Entity Relationship Diagram (ERD)

Dr. Shah Murtaza Rashid Al Masud

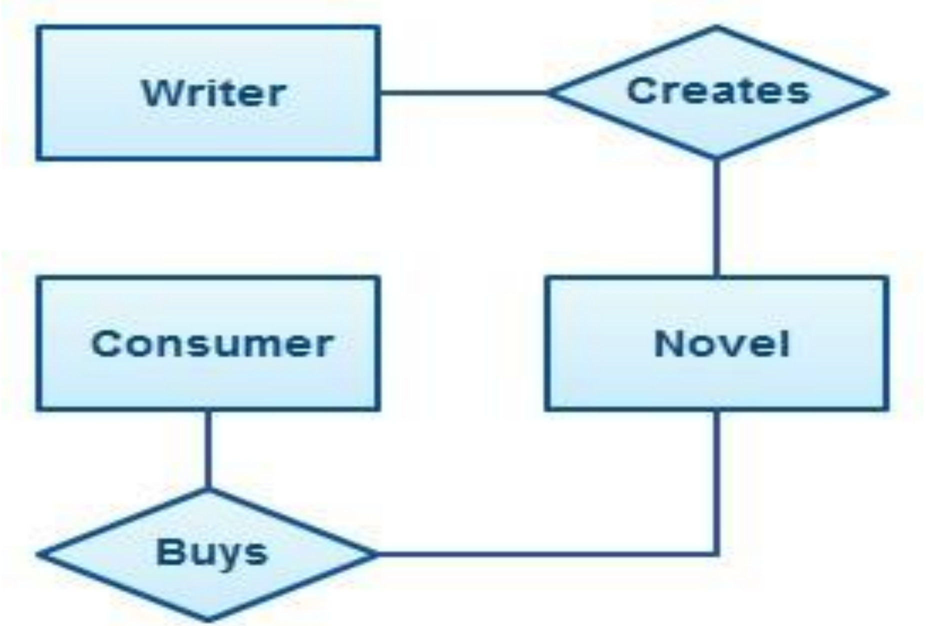
CSE Department

UAP

ENTITY RELATIONSHIP DIAGRAM

An Entity Relationship Diagram (ERD) is a visual representation of different data that describe how these data are related to each other.

For example, the elements writer, novel, and consumer may be described using ER diagrams this way:



The elements inside **rectangles** are called **<u>entities</u>** while the items inside **diamonds** denote the **<u>relationships</u>** between entities.

Any system can be modeled as:

- a collection of entities,
- relationship among entities.

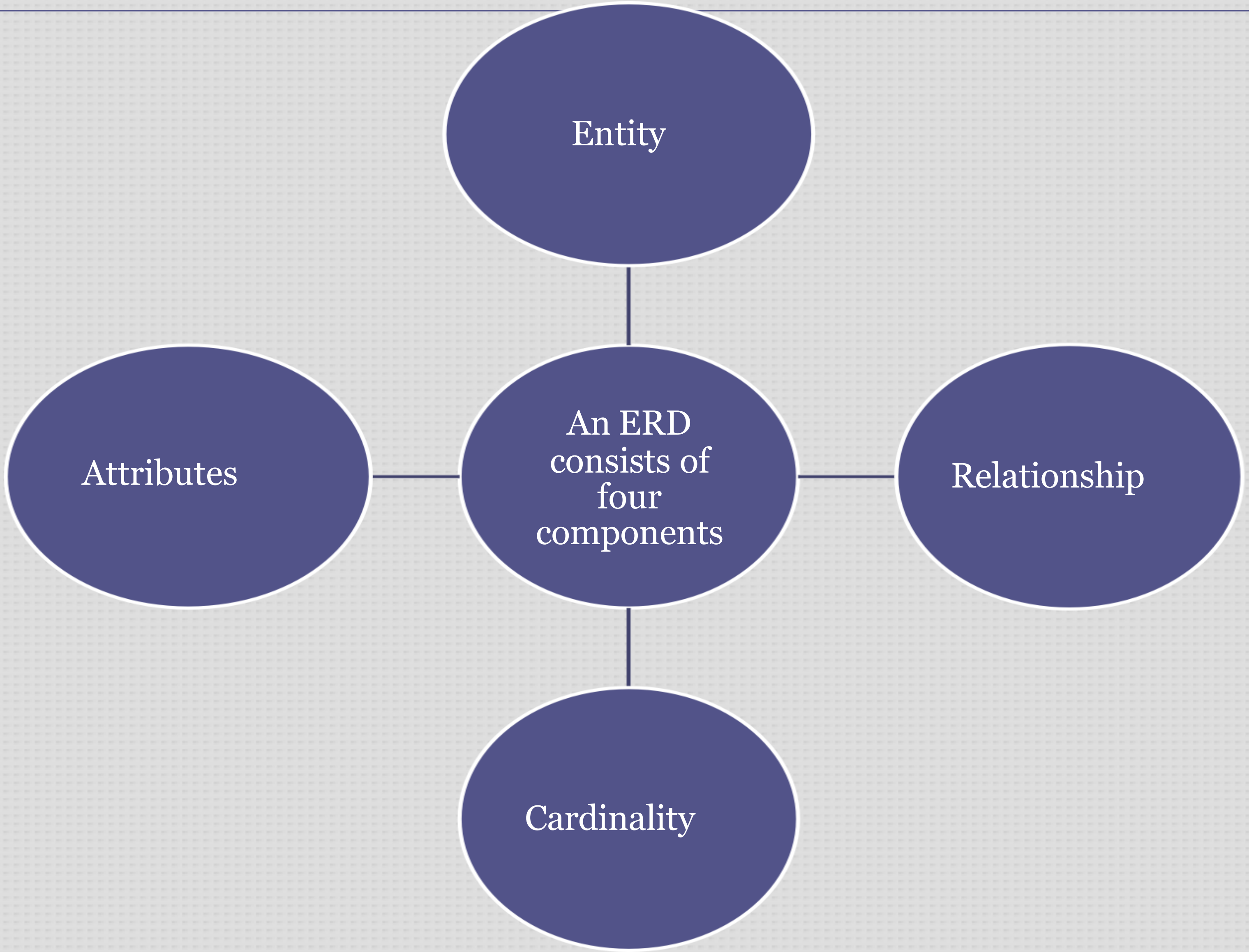
Entity Relationship Diagram

ERD is a graphical tool for modeling data.

ER model allows us to sketch database designs.

ERD is a model that identifies the entities that exist in a system and the relationships between those entities

Components of an ERD



Entity

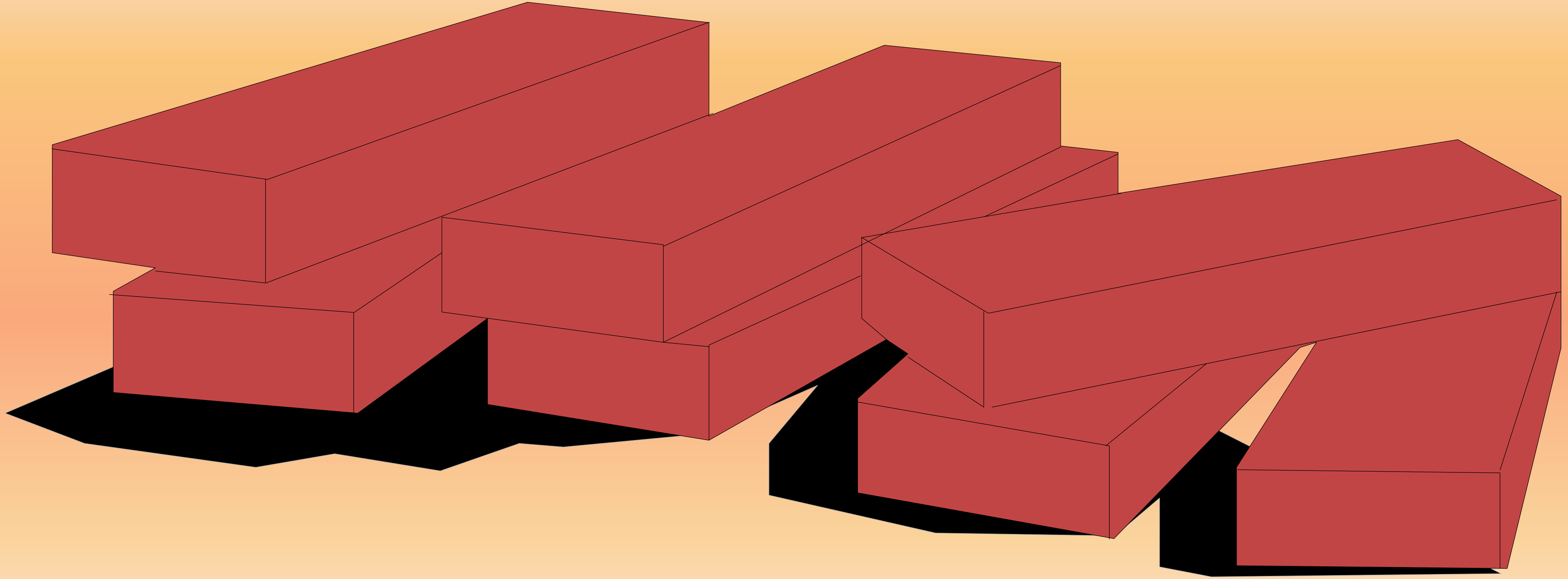
Attributes

An ERD consists of four components

Relationship

Cardinality

BASIC CONCEPTS

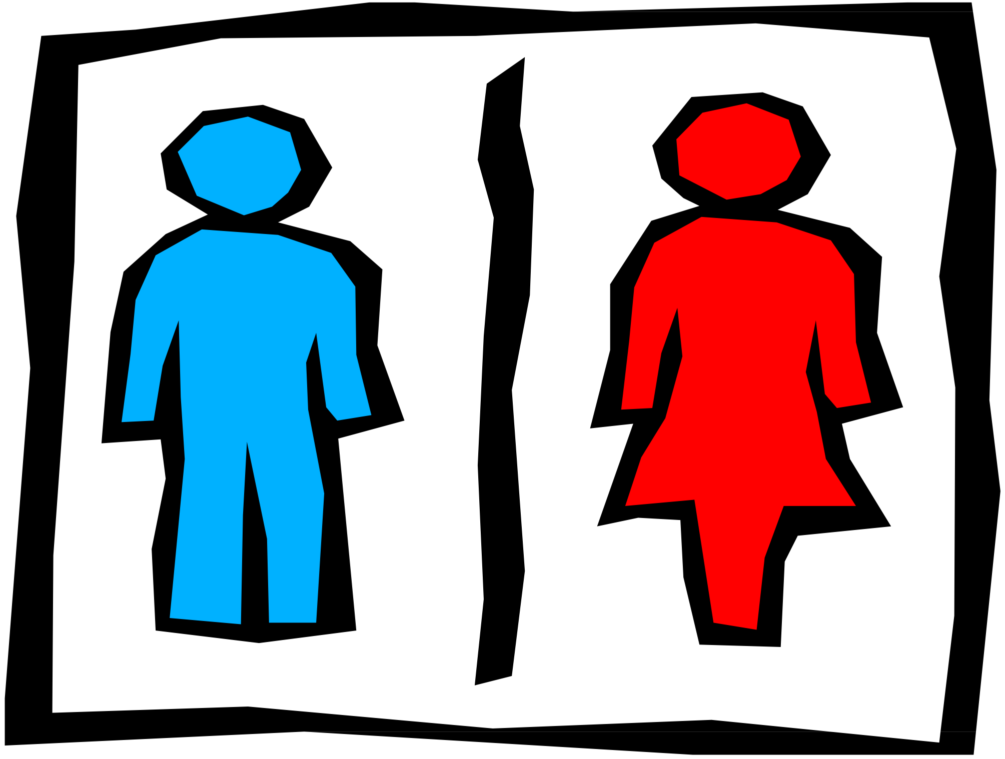


WHAT IS ENTITY?

- Person, place, object, event or concept about which data is to be maintained
- Examples of entities:
  - ***Person: EMPLOYEE, STUDENT, PATIENT***
  - ***Place: STORE, WAREHOUSE***
  - ***Object: MACHINE, PRODUCT, CAR***
  - ***Event: SALE,REGISTRATION, RENEWAL***
  - ***Concept: ACCOUNT, COURSE***

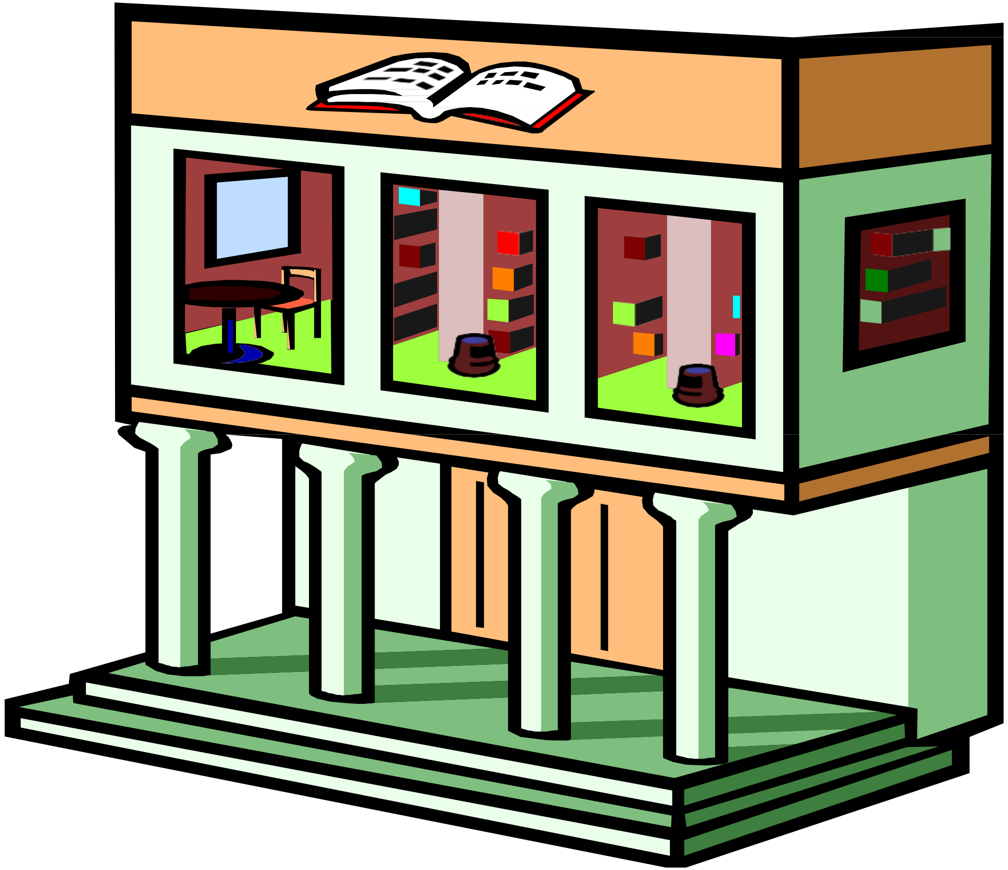
WHAT IS ENTITY?

