Chapter 5: Database Design Using the E-R Model

Database System Concepts, 7th Ed.

Design Phases

- Initial phase -- characterize fully the data needs of the prospective database users.
- Second phase -- choosing a data model
  - Applying the concepts of the chosen data model
  - Translating these requirements into a conceptual schema of the database.
  - A fully developed conceptual schema indicates the functional requirements of the enterprise.
    - Describe the kinds of operations (or transactions) that will be performed on the data.

Design Phases (Cont.)

- Final Phase -- Moving from an abstract data model to the implementation of the database
  - Logical Design – Deciding on the database schema.
    - Database design requires that we find a "good" collection of relation schemas.
    - Business decision – What attributes should we record in the database?
    - Computer Science decision – What relation schemas should we have and how should the attributes be distributed among the various relation schemas?
  - Physical Design – Deciding on the physical layout of the database

Design Alternatives

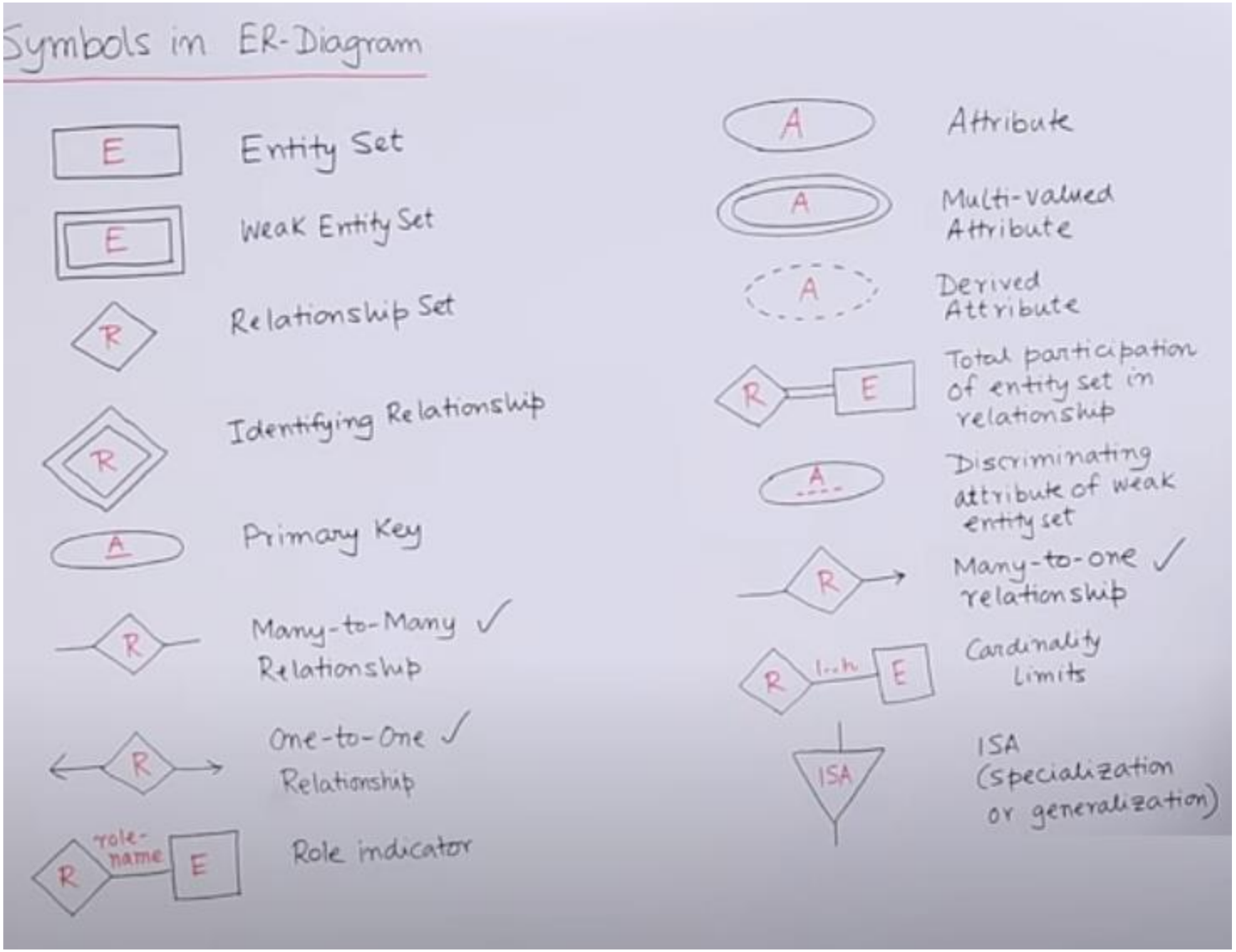
- In designing a database schema, we must ensure that we avoid two major pitfalls:
  - Redundancy: a bad design may result in repeat information.
    - Redundant representation of information may lead to data inconsistency among the various copies of information
  - Incompleteness: a bad design may make certain aspects of the enterprise difficult or impossible to model.
- Avoiding bad designs is not enough. There may be a large number of good designs from which we must choose.

