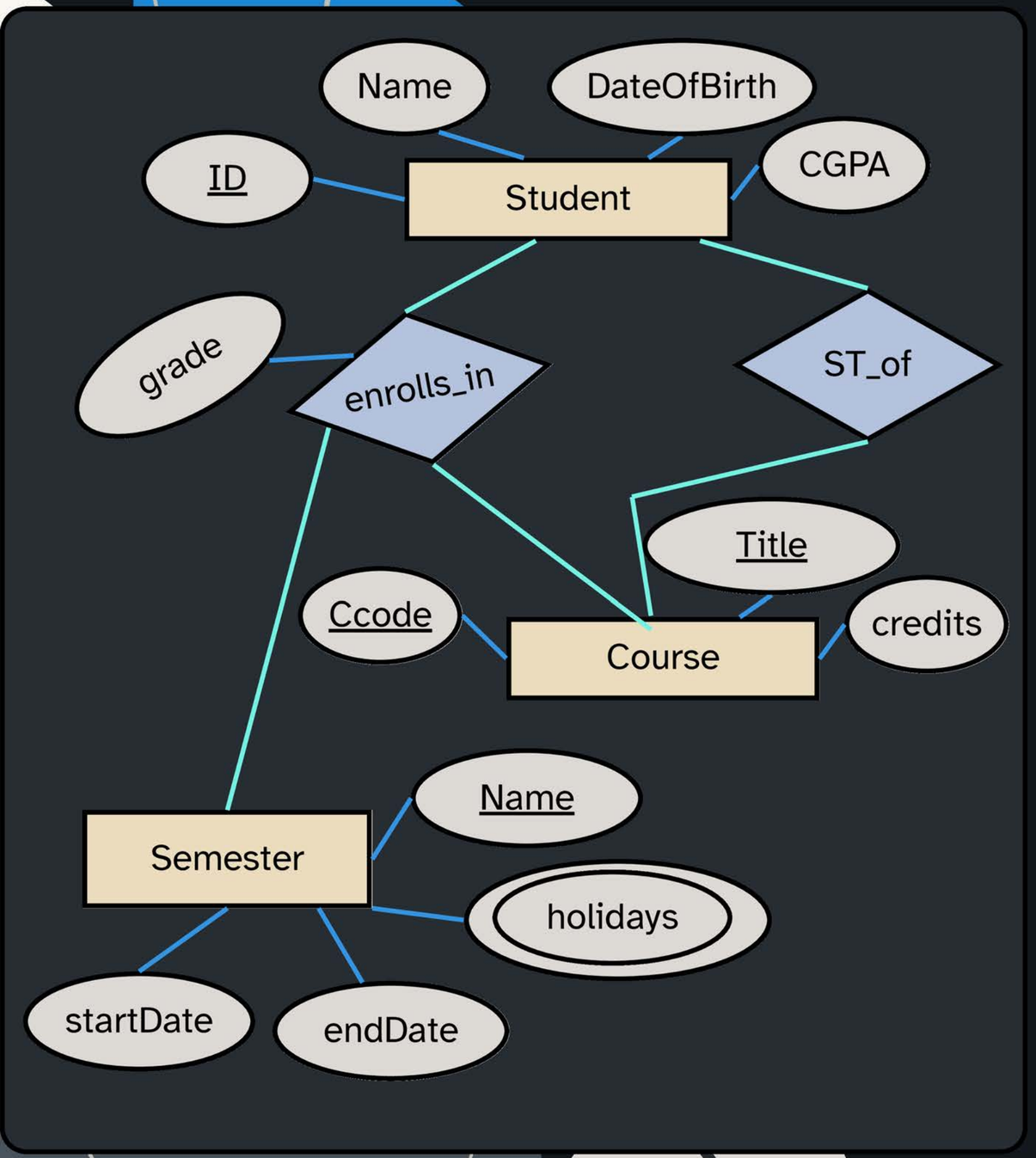



A **relationship** relates two or more distinct entities with a specific meaning. For example, EMPLOYEE John Smith works on the ProductX PROJECT, or STUDENT Ahnaf Atef enrolls in the CSE370 COURSE. Relationships of the same type are grouped together into a **relationship type**.