- **Person**



WHAT IS ENTITY?

- **Place**



WHAT IS ENTITY?

- **Object**

WHAT IS ENTITY?

- **Event**

Entity

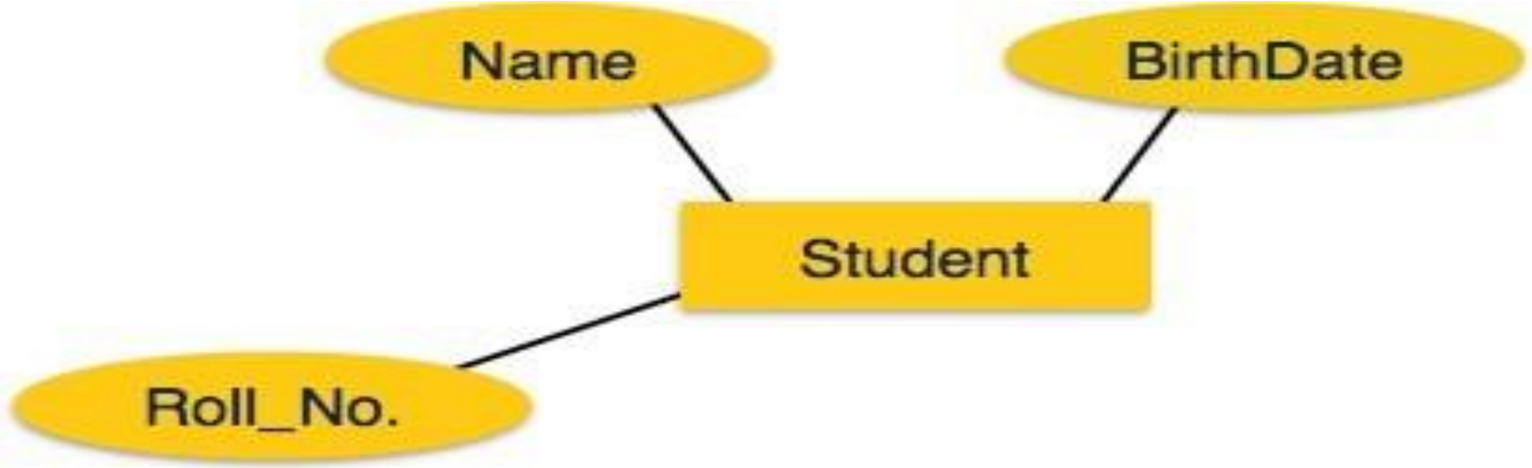
- ***An entity can be a real-world object***, either animate or inanimate, that can be ***easily identifiable***.
  - For example, in a school database, **students, teachers, classes,** and **courses offered** can be considered as **entities**.
  - All these entities have some ***attributes or properties*** that give them their identity.
- An **entity set** is a collection of similar types of entities.
  - An entity set may contain ***entities with attribute sharing similar values.***
  - For example,
  - a Students set may contain all the students of a school;
  - likewise a Teachers set may contain all the teachers of a school from all faculties. Entity sets need not be disjoint.

ATTRIBUTES

- Entities are represented by means of their ***properties***, called **attributes**.
- All attributes have values.
- For example, a **student entity** may have **name, class,** and **age** as attributes.
- ***There exists a domain or range of values that can be assigned to attributes.***
- For example, a **student's name** cannot be a numeric value. It has to be ***alphabetic***. A **student's age cannot be negative**, etc.

ATTRIBUTES

- Attributes are the properties of entities. Attributes are represented by means of
- ellipses.
- Every ellipse represents one attribute and is c

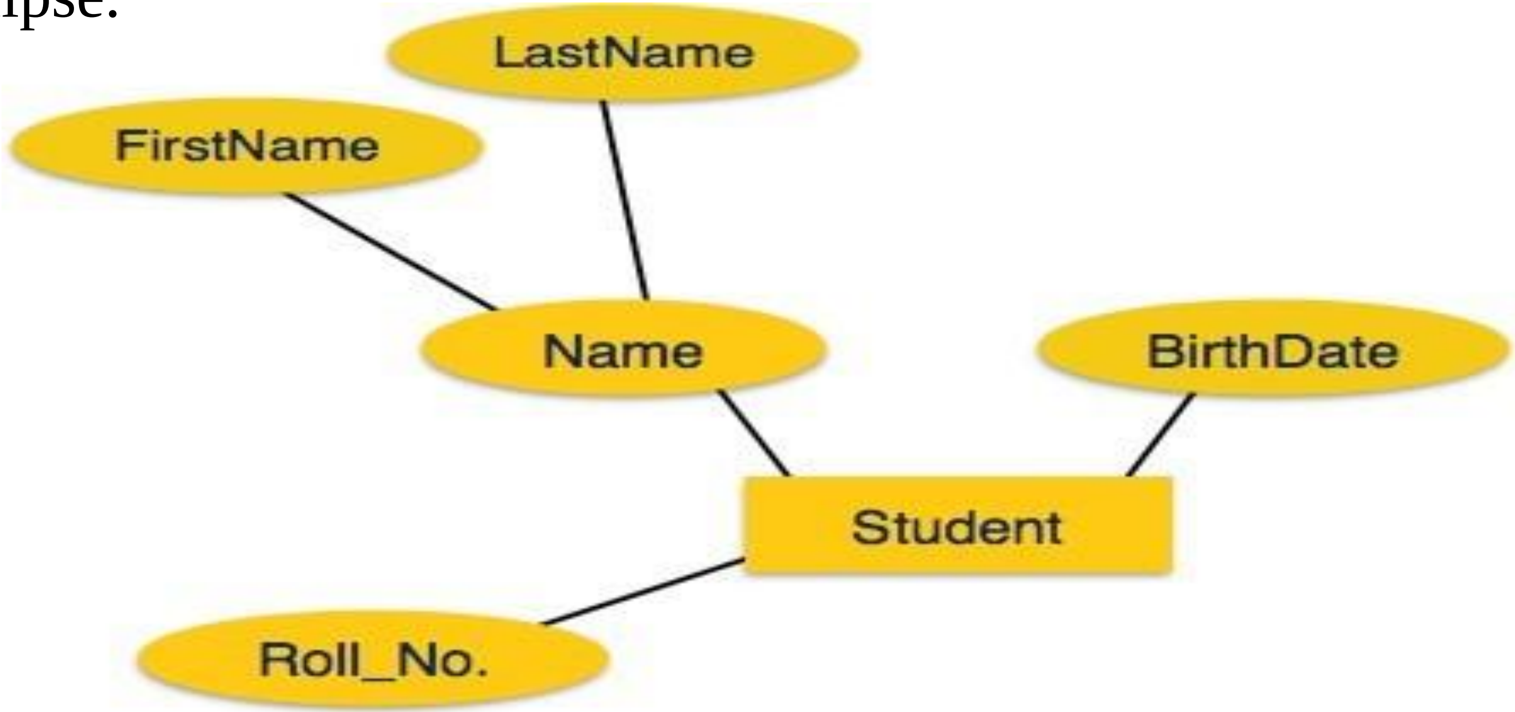


Types of Attributes

- **Simple attribute** − Simple attributes are atomic values, which cannot be divided further. For example, **a student's phone number is an atomic value of 10 digits**.

- **Composite attribute** − Composite attributes are made of more than one simple attribute. For example, **a student's complete name may have first_name and last_name.**

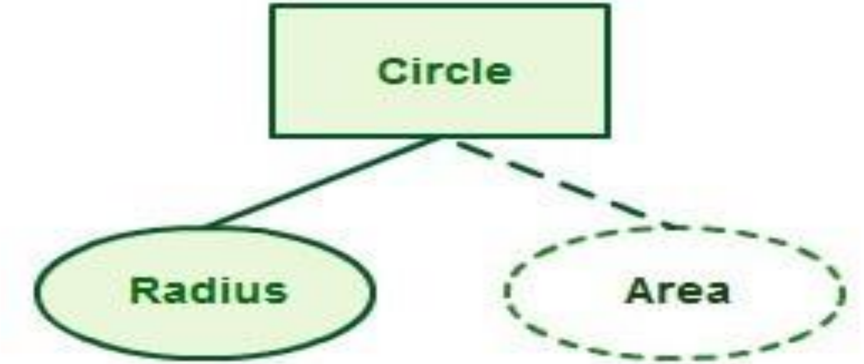
If the attributes are **composite**, they are **further divided in a tree like structure**. Every node is then connected to its attribute. That is, composite attributes are represented by ellipses that are connected with an ellipse.



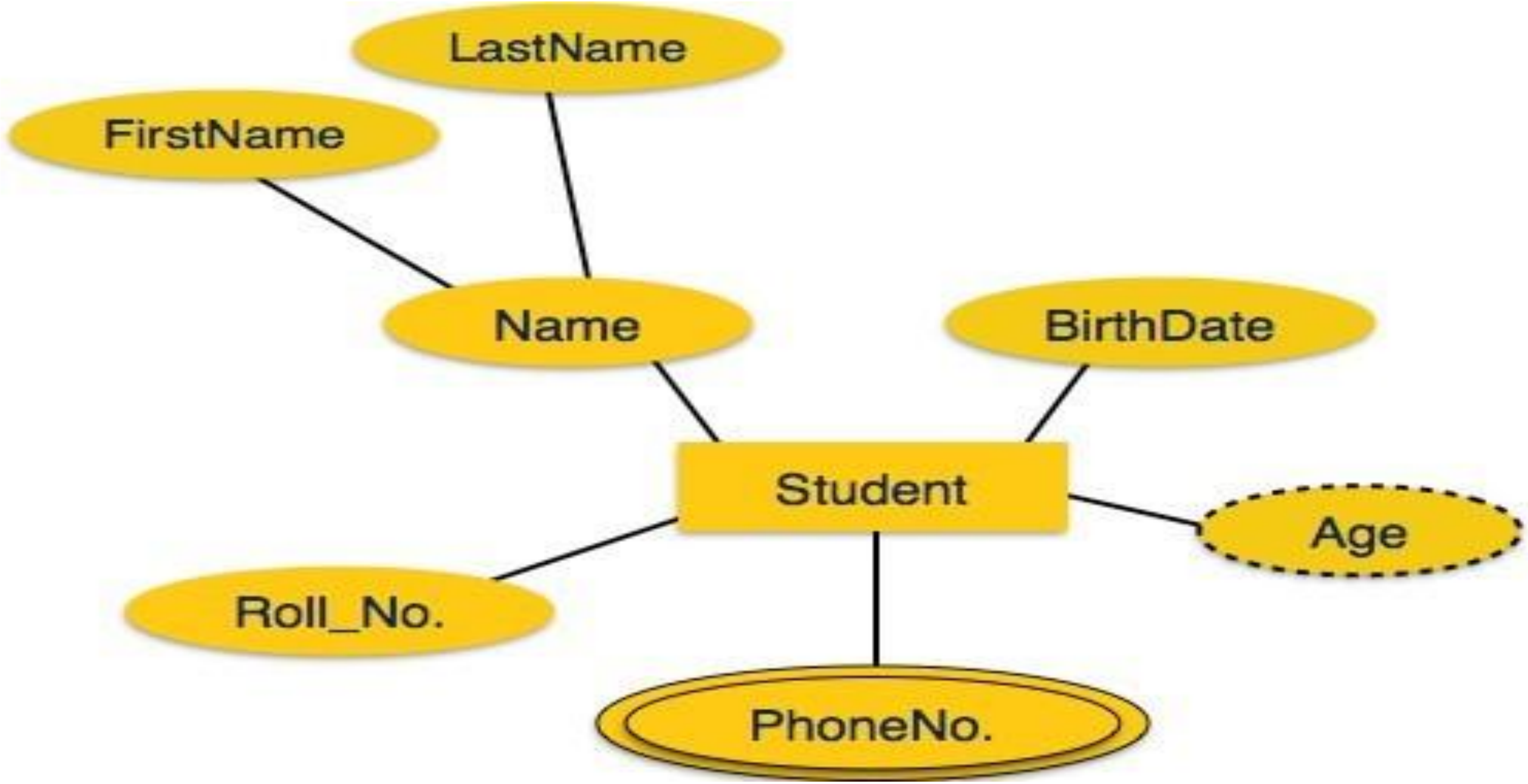
TYPES OF ATTRIBUTES (CONTD.,)

- **Derived attribute** − Derived attributes are the attributes that do not exist in the physical database, but their **values are derived from other attributes** present in the database.