Design Approaches

- Entity Relationship Model (covered in this chapter)
  - Models an enterprise as a collection of *entities* and *relationships*
    - Entity: a “thing” or “object” in the enterprise that is distinguishable from other objects
      - Described by a set of *attributes*
    - Relationship: an association among several entities
  - Represented diagrammatically by an *entity-relationship diagram:*

Outline of the ER Model

Symbols in ER-Diagram



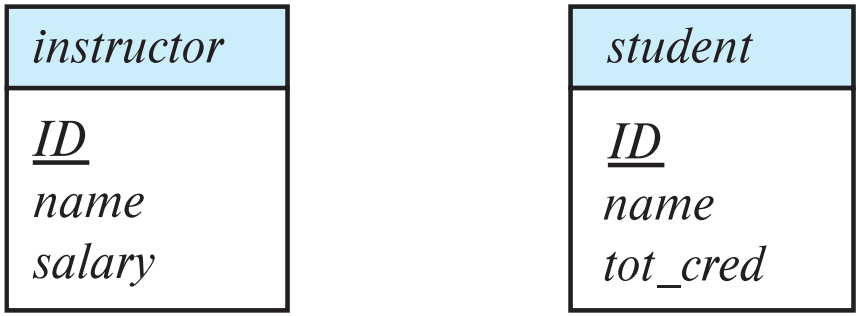
Entity Sets

- An **entity** is an object that exists and is distinguishable from other objects.
  - Example: specific person, company, event, plant
- An **entity set** is a set of entities of the same type that share the same properties.
  - Example: set of all persons, companies, trees, holidays
- An entity is represented by a set of attributes; i.e., descriptive properties possessed by all members of an entity set.
  - Example:

    instructor = (*ID, name, salary*)
    course= (*course_id, title, credits*)
- A subset of the attributes form a **primary key** of the entity set; i.e., uniquely identifying each member of the set.

Representing Entity sets in ER Diagram

- Entity sets can be represented graphically as follows:
  - Rectangles represent entity sets.
  - Attributes listed inside entity rectangle
  - Underline indicates primary key attributes



Relationship Sets

- A **relationship** is an association among several entities

  Example:

44553 (Peltier)	*advisor*	22222 (Einstein)
student entity	relationship set	*instructor* entity

- A **relationship set** is a mathematical relation among $n \geq 2$ entities, each taken from entity sets

$$\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$$

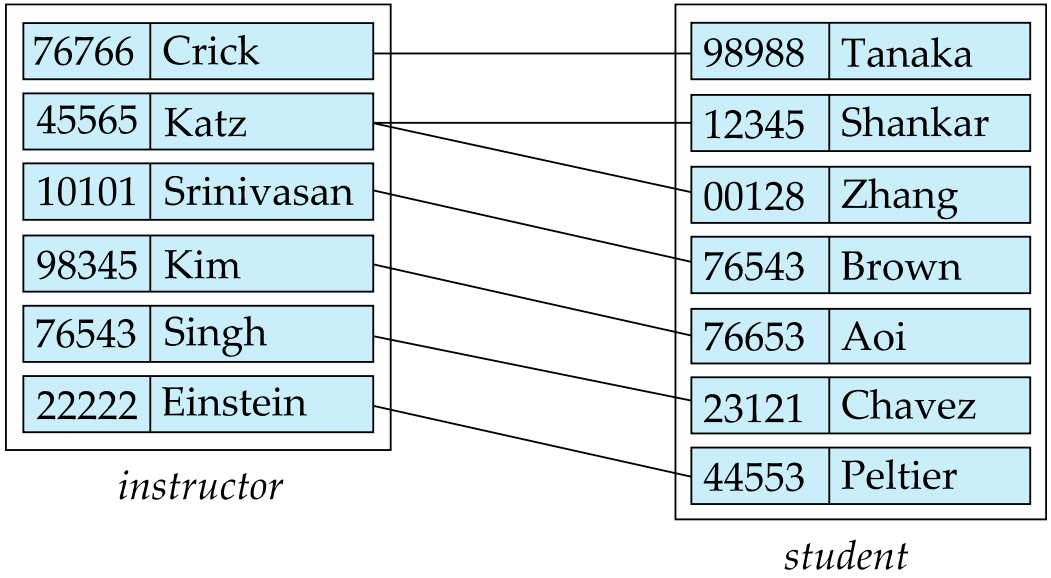
  where $(e_1, e_2, \ldots, e_n)$ is a relationship

  - Example:

$$(44553,22222) \in advisor$$

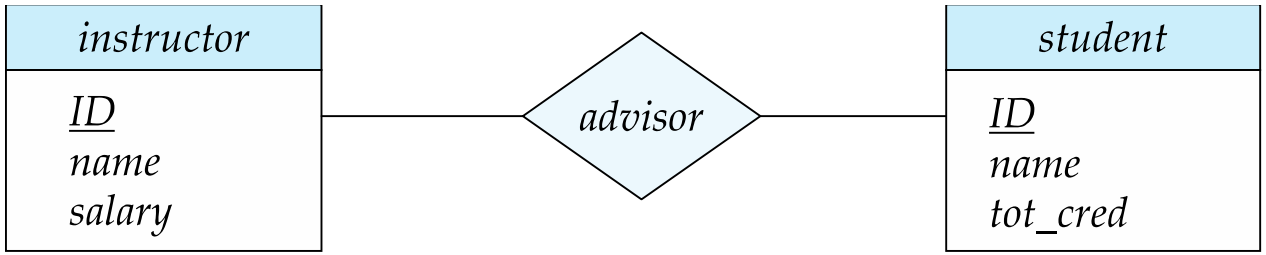
Relationship Sets (Cont.)

- Example: we define the relationship set *advisor* to denote the associations between students and the instructors who act as their advisors.
- Pictorially, we draw a line between related entities.