Relationship types are shown using a **“diamond”** shape connected to the related entity types.

◆ Relationship types may or may not have attributes. In the example, “grade” is a relationship attribute because unless a STUDENT enrolls in a particular COURSE, there will be no grade for that student. Thus, the grade value will only exist if a relationship is established between a specific student and a specific course.

◆ The same entity types may have different relationships between them, each relationship type conveying different meanings. For example, STUDENT and COURSE have two different relationship types: “enrolls_in” and “ST_of”.




# Degree of Relationships

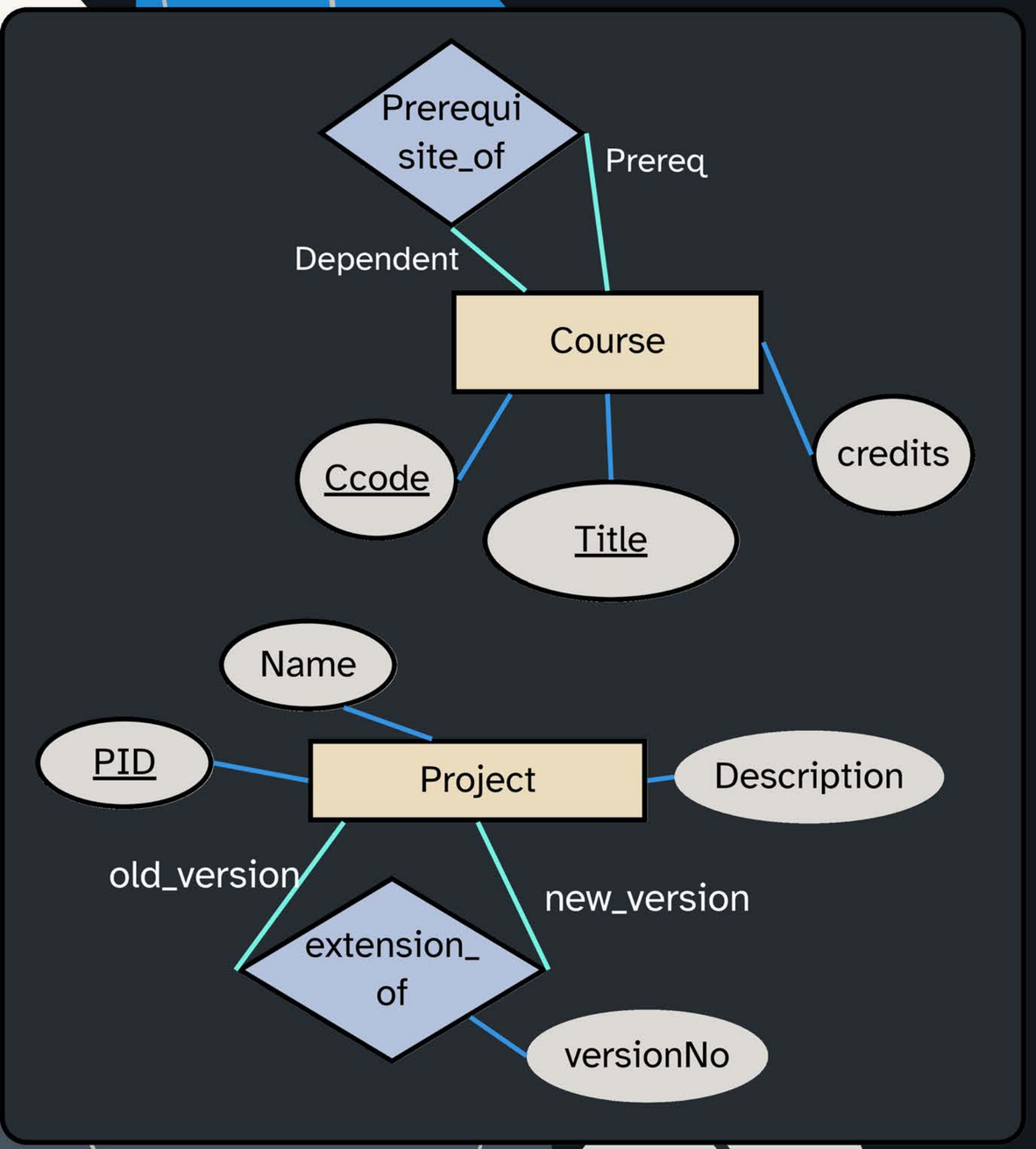
- The **Degree** of a Relationship Type is the number of entity types participating in the relationship.