- For example, average_salary in a department should not be saved directly in the database, instead it can be derived.
- For another example, age can be derived from data_of_birth.
- For example for a circle the area can be derived from the radius.



- **Derived** attributes are depicted by dashed ellipse.

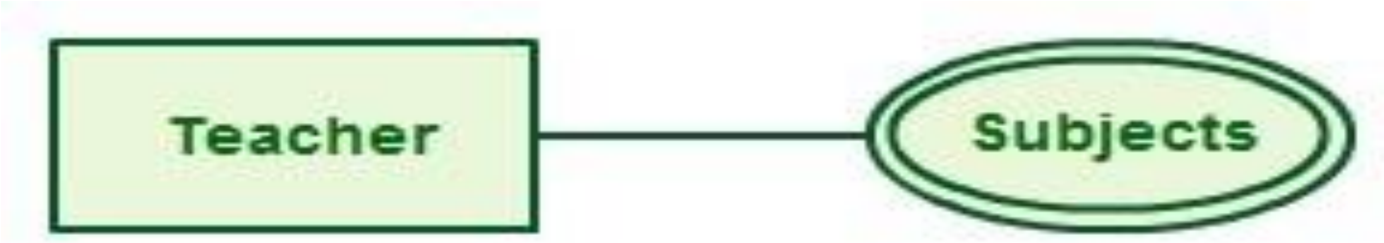


TYPES OF ATTRIBUTES (CONTD.,)

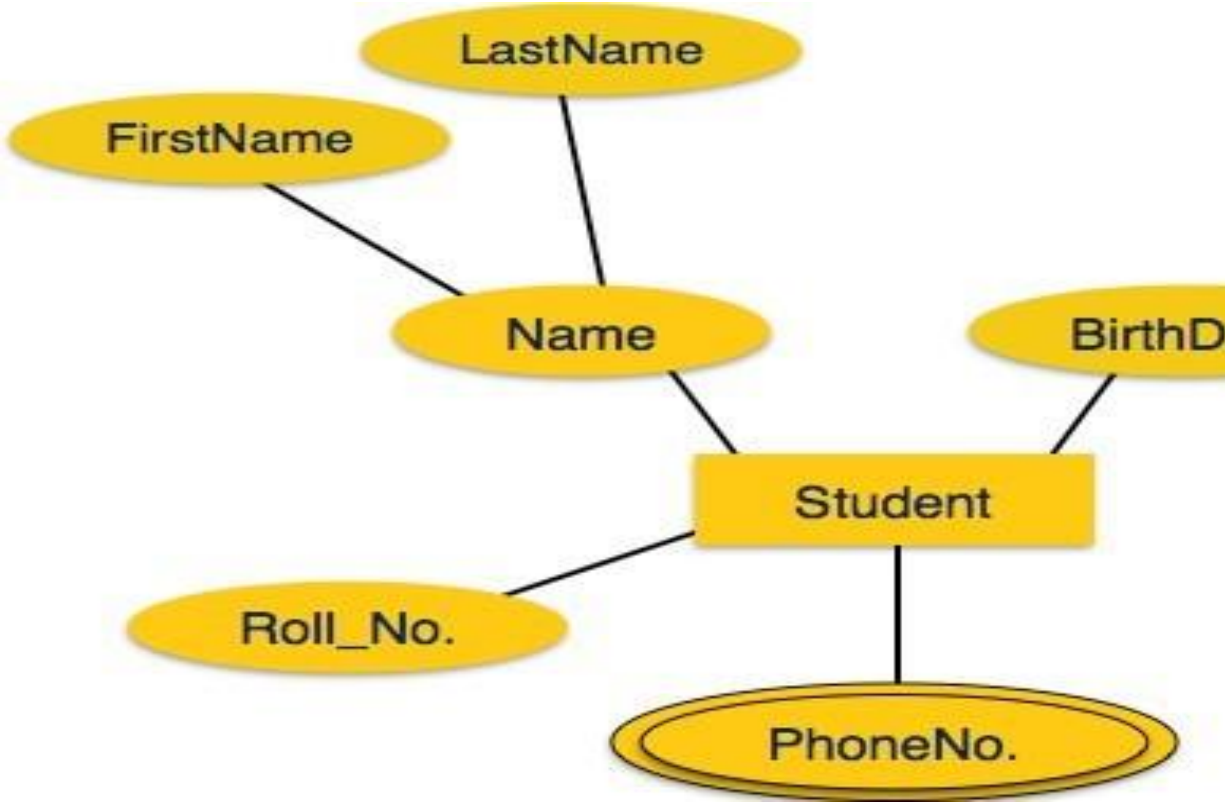
- **Single-value attribute** – Single-value attributes **contain single value**. For example Social_Security_Number.

- **Multi-value attribute** – Multi-value attributes may **contain more than one values**. For example, a person can have more than one phone number, email_address, etc.

- For example a teacher entity can have multiple subject values.

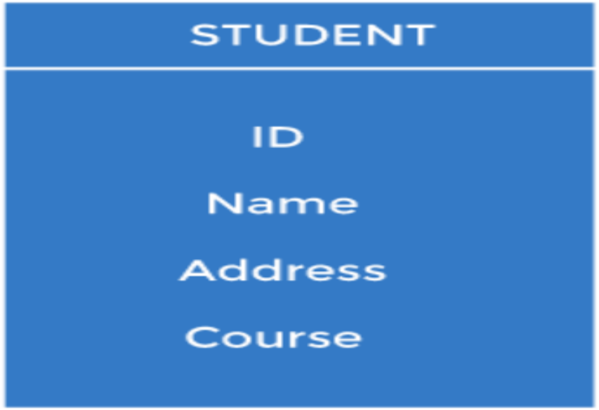


- **Multivalued** attributes are depicted by double ellipse.



KEYS

- ✓ Key plays an important role in the database.
- ✓ Key is used to uniquely identify any record or row of data from the table. It is also used to establish and identify relationships between tables.
- ✓ **For example,** ID is used as a key in the Student table because it is unique for each student. In the PERSON table, passport_number, license_number, SSN are keys since they are unique for each person.



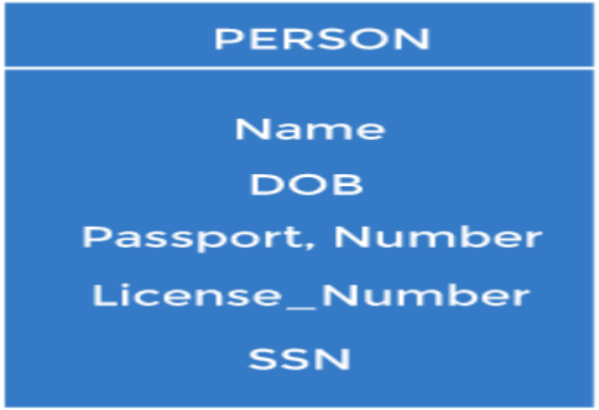
STUDENT

ID

Name

Address

Course



PERSON

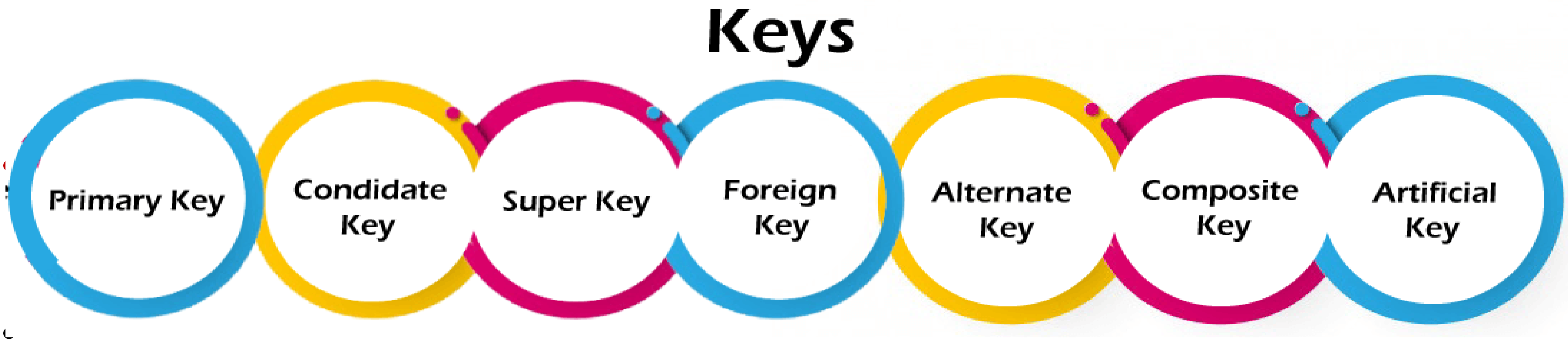
Name

DOB

Passport, Number

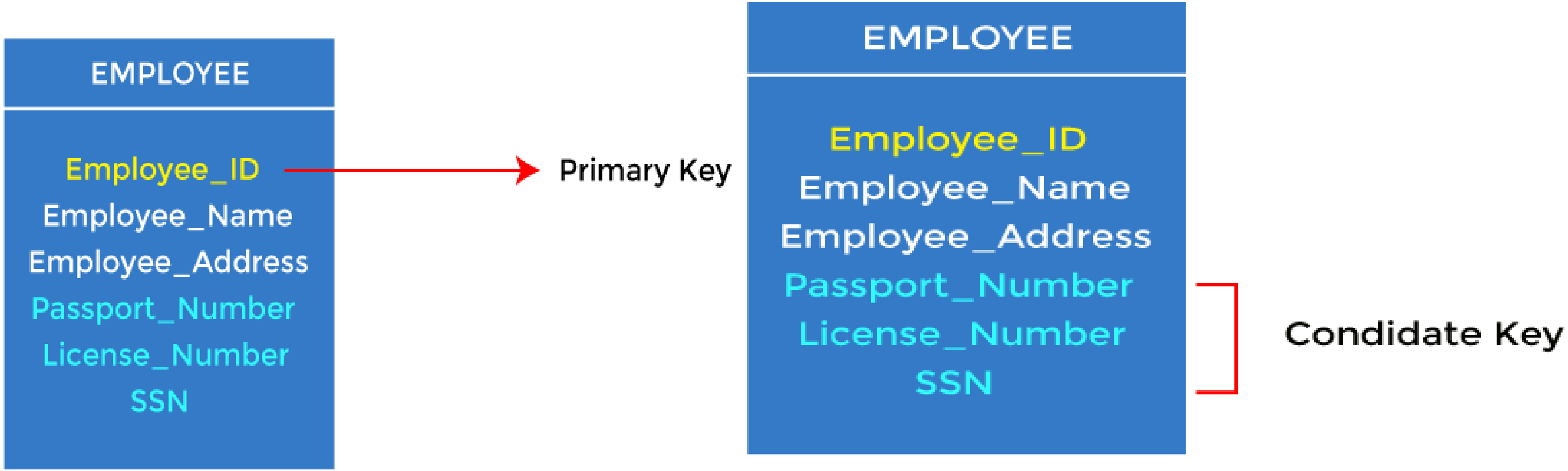
License_Number

SSN



PRIMARY KEY

- It is the first key used to identify one and only one instance of an entity uniquely. An entity can contain multiple keys, as we saw in the PERSON table. The key which is most suitable from those lists becomes a primary key.
- In the EMPLOYEE table, **ID can be the primary key** since it is unique for each employee. In the EMPLOYEE table, we can even select License_Number and Passport_Number as primary keys since they are also unique.
- For each entity, the primary key selection is based on requirements and developers.



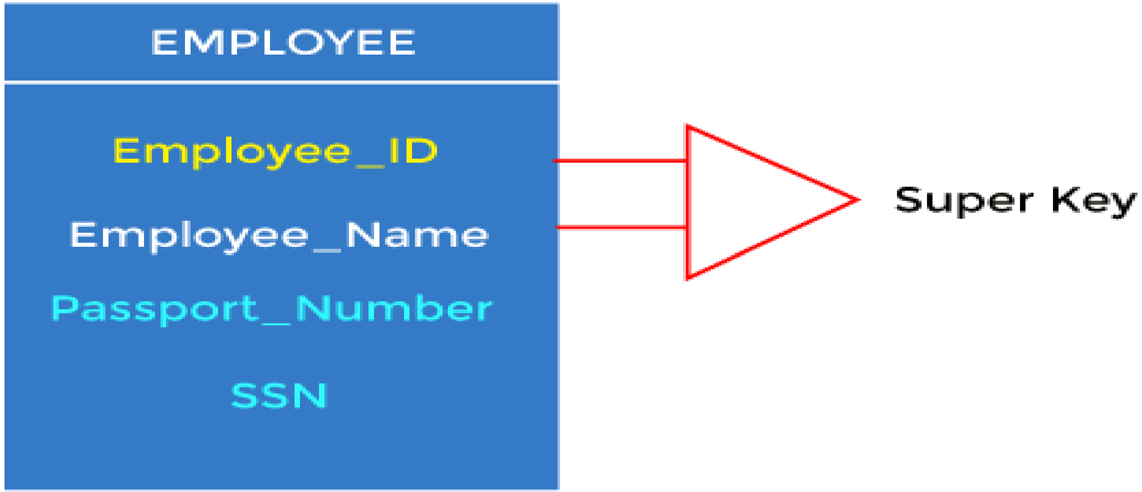
CANDIDATE KEY

- ❖ A candidate key is an attribute or set of attributes that can uniquely identify a tuple.
- ❖ Except for the primary key, the remaining attributes are considered a candidate key. The candidate keys are as strong as the primary key.
- ❖ **For example:** In the EMPLOYEE table, id is best suited for the primary key. The rest of the attributes, like SSN, Passport_Number, License_Number, etc., are considered a candidate key.