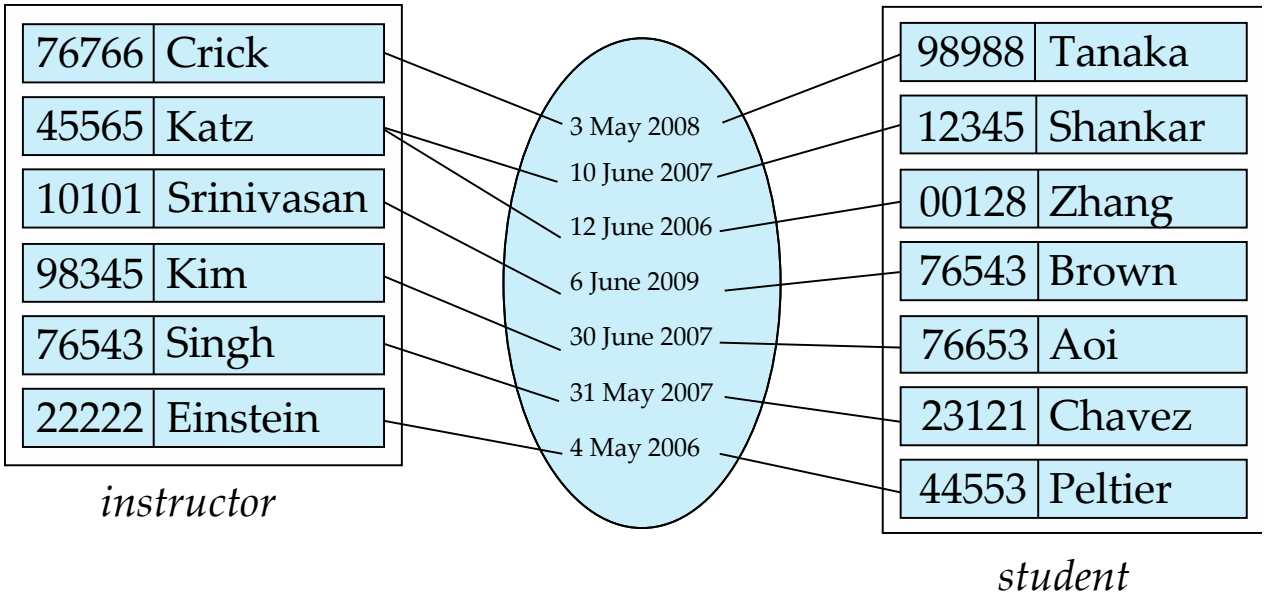
Representing Relationship Sets via ER Diagrams

- Diamonds represent relationship sets.

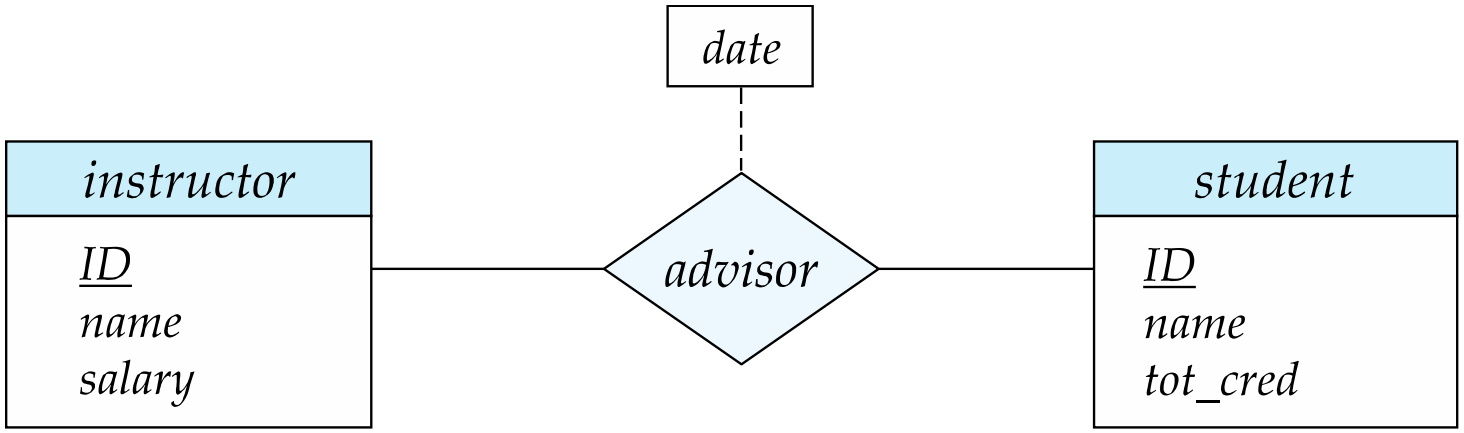


Relationship Sets (Cont.)

- An attribute can also be associated with a relationship set.
- For instance, the *advisor* relationship set between entity sets *instructor* and *student* may have the attribute *date* which tracks when the student started being associated with the advisor