- "ST_of" relationship has a **degree of 2** because 2 entity types are participating in this relationship. A relationship of degree 2 is called a **Binary Relationship.**

- Here "enrolls_in" is a **Ternary Relationship** of **degree 3** as 3 entity types are participating in this relationship.

- A relationship of **degree n** is called an **n-ary relatiosnhip,** meaning “n” number of entity types are participating in that relationship.

# Recursive Relationship

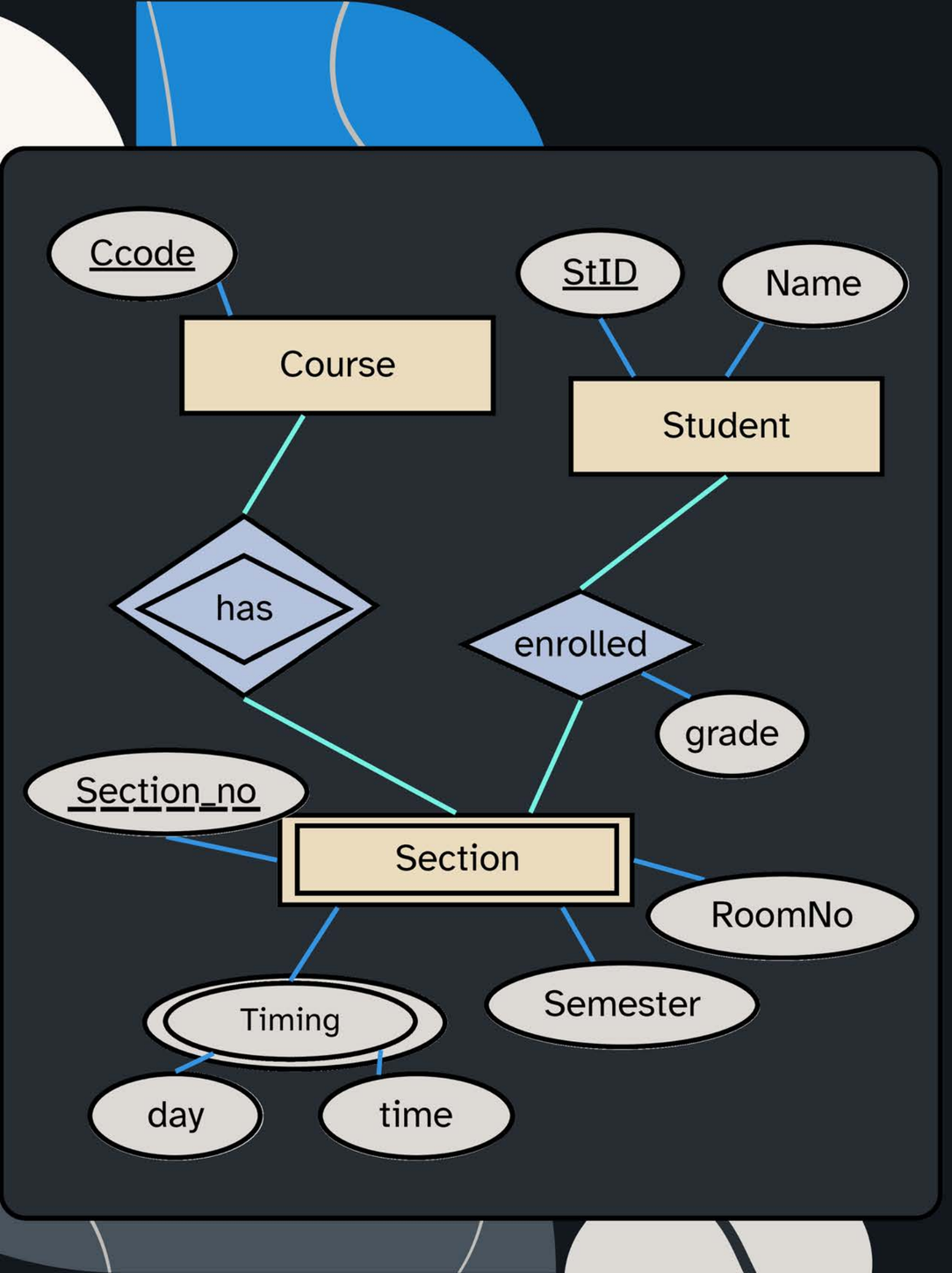



- When two entities of the same entity type participate in a relationship, it is called **Recursive Relationship**. For example, CSE221 is a prerequisite of CSE370 course. Both these courses belong to the same entity type and they have a relationship between them.

- Even though, the participating entities belong to the same type, they will have distinct or different roles, e.g. CSE221 is the prerequisite and CSE370 is the dependent course; or project1 is the old version, while project1b is the new version.