SUPER KEY

Super key is an attribute set that can uniquely identify a tuple. A super key is a superset of a candidate key. since **Candidate Keys are selected from super key**

For example: In the above EMPLOYEE table, for(EMPLOEE_ID, EMPLOYEE_NAME), the name of two employees can be the same, but their EMPLYEE_ID can't be the same. Hence, this combination can also be a key.

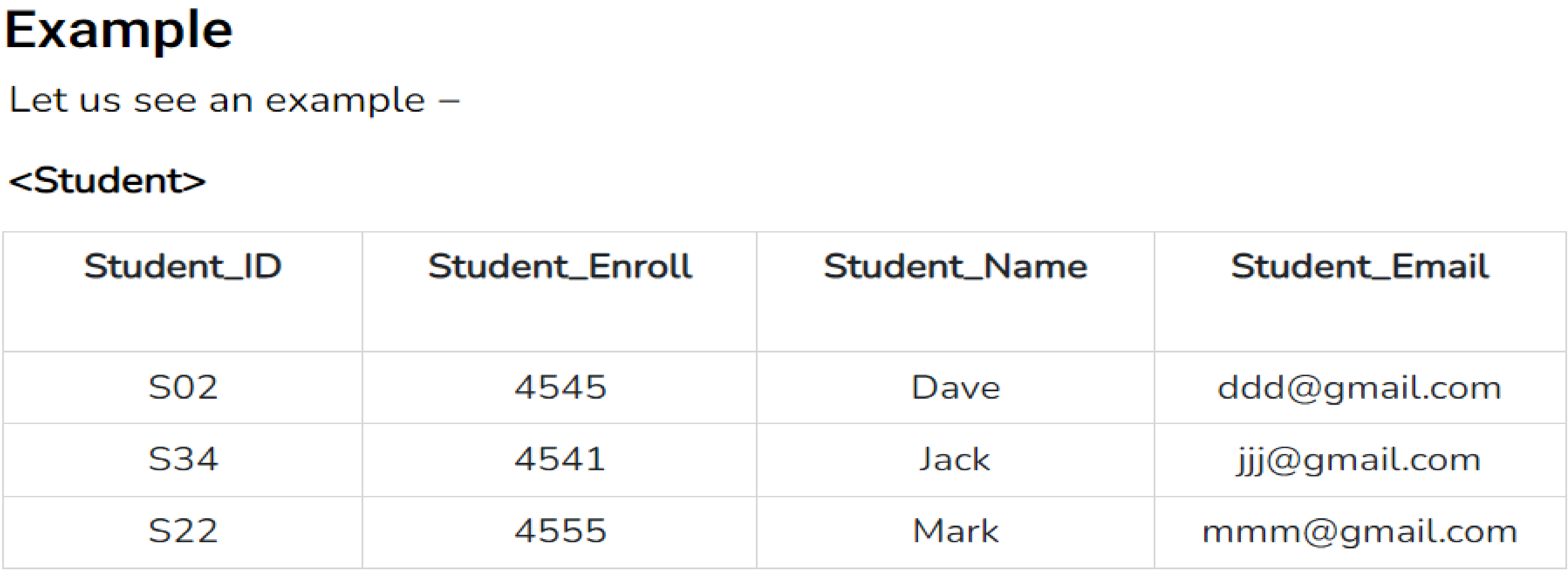


SUPER KEY

Note:

In set theory, set A is considered as the superset of B, if all the elements of set B are the elements of set A. For example, **if set A = {1, 2, 3, 4} and set B = {1, 3, 4}, we can say that set A is the superset of B**.

Super Key



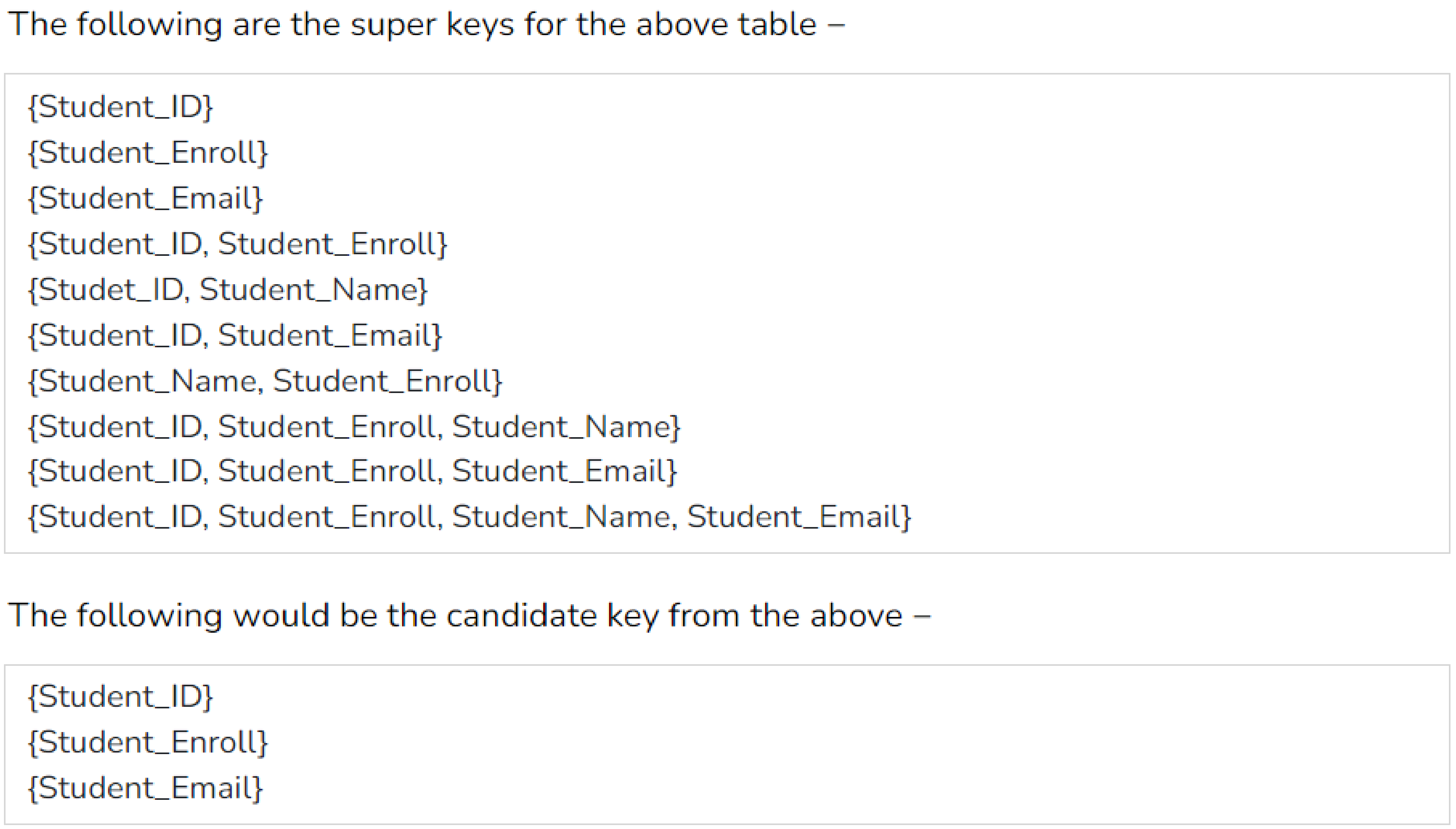
Example

Let us see an example –

<Student>

Student_ID	Student_Enroll	Student_Name	Student_Email
S02	4545	Dave	ddd@gmail.com
S34	4541	Jack	jjj@gmail.com
S22	4555	Mark	mmm@gmail.com

SUPER KEY



The following are the super keys for the above table –

{Student_ID}

{Student_Enroll}

{Student_Email}

{Student_ID, Student_Enroll}

{Studet_ID, Student_Name}

{Student_ID, Student_Email}

{Student_Name, Student_Enroll}

{Student_ID, Student_Enroll, Student_Name}

{Student_ID, Student_Enroll, Student_Email}

{Student_ID, Student_Enroll, Student_Name, Student_Email}

The following would be the candidate key from the above –

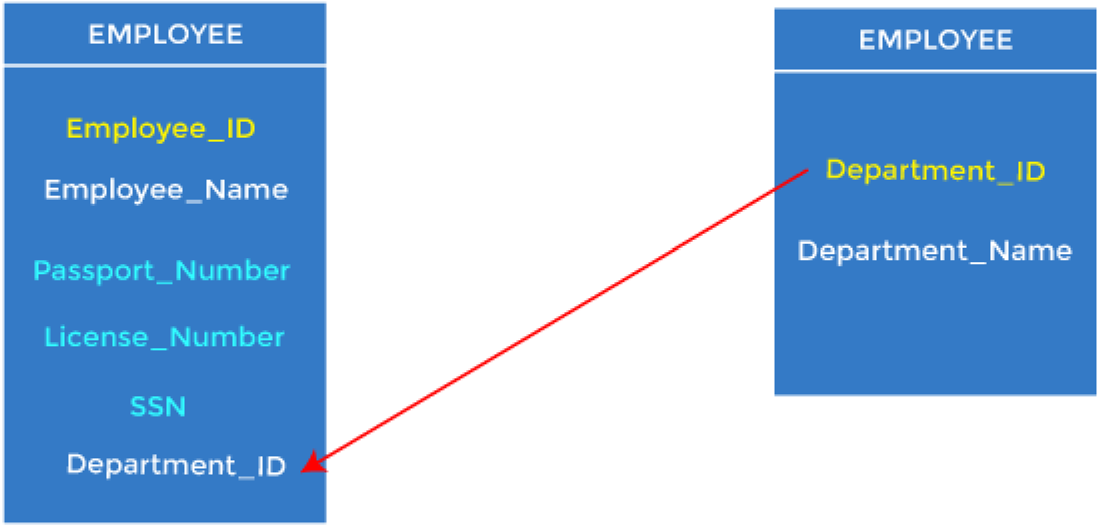
{Student_ID}

{Student_Enroll}

{Student_Email}

FOREIGN KEY

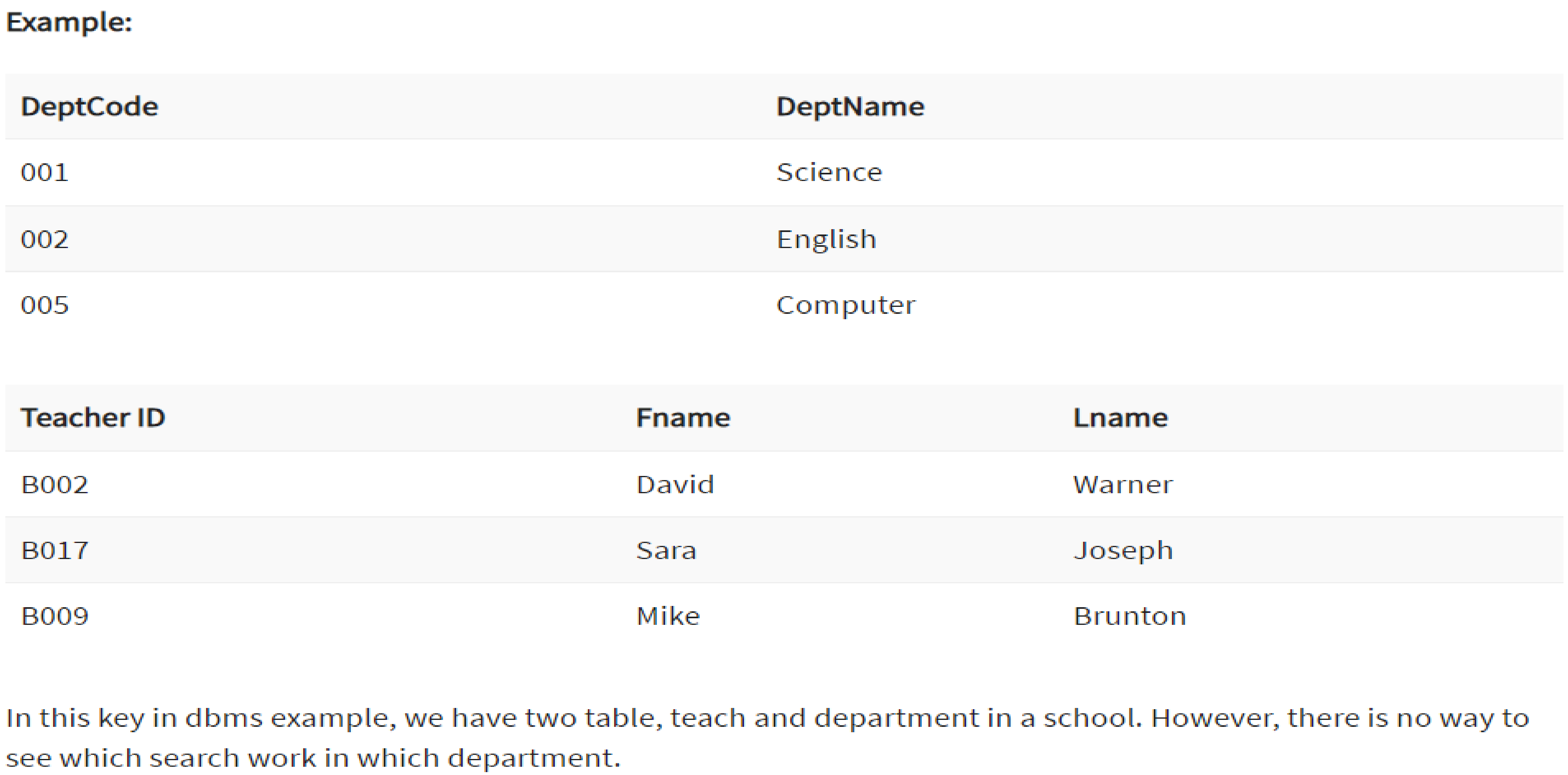
- Foreign keys are the column of the table used to point to the primary key of another table.
- Every employee works in a specific department in a company, and employee and department are two different entities. So we can't store the department's information in the employee table. That's why we link these two tables through the primary key of one table.
- We add the primary key of the DEPARTMENT table, Department_Id, as a new attribute in the EMPLOYEE table.
- In the EMPLOYEE table, Department_Id is the foreign key, and both the tables are related.