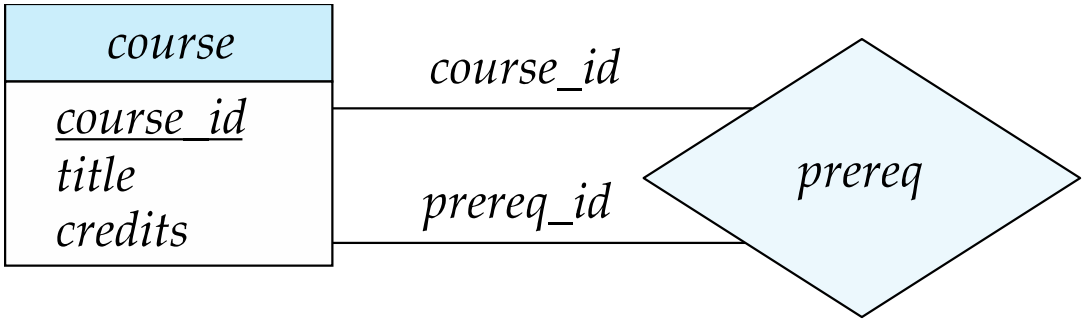


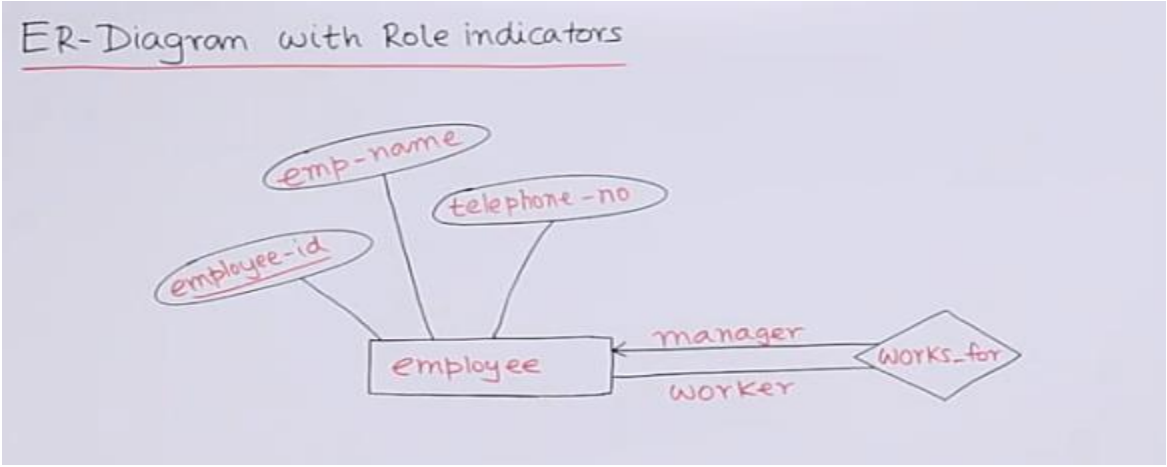
Relationship Sets with Attributes



Roles

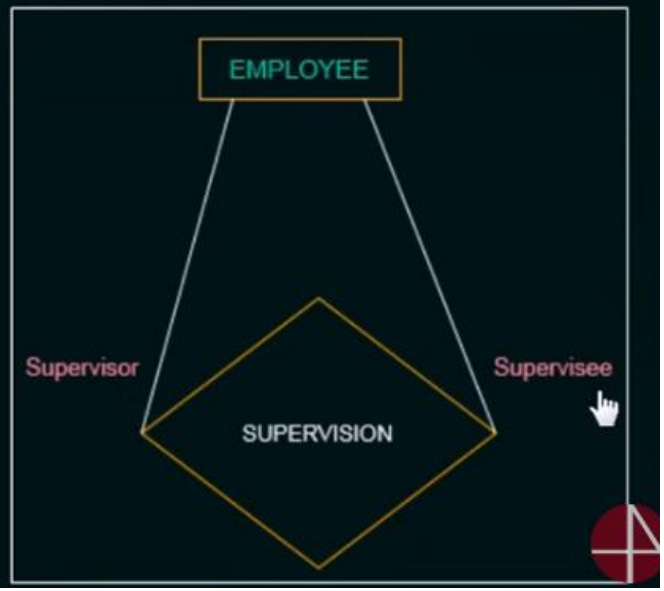
- Entity sets of a relationship need not be distinct
  - Each occurrence of an entity set plays a “role” in the relationship
- The labels “*course_id*” and “*prereq_id*” are called **roles**.





Degree of a Relationship Set

- **Relationships**
  - **Role Names**
    - Signifies the role that a participating entity plays in each relationship instance.
  - **Recursive Relationships**
    - Same entity type participates more than once in a relationship type in different roles.



nesoacademy.org

Degree of a Relationship Set

- Binary relationship
  - involve two entity sets (or degree two).
  - most relationship sets in a database system are binary.
- Relationships between more than two entity sets are rare. Most relationships are binary. (More on this later.)
  - Example: *students* work on research *projects* under the guidance of an *instructor*.
  - relationship *proj_guide* is a ternary relationship between *instructor, student,* and *project*

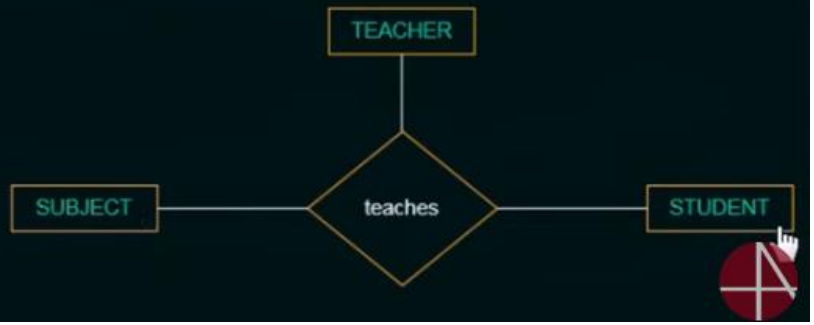
Degree of a Relationship Set

Relationships

★ Degree of Relationship:

2. Binary relationship:
   - ➢ Exists when there is an association among two entities.

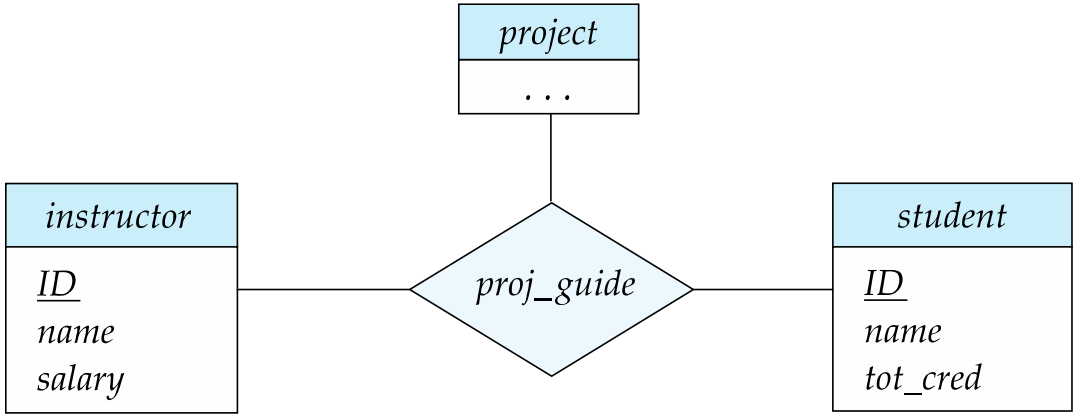
3. Ternary relationship:
   - ➢ Exists when there is an association among three entities.



nesoacademy.org

Non-binary Relationship Sets

- Most relationship sets are binary
- There are occasions when it is more convenient to represent relationships as non-binary.
- E-R Diagram with a Ternary Relationship



Complex Attributes

- Attribute types:
  - **Simple** and **composite** attributes.
  - **Single-valued** and **multivalued** attributes
    - Example: multivalued attribute: *phone_numbers*
  - **Derived** attributes
    - Can be computed from other attributes
    - Example: age, given date_of_birth
- **Domain** – the set of permitted values for each attribute

Complex Attributes

- **Simple Attribute**:
  - Example: "Height"
  - Description: A straightforward, indivisible attribute without internal structure. For instance, the height of a person would be a simple attribute.
- **Composite Attribute**:
  - Example: "Address" (composed of street, city, state, and postal code)
  - Description: A composite attribute is made up of multiple simple attributes. For instance, an address comprises various components like street, city, state, and postal code.

Complex Attributes

- **Single-Valued Attribute**:
  - Example: "Date of Birth"
  - Description: An attribute having a single value for an entity. Each person has one specific date of birth.
- **Multi-Valued Attribute**:
  - Example: "Phone Numbers"
  - Description: An attribute that can hold multiple values for an entity. For instance, a person might have multiple phone numbers (home, work, mobile), making "Phone Numbers" a multi-valued attribute.

Complex Attributes

- **Derived Attribute**:
  - Example: "Age" (derived from "Date of Birth")
  - Description: An attribute whose value is derived or computed based on another attribute(s). For example, age can be calculated by subtracting the date of birth from the current date.
- **Domain**:
  - Example: "Gender" (permitted values: Male, Female, Other)
  - Description: The set of permitted values for an attribute. In this case, the domain for the "Gender" attribute would consist of predefined options like Male, Female, or Other, limiting the possible values for this attribute.
-

Complex Attributes

Single-Valued Attributes

- ★ Have a single value for a particular entity.
- ★ Ex: Age → single-valued attribute of a person.

Multivalued Attributes

- ★ Can have set of values for a particular entity.
- ★ Ex: College degree, languages known → multivalued attributes of a

Complex Attributes

Derived Attributes

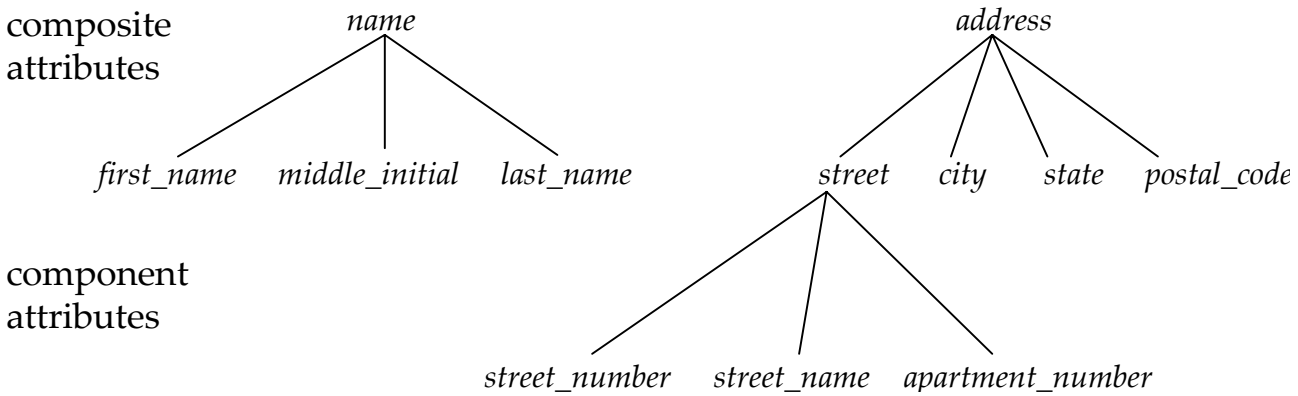
- Can be derived from other attributes.
- Ex: Age → can be derived from date of birth.