- The distinct roles of the entities in a recursive relationship must be shown as **'labels'** on the relationship edges in the ER diagram.

# Identifying Relationship




- A weak entity **must** participate in an **Identifying Relationship** with an owner or identifying entity type**.** For example, here a COURSE is used to identify weak entity SECTION, therefore course is the owner/identifying entity for SECTION, thus SECTION will have an **identifying relationship** with COURSE. Identifying relationships are shown using a **"double diamond"** shape in the ER diagram.

- A weak entity is identified using the combination of the partial key of the weak entity and the identifying entity's key attribute.

- A weak entity can have normal "non-identifying" relationships with other entity types. for example, SECTION can have a relationship with STUDENTs, here the relationship is not identifying, so it will be shown using "single" diamond shape as usual.

# Relationship Constraints (1)

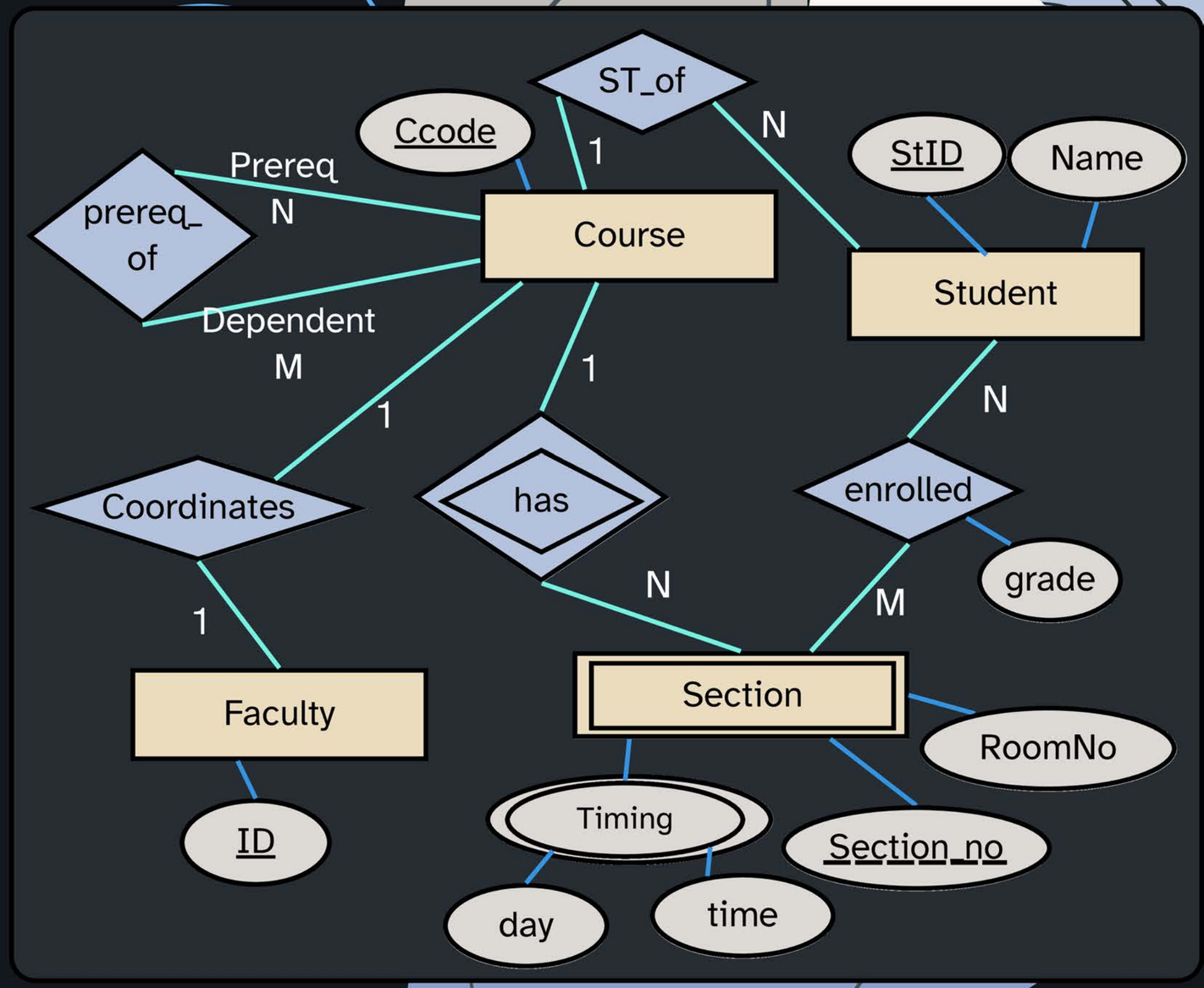
**Cardinality Ratio** (specifies maximum participation):