FOREIGN KEY

FOREIGN KEY is a column that creates a relationship between two tables.

The purpose of Foreign keys is to maintain data integrity and allow navigation between two different instances of an entity. It acts as a cross-reference between two tables as it references the primary key of another table.



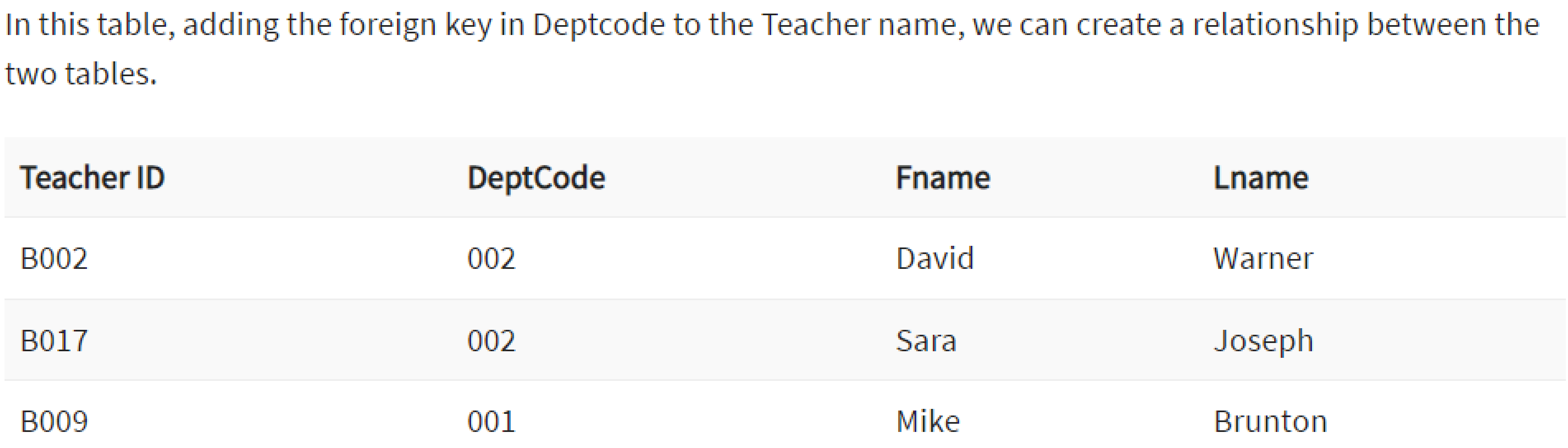
Example:

DeptCode	DeptName
001	Science
002	English
005	Computer

Teacher ID	Fname	Lname
B002	David	Warner
B017	Sara	Joseph
B009	Mike	Brunton

In this key in dbms example, we have two table, teach and department in a school. However, there is no way to see which search work in which department.

FOREIGN KEY



In this table, adding the foreign key in Deptcode to the Teacher name, we can create a relationship between the two tables.

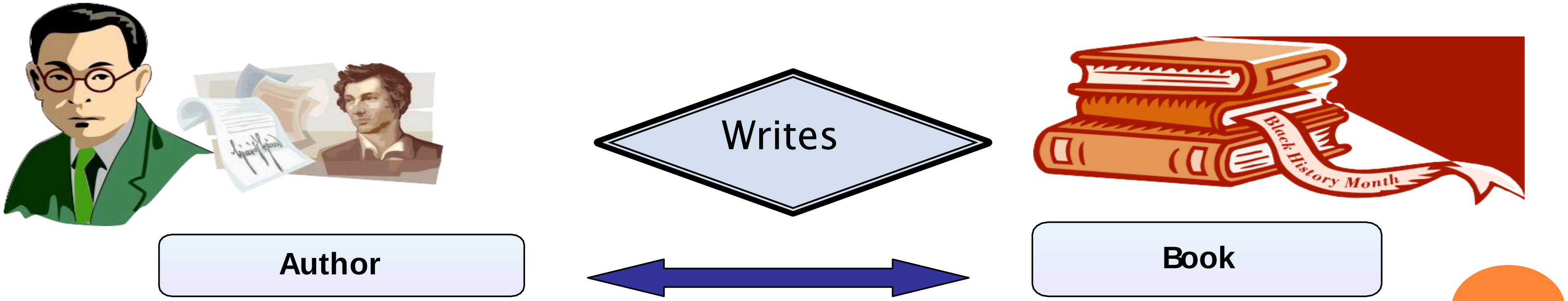
Teacher ID	DeptCode	Fname	Lname
B002	002	David	Warner
B017	002	Sara	Joseph
B009	001	Mike	Brunton

Relationships

→ ASSOCIATIONS BETWEEN INSTANCES OF ONE OR MORE ENTITY TYPES THAT IS OF INTEREST

→Meaningful association among several entities.

→ Given a name that describes its function.



Writes

Author

Book

Relationships

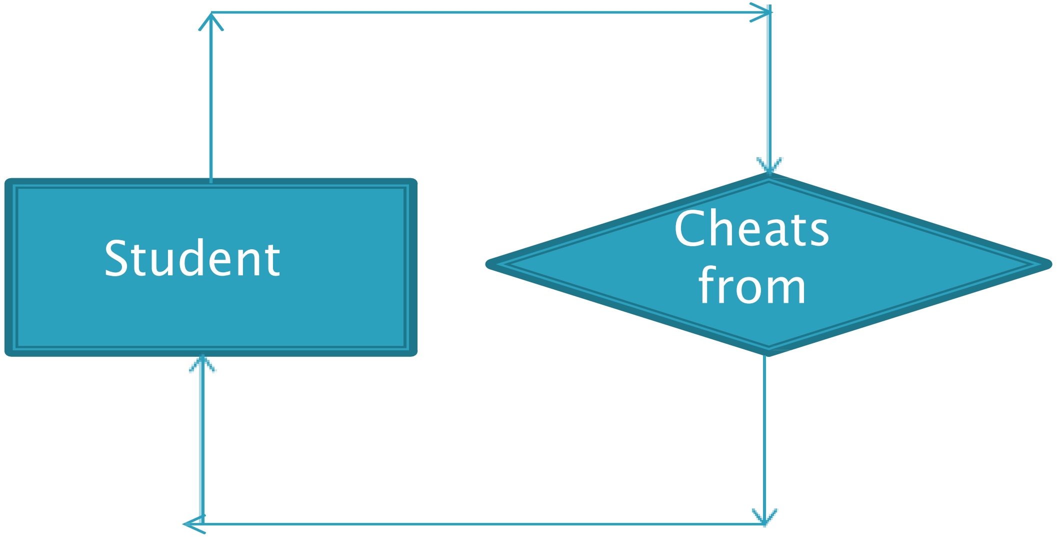
Relationships are represented by **diamond-shaped box**.

Name of the relationship is written inside the diamond-box.

All the entities (rectangles) participating in a relationship, are connected to it by a line.

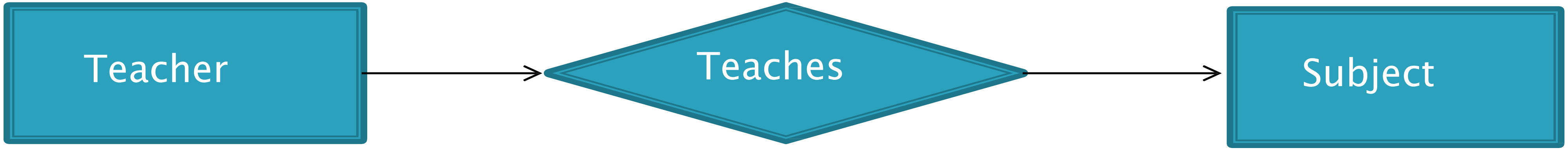
Degree Of Relationship

- Unary Relationship: between two instances of one entity type.



an **instance** is one occurrence of a class or object. For example, a program may have a class/object named Animal, but there could be many instances of Animal, such as lion, cat, and dog.

BINARY RELATIONSHIP : BETWEEN THE INSTANCES OF TWO ENTITY TYPES.

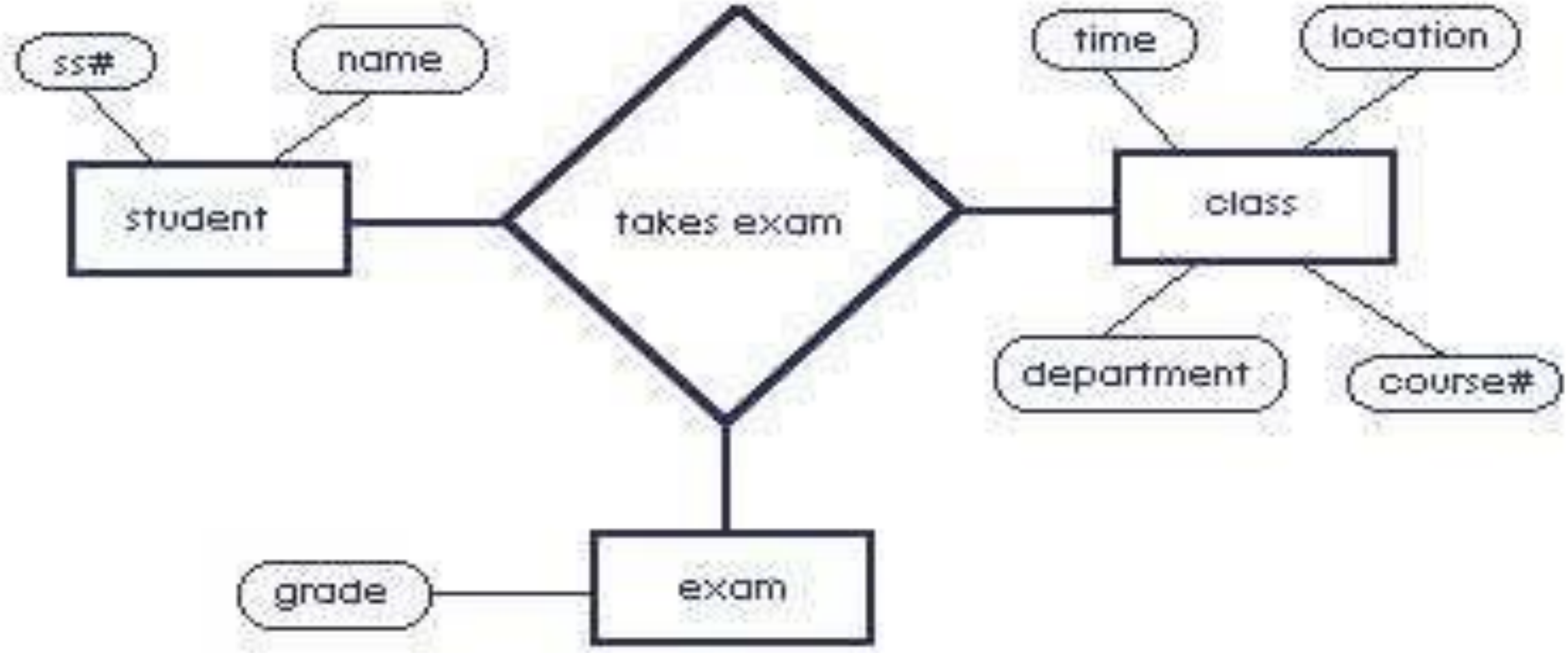

Teacher

Teaches

Subject

A relationship where two entities are participating is called a **binary relationship**.

- Ternary Relationship : among the instances of three entity types



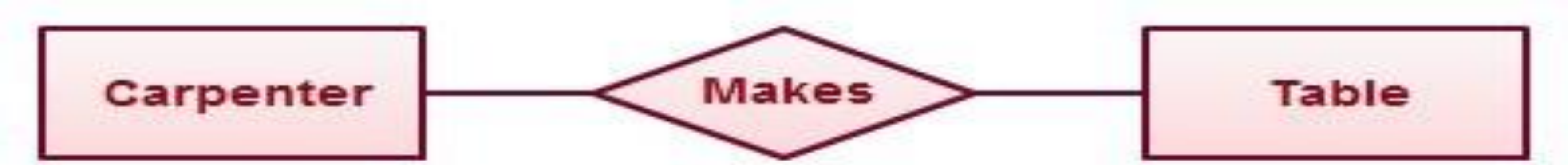
DEGREE OF RELATIONSHIP

- The number of participating entities in a relationship defines the degree of the relationship.