Stored Attributes

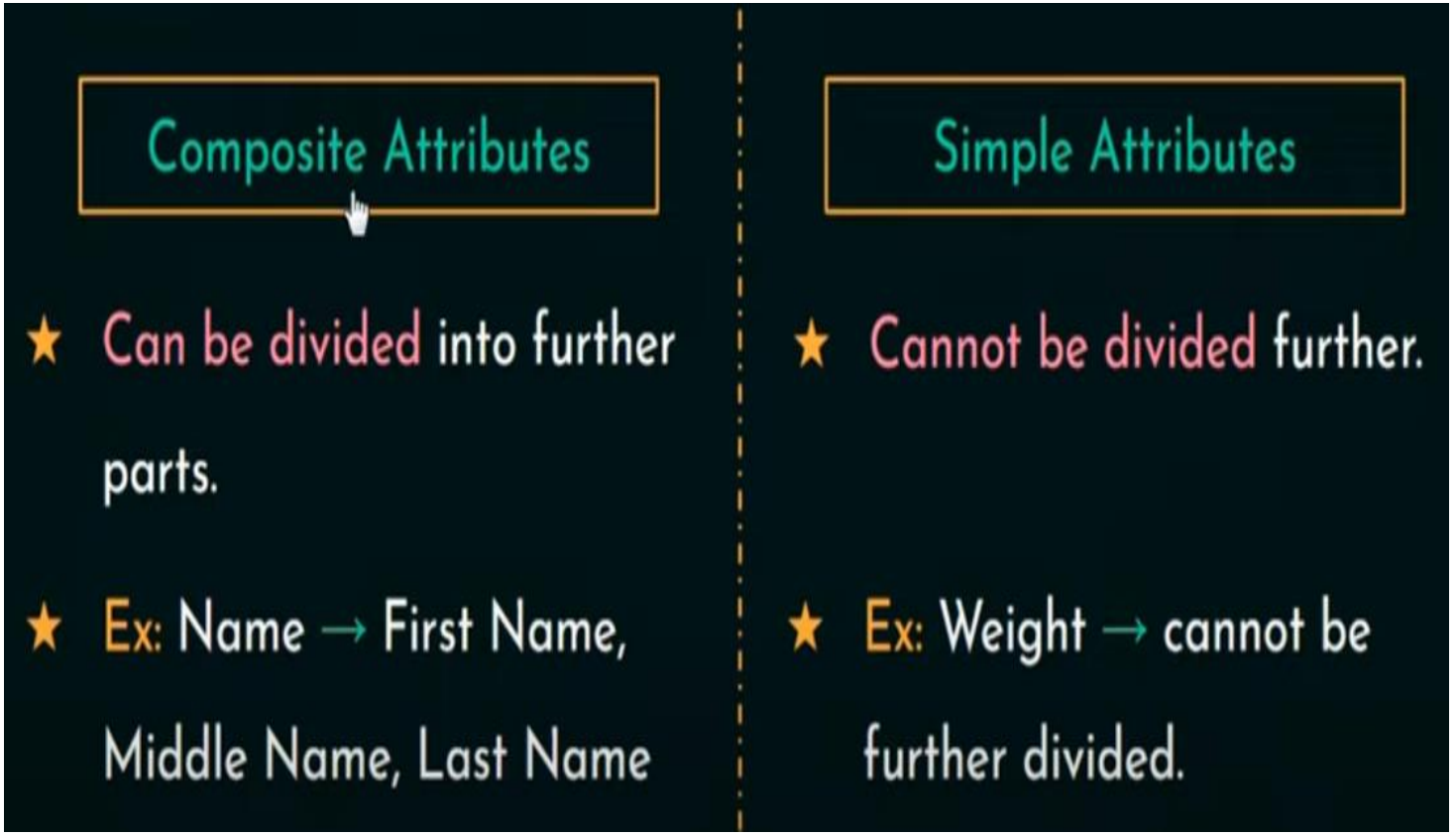
- From which the value of other attributes are derived.
- Ex: BirthDate of a person

Composite Attributes

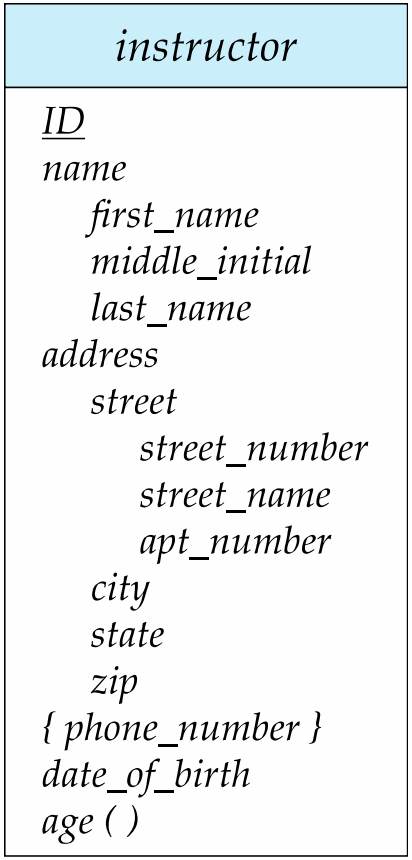
- Composite attributes allow us to divided attributes into subparts (other attributes).



Representing Complex Attributes in ER Diagram



Representing Complex Attributes in ER Diagram



Representing Complex Attributes in ER Diagram

★ Complex Attributes:

- Has multivalued & composite components in it.
- Multivalued attributes → represented within '{ }'.
- Composite attributes → represented within '( )'.
- Ex: {CollegeDegrees(College, Year, Degree, Field)}

★ Null Values:

- Null is something which is not applicable or unknown.