Shown in the ER by placing appropriate numbers on the relationship edges:

🔷 One-to-one (1:1): A single entity in one entity type is related to a single entity in the other entity type. E.g. 1 FACULTY member can coordinate only 1 COURSE at a time and a COURSE will have only 1 COORDINATOR.

🔷 One-to-many (1:N) or Many-to-one (N:1): An entity from one type can be related to multiple entities from the other entity type, or vice versa. 1 COURSE has many SECTIONs, but a SECTION belongs to only 1 COURSE.

🔷Many-to-many (M:N): Multiple entities from one type are related to multiple entities from the other type. Many STUDENTS are enrolled in a SECTION, and a SECTION has many STUDENTS in it.



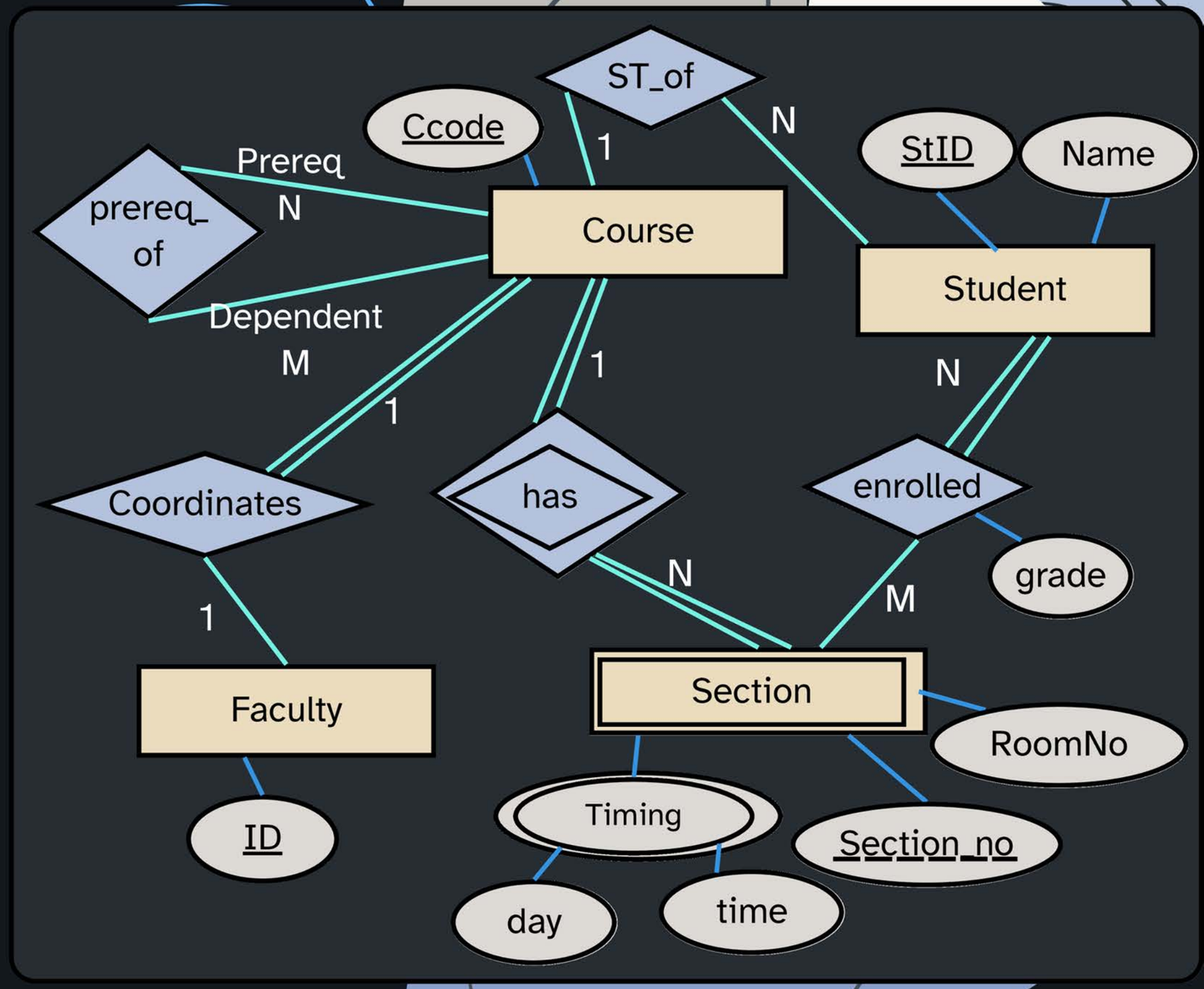
# Relationship Constraints (2)

- **Participation Constraint** (specifies minimum participation) (also called Existence Dependency constraint):

  - Zero (optional participation): Participation is optional, relationship edge is indicated by a “single line” in the ER. For example, it is optional for a STUDENT to become an ST of a COURSE. Also every COURSE will not have an ST

  - One or more (mandatory participation): Participation is mandatory, relationship edge is indicated by a “double line” in the ER. For example, it is mandatory for every COURSE to have a coordinator, however, all FACULTY members may not coordinate COURSEs.

  Note, if Cardinality Ratio/Participation Constraints are not mentioned in the data requirement, you may make an assumption based on logic and real world knowledge



# FROM DATA REQUIREMENTS TO ER DIAGRAM

Given data requirements for a mini-world, identify the entities, attributes and relationships and construct an ER model.

**Collect Data Requirements**

**Analyze Data and Identify ER components**

**Create the ER Model**

You want to create an employee and project management system for the “XYZ” company. Construct an ER model according to the given data requirements:

→ The company is organized into departments. Each department has a unique name,  unique number and an employee who manages the department. We keep track of the start date of the department manager. A department may have several locations.