  - Unary = Degree 1
  - Binary = Degree 2
  - Ternary = Degree 3
  - N-ary = Degree N

RELATIONSHIP EXAMPLE

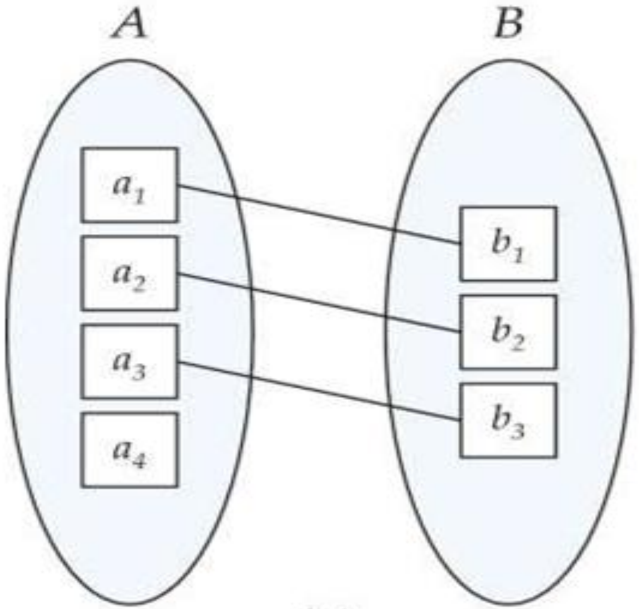
- The association among entities is called a relationship.
- For example, an employee **works_at** a department, a student **enrolls** in a course.
- Here, Works_at and Enrolls are called relationships.
- For example, the entity “**carpenter**” may be related to the **entity** “**table**” by the **relationship “builds” or “makes**”. Relationships are represented by diamond shapes and are labeled using verbs.



TYPES OF RELATIONSHIP

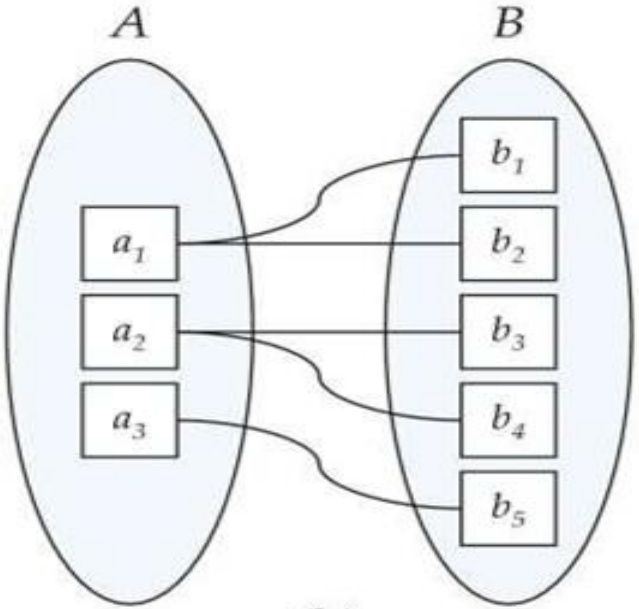
- Types of Relationships
  - Three types of relationships can exist between entities
  - One-to-one relationship (1:1): One instance in an entity (parent) refers to one and only one instance in the related entity (child).
  - One-to-many relationship (1:M): One instance in an entity (parent) refers to one or more instances in the related entity (child)

TYPES OF RELATIONSHIP



(a)

One to one

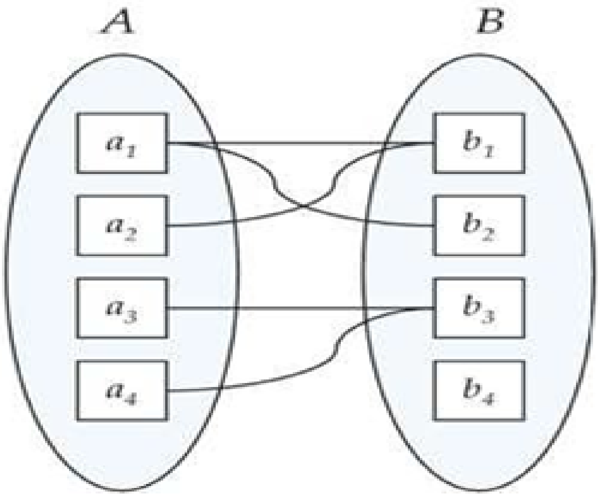


(b)

One to many

TYPES OF RELATIONSHIP

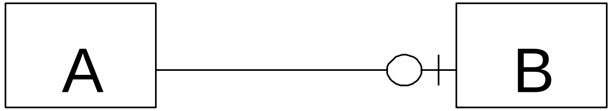
- Types of Relationships
  - Many-to-many relationship (M:N): exists when one instance of the first entity (parent) can relate to many instances of the second entity (child), and one instance of the second entity can relate to many instances of the first entity.



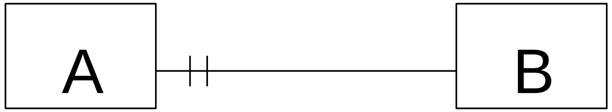
Many to many

Cardinality

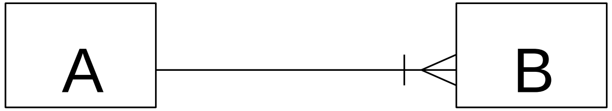
Cardinality is the number of instance of an entity from a relation that can be associated with the relation.



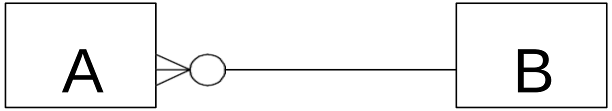
Each instance of A is related to a minimum of zero and a maximum of one instance of B



Each instance of B is related to a minimum of one and a maximum of one instance of A



Each instance of A is related to a minimum of one and a maximum of many instances of B

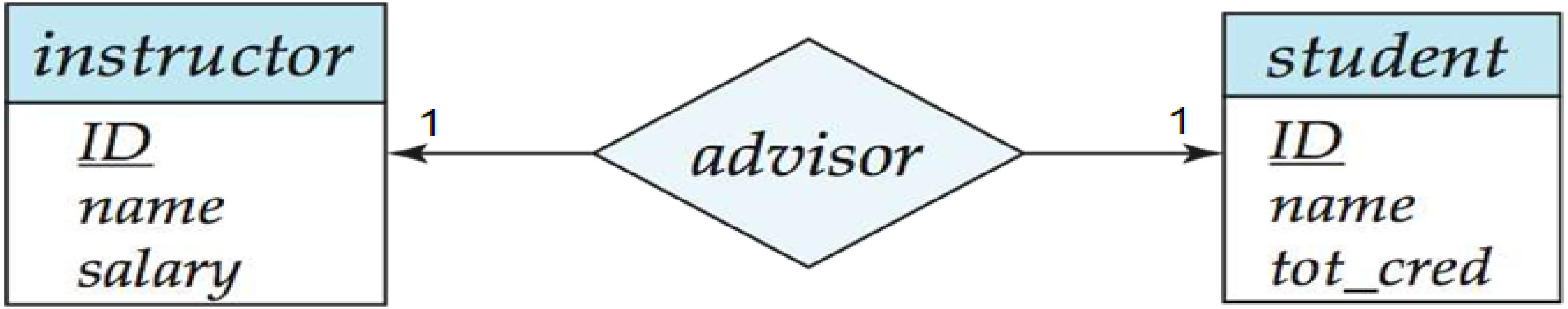


Each instance of B is related to a minimum of zero and a maximum of many instances of A

CARDINALITY CONSTRAINTS

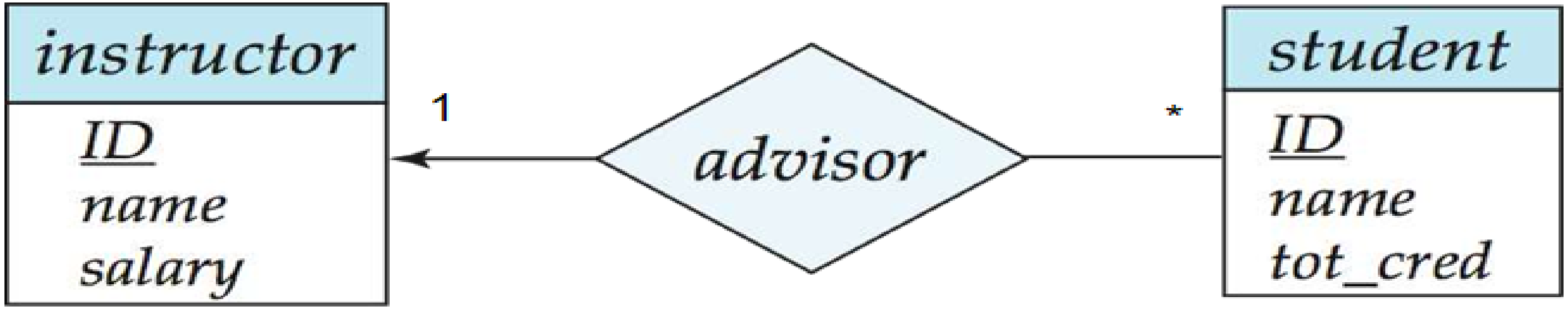
- We express cardinality constraints by drawing either a directed line (→), signifying "one," or an undirected line (—), signifying "many," between the relationship set and the entity set.
- Or, by numbering each entity. * or, m for many.

- One-to-one relationship:
  - A student is associated with at most one *instructor* via the relationship *advisor*
  - A *student* is associated with at most one *department* via *stud_dept*



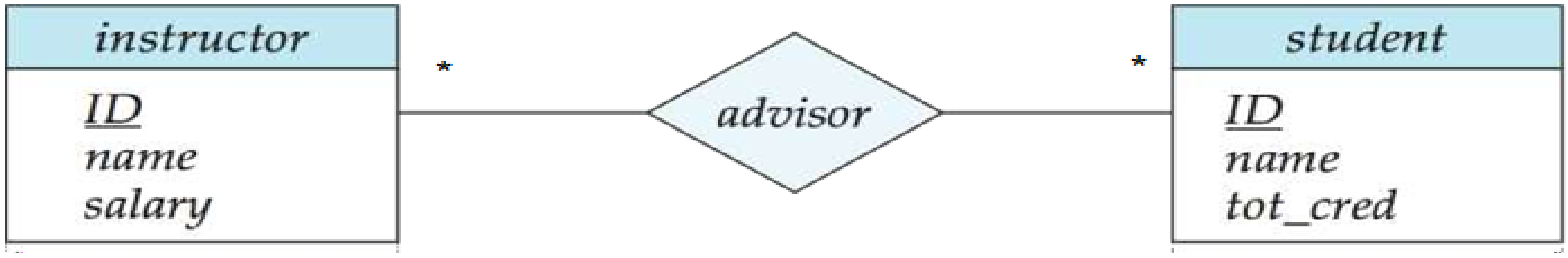
One-to-Many Relationship

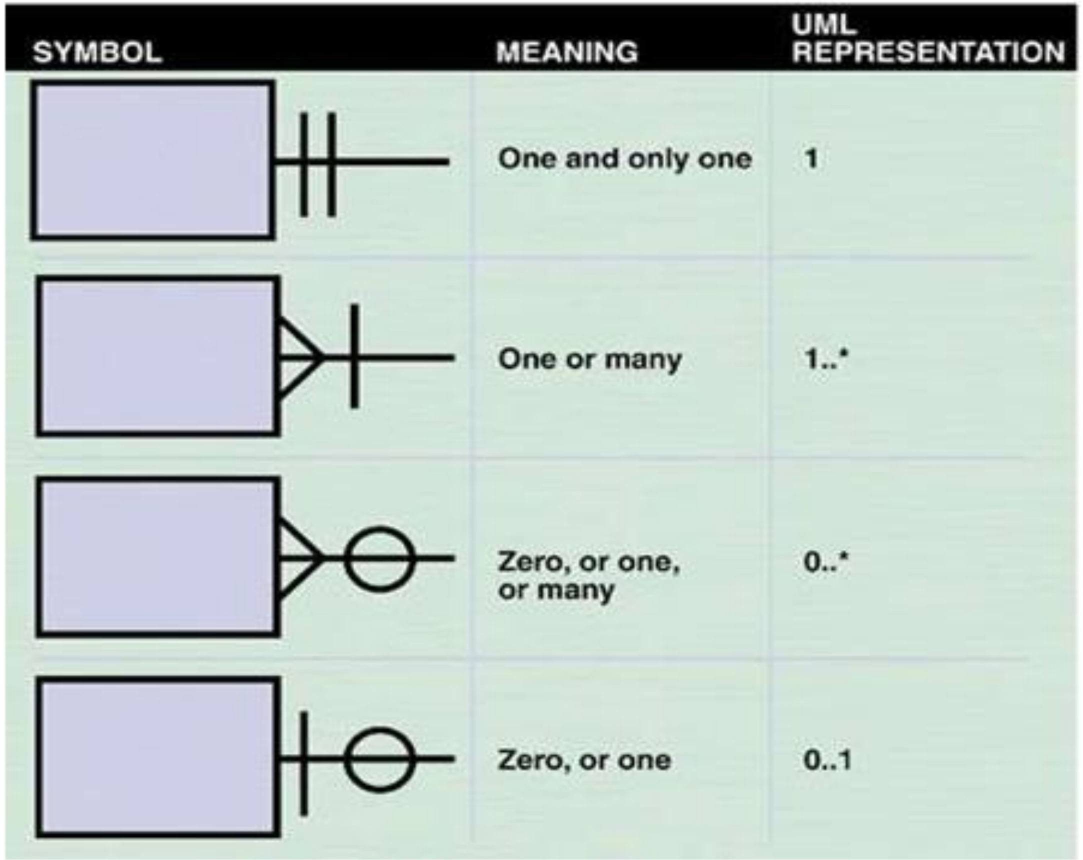
- one-to-many relationship between an *instructor* and a *student*
  - an instructor is associated with several (including 0) students via *advisor*
  - a student is associated with at most one instructor via advisor,