STUDENT_ID	NAME	AGE	PHONE_NO
1	HARRY	20	123546897
2	JEREMY	22	NULL
3	JOSHUA	NULL	25254689

nesoacademy.org

Representing Complex Attributes in ER Diagram

★ Entity Type:

- A collection of entities that have the same attributes.
- Ex: STUDENT

STUDENT

STUDENT_ID	NAME	AGE
1	HARRY	20
2	JEREMY	22
3	JOSHUA	18

★ Entity Set:

- Collection of entities of a particular entity type at a point in time.

Representing Complex Attributes in ER Diagram

★ Key Attribute:

- That attribute that is capable of identifying each entity uniquely.
- Ex: Roll number of a student

STUDENT

STUDENT_ID	NAME	AGE
1	PRIYANKA	20
2	JEREMY	21
3	PRIYANKA	20

★ Value Set of Attributes:

- The set of values that can be assigned to an attribute.

Representing Complex Attributes in ER Diagram

Consider an entity "Car" with a complex attribute "Engine," which comprises multiple details such as "Horsepower," "Fuel Type," and "Cylinder Count." This composite attribute can be represented within the ER diagram as a set of connected attributes.

plaintext Copy code

```
Car
└─ Engine
   ├─ Horsepower
   ├─ Fuel Type
   └─ Cylinder Count
```

In this representation, "Engine" is a composite attribute composed of three sub-attributes.

Representing Complex Attributes in ER Diagram

For instance, let's take an entity "Person" with a multivalued attribute "Skills," which could possess multiple skills such as "Programming," "Communication," and "Leadership."

plaintext Copy code

```
Person
└─ Skills
   ├─ Programming
   ├─ Communication
   └─ Leadership
```

Here, "Skills" is a multivalued attribute containing multiple individual skill attributes.

Representing Complex Attributes in ER Diagram

Entity: **Household**

Composite Attribute: **Address**

Attributes within the composite attribute:

- Street
- City
- State
- Postal Code

plaintext Copy code

```
Household
└─ Address
   ├─ Street
   ├─ City
   ├─ State
   └─ Postal Code
```

Here, the "Address" attribute is composite, consisting of sub-attributes that represent different parts of an address.

Representing Complex Attributes in ER Diagram

Entity: **Household**

Multivalued Attribute: **Phone Numbers**

Attributes within the multivalued attribute:

- Home Number
- Work Number
- Mobile Number

plaintext Copy code

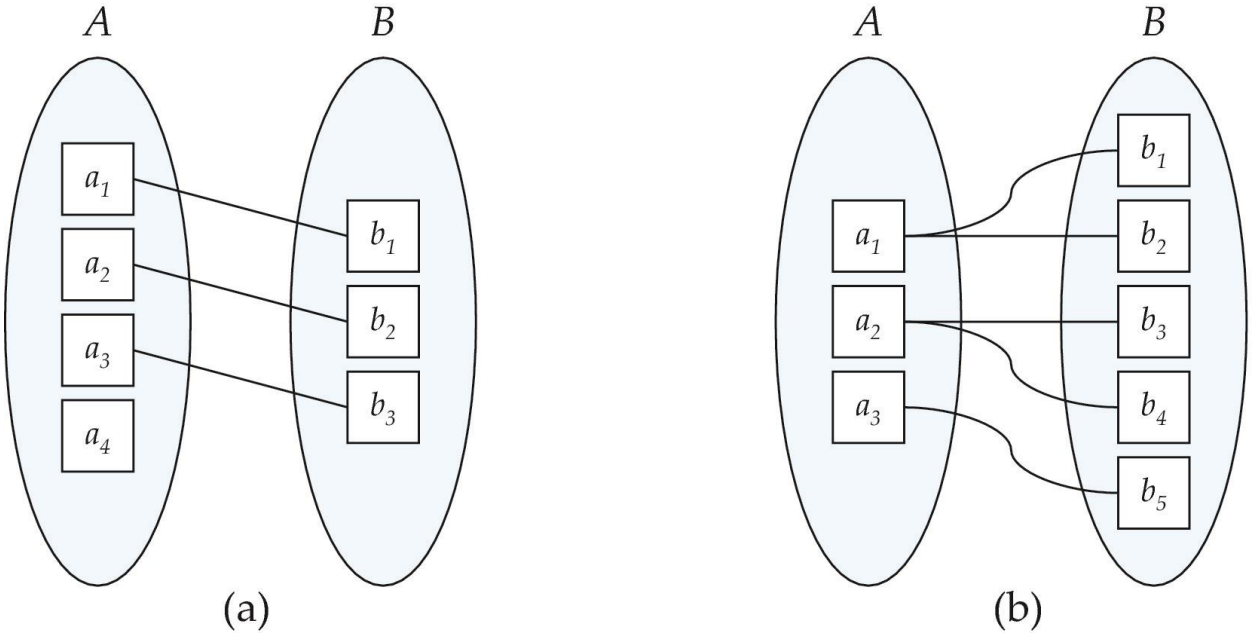
```
Household
└─ Phone Numbers
   ├─ Home Number
   ├─ Work Number
   └─ Mobile Number
```

In this case, the "Phone Numbers" attribute is multivalued, containing various sub-attributes for different types of phone numbers associated with the household.