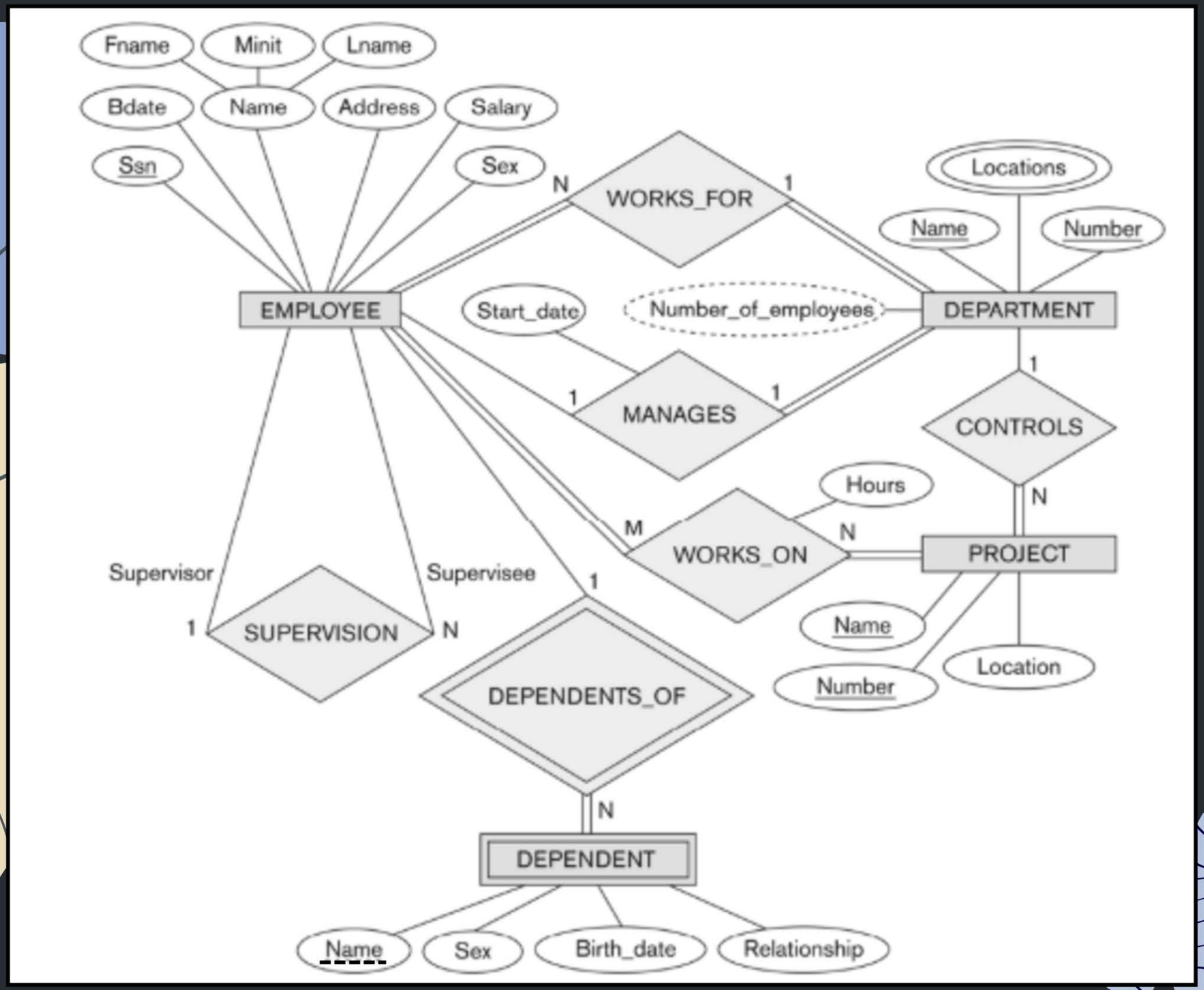
→ Each department controls a number of projects. Each project has a unique name, unique number and is located at a single location.

→ We store each EMPLOYEE’s social security number, name (composed of first name, middle initial and last name), address, salary, sex, and birthdate. Each employee works for one department but may work on several projects.

→ We keep track of the number of hours per week that an employee currently works on each project. We also keep track of the direct supervisor of each employee.

→ Each employee may have a number of dependents. For each dependent, we only keep track of their name, sex, birthdate and relation with the employee.

→ We also need to know the number of employees working in a department. However, this value can be calculated from counting the number from the relationship instances, therefore, this value does not need to be stored.


Fname
Minit
Lname
Bdate
Name
Address
Salary
Ssn
Sex
N
WORKS_FOR
1
Locations
Name
Number
EMPLOYEE
Start_date
Number_of_employees
DEPARTMENT
1
1
MANAGES
1
CONTROLS
Hours
N
M
N
WORKS_ON
PROJECT
Supervisor
Supervisee
1
1
SUPERVISION
N
Name
Location
Number
DEPENDENTS_OF
N
DEPENDENT
Name
Sex
Birth_date
Relationship

# WHAT NEXT ?

## Lecture 3: Extending the ER Model Using EER

LOADING.....