Many-to-Many Relationship

- An instructor is associated with several (possibly 0) students via *advisor*
- A student is associated with several (possibly 0) instructors via *advisor*

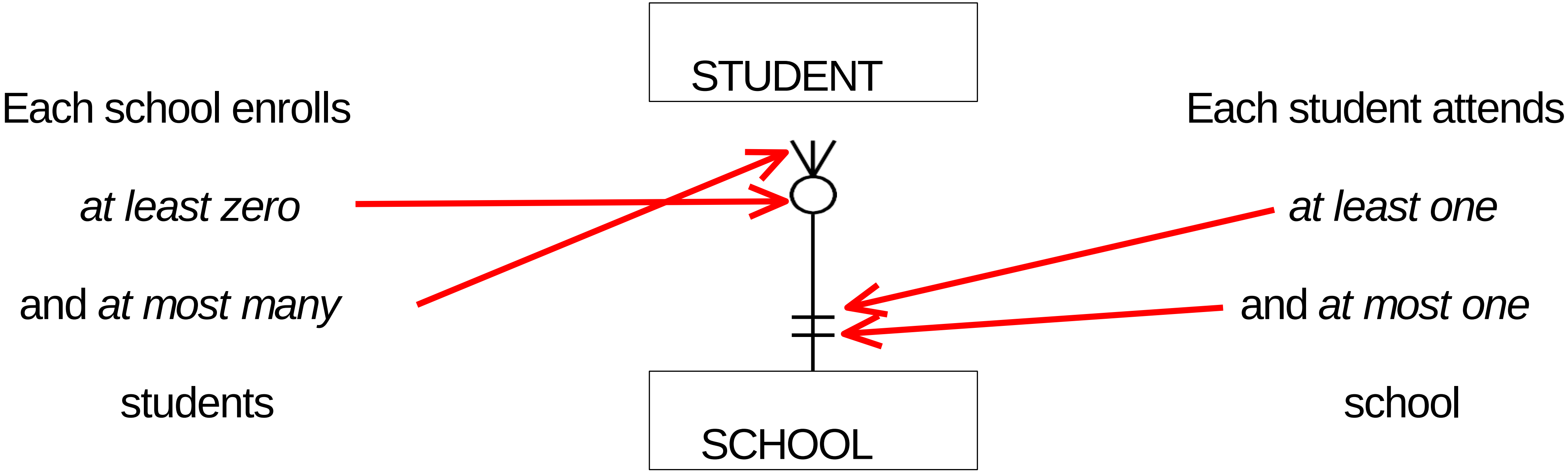




SYMBOL	MEANING	UML REPRESENTATION
	One and only one	1
	One or many	1..*
	Zero, or one, or many	0..*
	Zero, or one	0..1

Crow's foot notation is a common method of indicating cardinality. The four examples show how you can use various symbols to describe the relationships between entities.

CARDINALITY EXAMPLE



STUDENT

SCHOOL

Each school enrolls

at least zero

and *at most many*

students

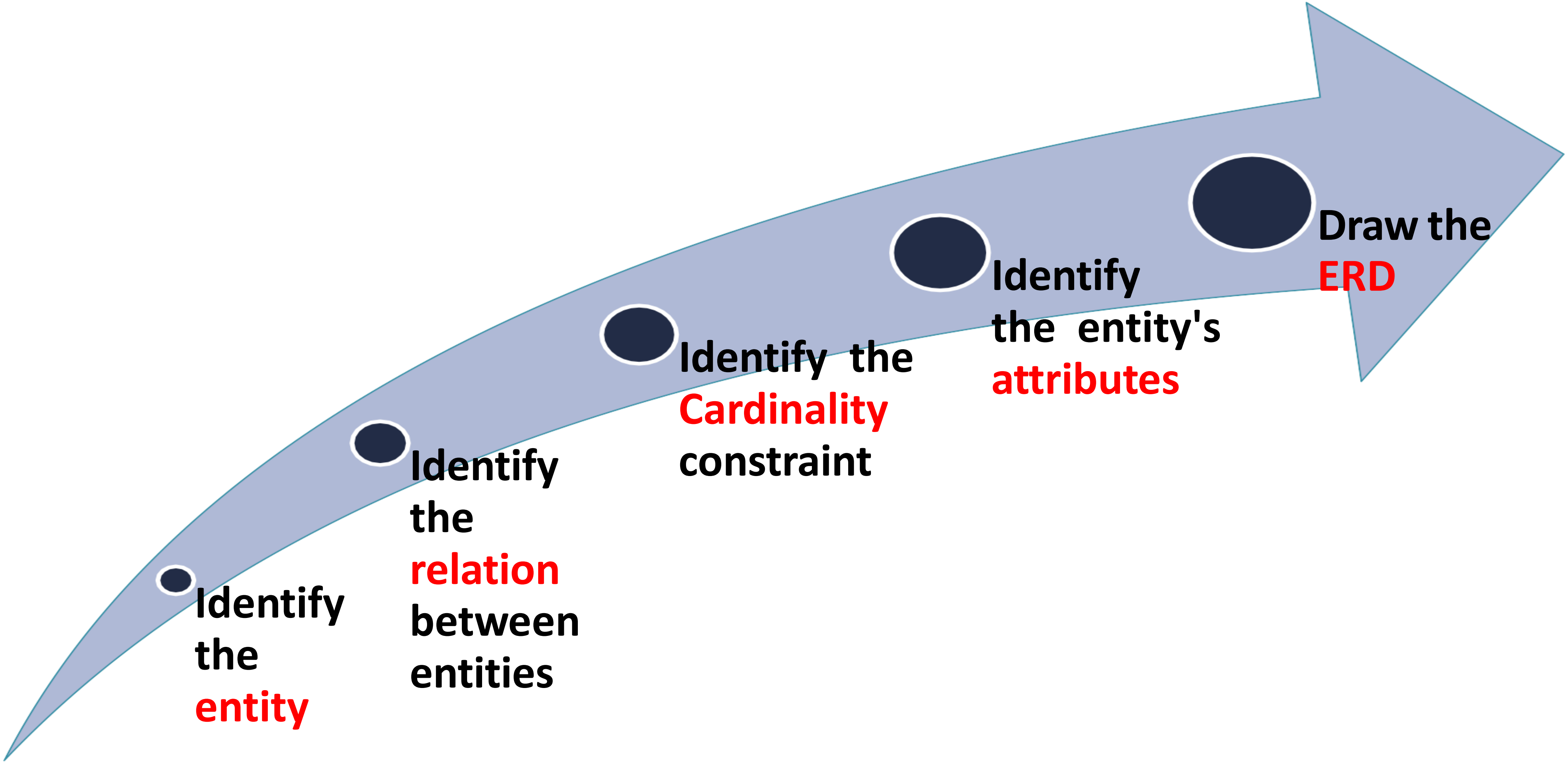
Each student attends

at least one

and *at most one*

school

General Steps to create an ERD



Identify the **entity**

Identify the **relation** between entities

Identify the **Cardinality** constraint

Identify the entity's **attributes**

Draw the **ERD**

A Simple Example

A *company has several departments. Each* department has a supervisor and at least one employee. Every supervisor has only one department under him. Employees must be assigned to at least one, but possibly more departments. At least one employee is assigned to a project, but an employee may be on vacation and not assigned to any projects. The important data fields **(entity)** are the names of the **departments, projects, supervisors** and **employees**, as well as the supervisor and employee number and a unique project number.

Identify Entities

- *A company has several **departments**. Each department has a **supervisor** and at least one **employee**. Every supervisor has only one department under him Employees must be assigned to at least one, but possibly more departments. At least one employee is assigned to a **project,** but an employee may be on vacation and not assigned to any projects. The important data fields are the names of the departments, projects, supervisors and employees, as well as the supervisor and employee number and a unique project number.*

- *A true entity should have more than one instance*

Identified Relationships

A Department is assigned an employee

A Department is run by a supervisor An

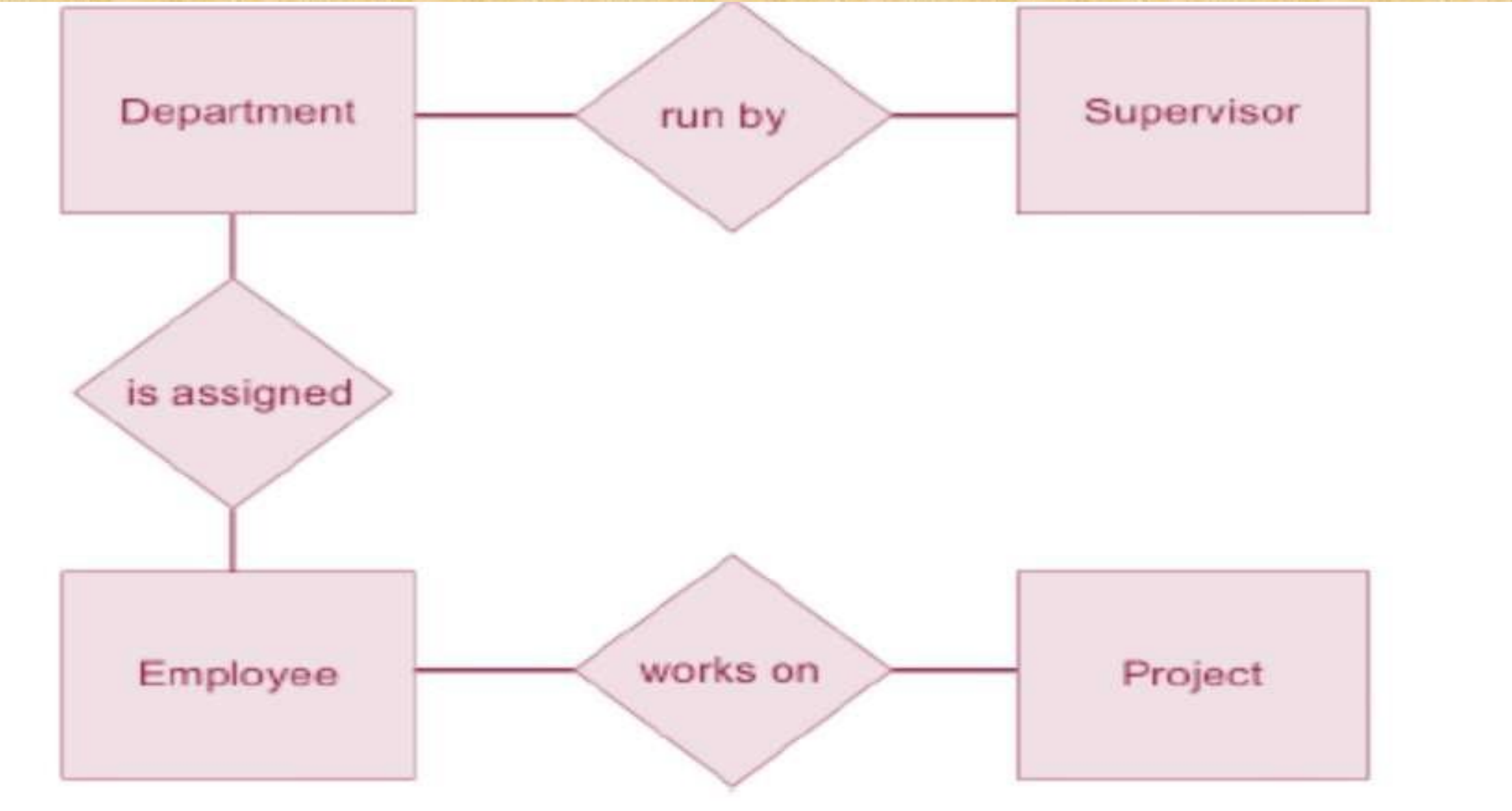
employee belongs to a department An

employee works on a project

A supervisor runs a department A

project uses an employee

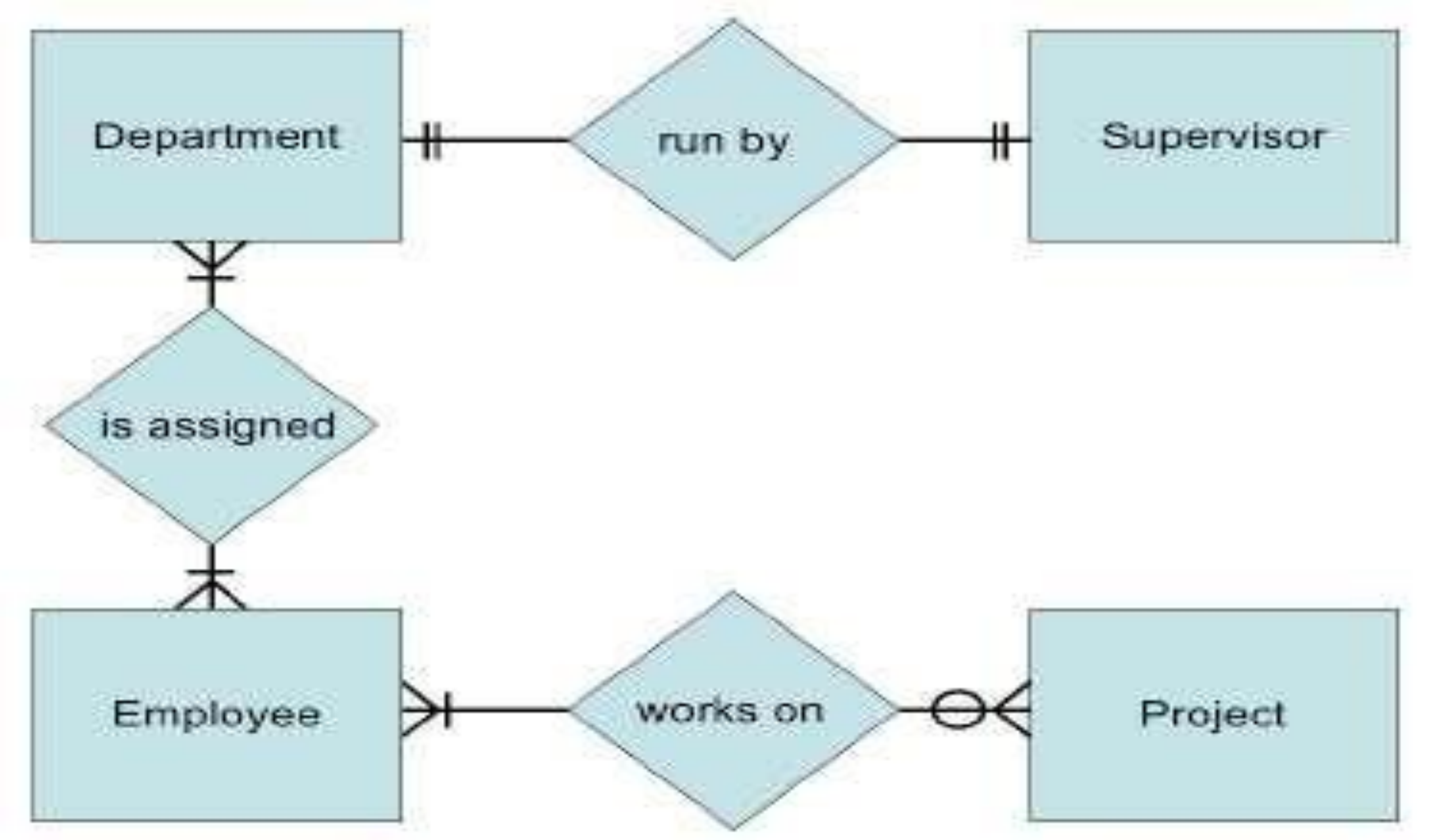
DRAWING ROUGH ERD



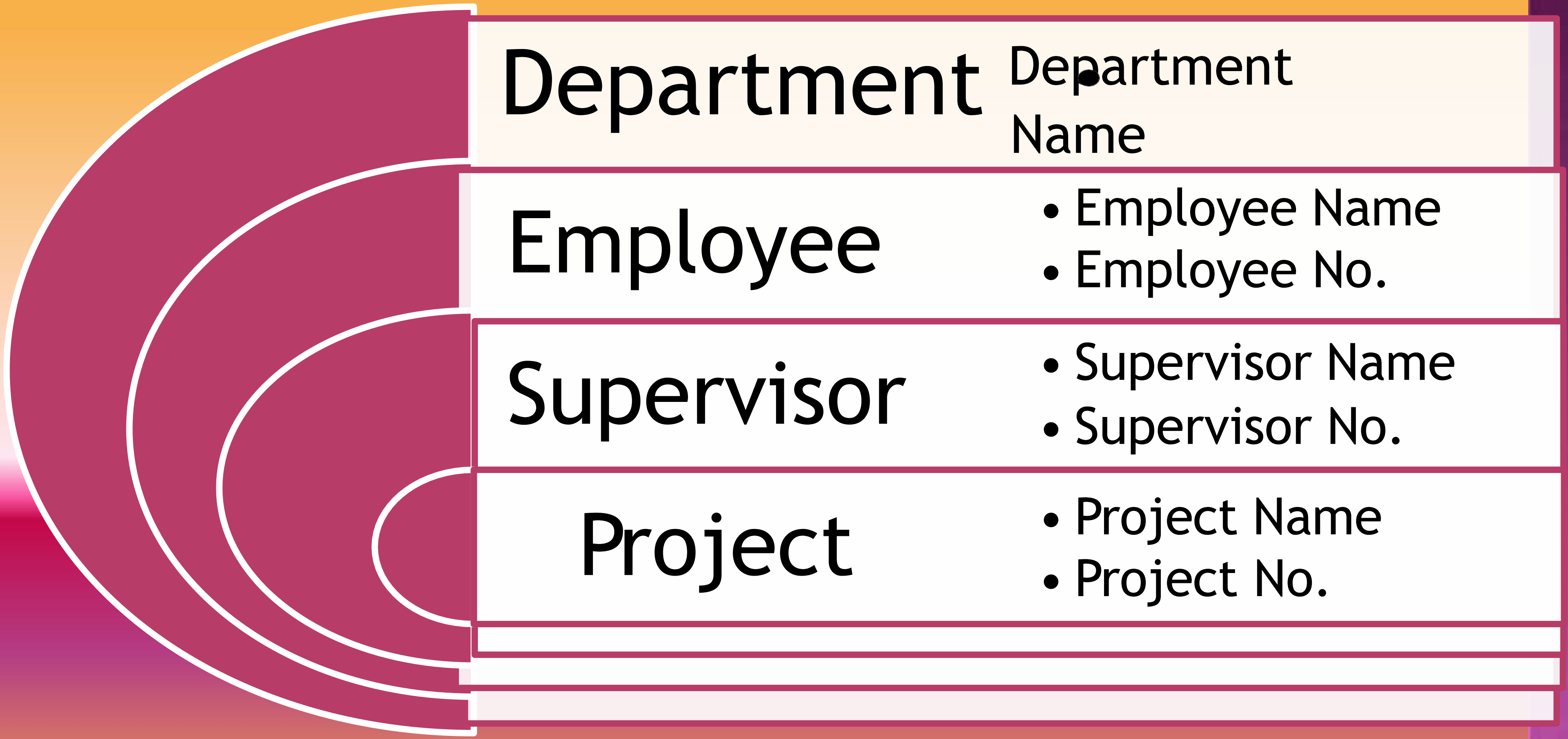
Fill in Cardinality

- Supervisor
  - Each department has **one** supervisor.
- Department
  - Each supervisor has **one** department.
  - Each employee can belong to **one or more** departments
- Employee
  - Each department must have **one or more** employees
  - Each project must have **one or more** employees
- Project
  - Each employee can have **0 or more** projects.

ERD WITH CARDINALITY



ATTRIBUTES



Department

- Department Name

Employee

- Employee Name
- Employee No.

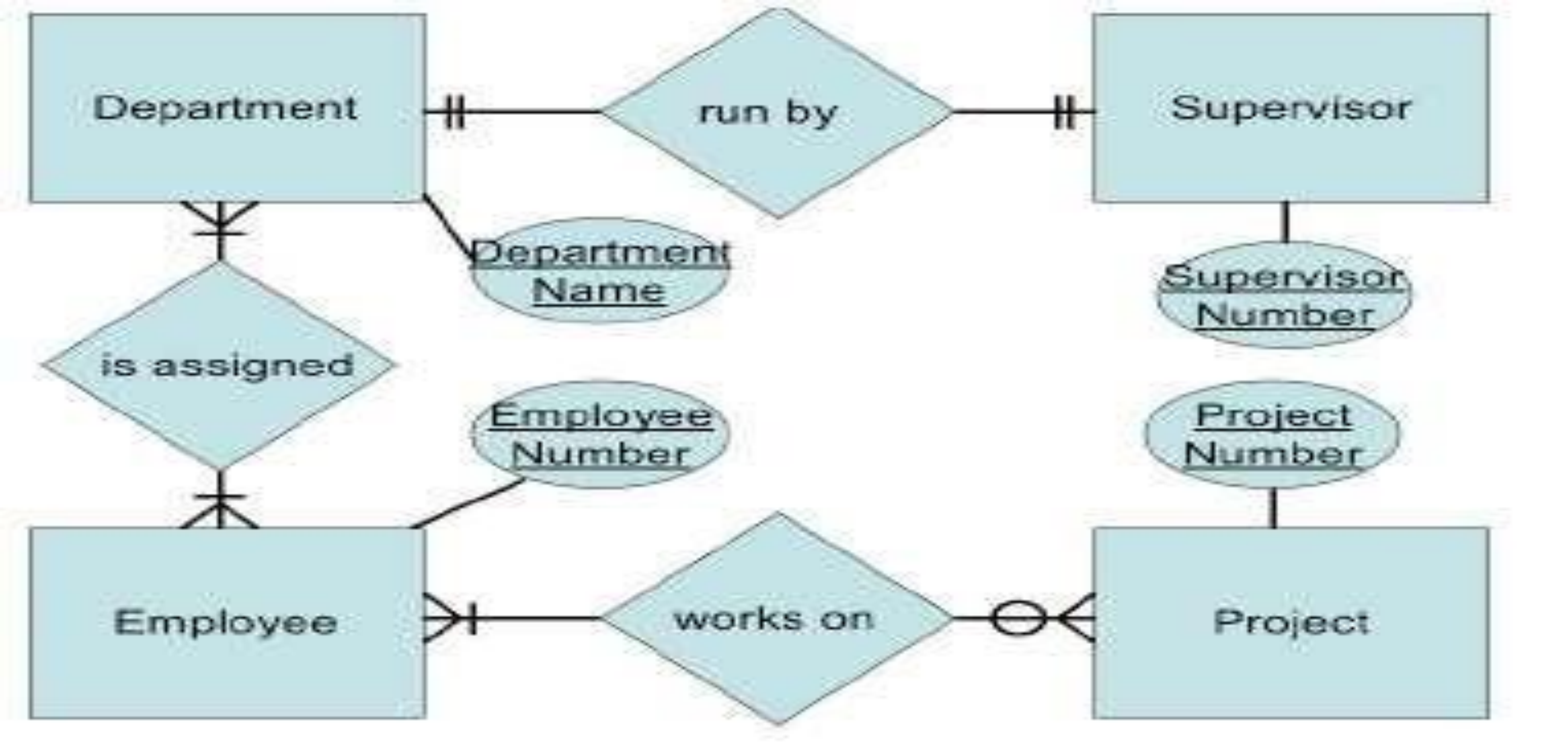
Supervisor

- Supervisor Name
- Supervisor No.

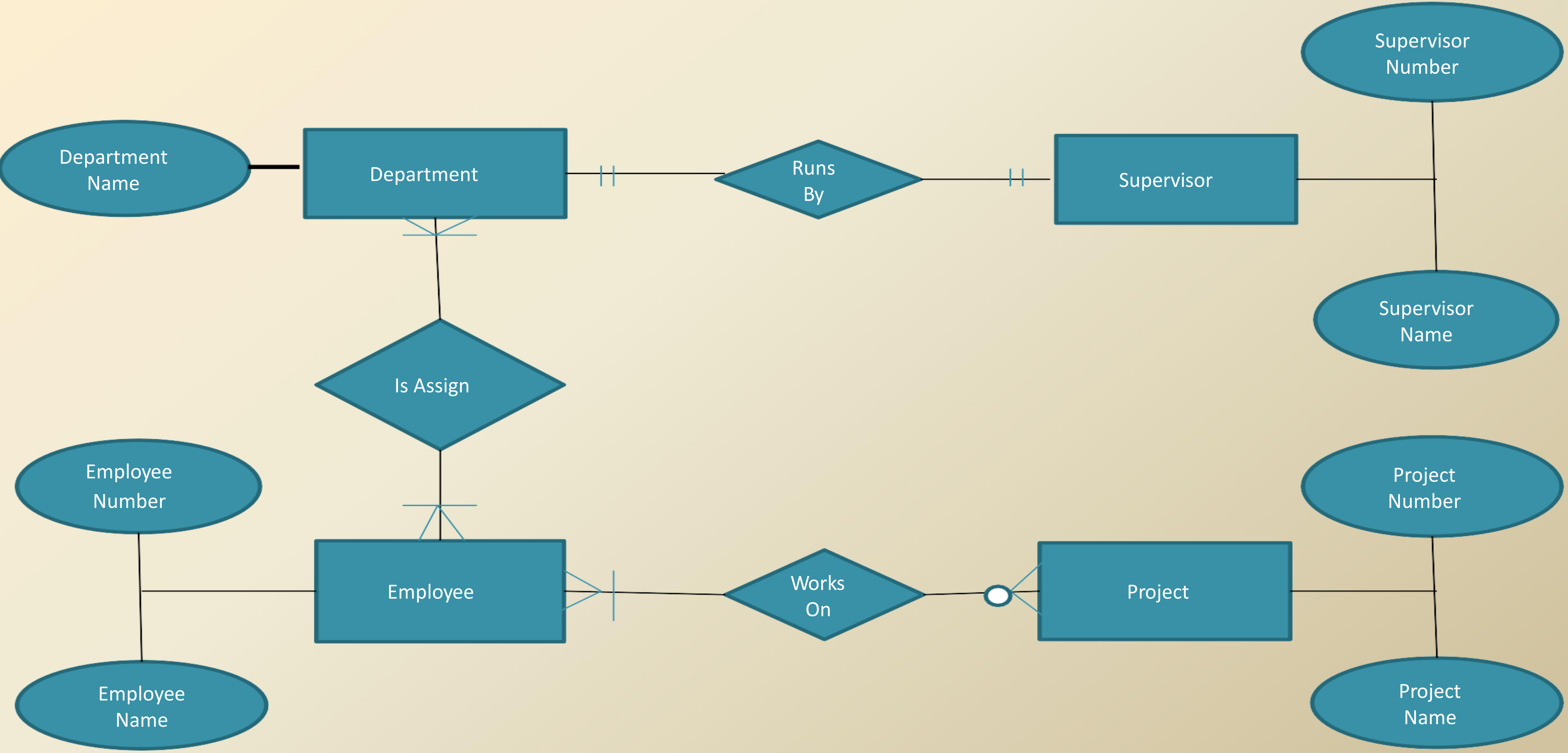
Project

- Project Name
- Project No.

Rough ERD Plus Primary Keys



Fully Attributed ERD



Thank You !!