Mapping Cardinality Constraints

- Express the number of entities to which another entity can be associated via a relationship set.
- Most useful in describing binary relationship sets.
- For a binary relationship set the mapping cardinality must be one of the following types:
  - One to one
  - One to many
  - Many to one
  - Many to many

Mapping Cardinalities

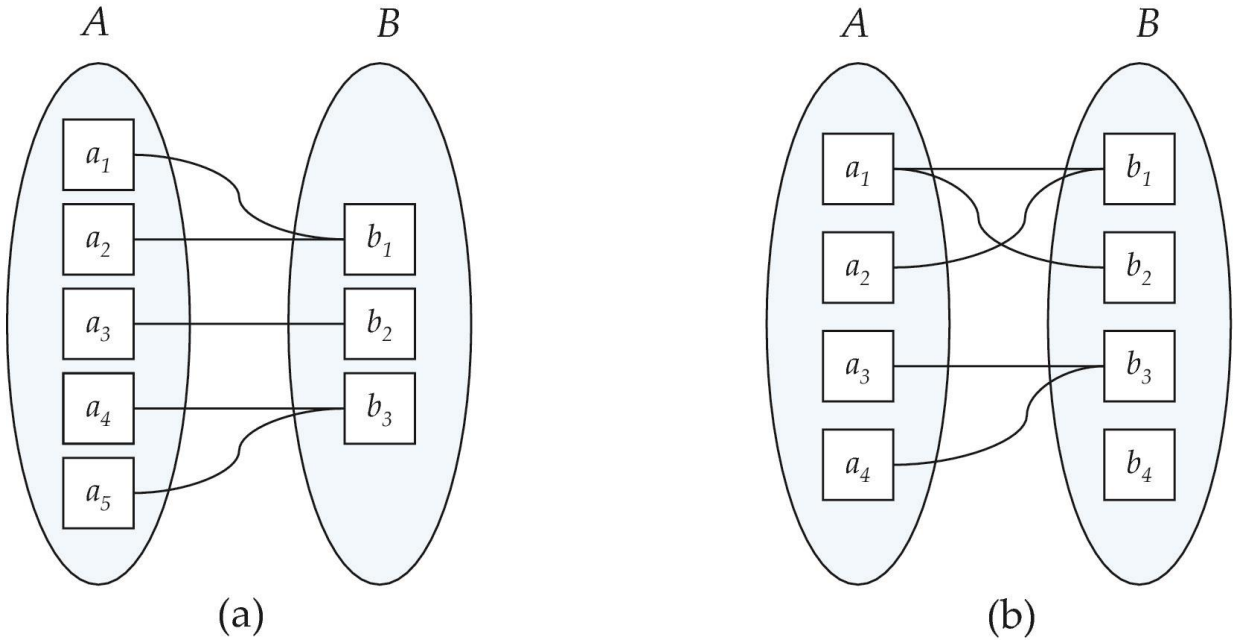


One to one

One to many

Note: Some elements in *A* and *B* may not be mapped to any elements in the other set

Mapping Cardinalities



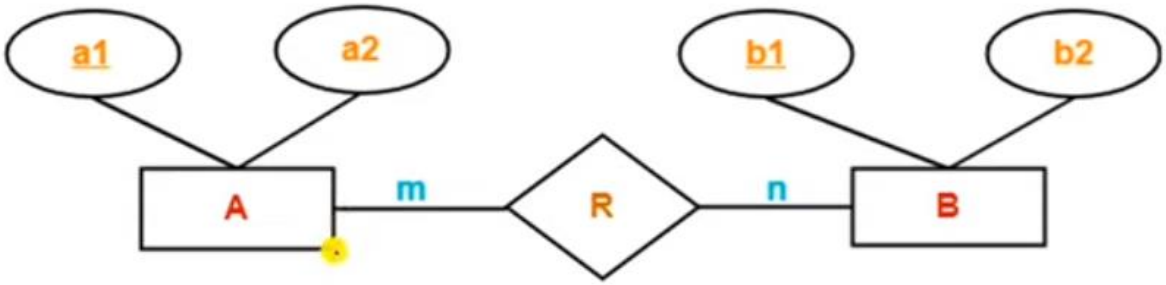
Many to one

Many to many

Note: Some elements in A and B may not be mapped to any elements in the other set

Mapping Cardinality Constraints

Case-1: For Binary Relationship with Cardinality Ratio m:n

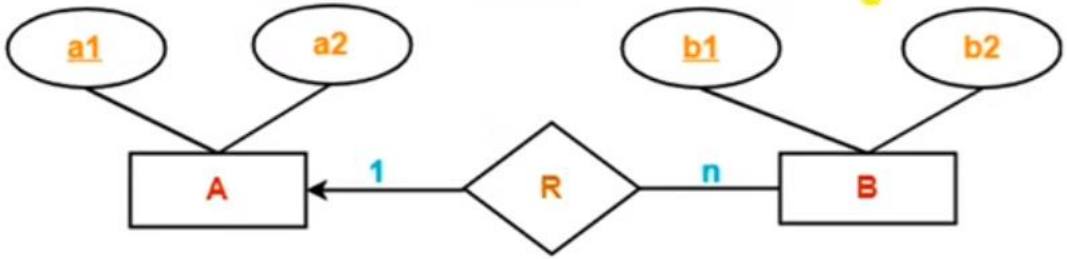


In ***Many-to-Many relationship***, **three tables** will be required-

1. **A (a1 , a2)**
2. **R (a1 , b1)**
3. **B (b1 , b2)**

Mapping Cardinality Constraints

Case-2: For Binary Relationship with Cardinality Ratio 1:n



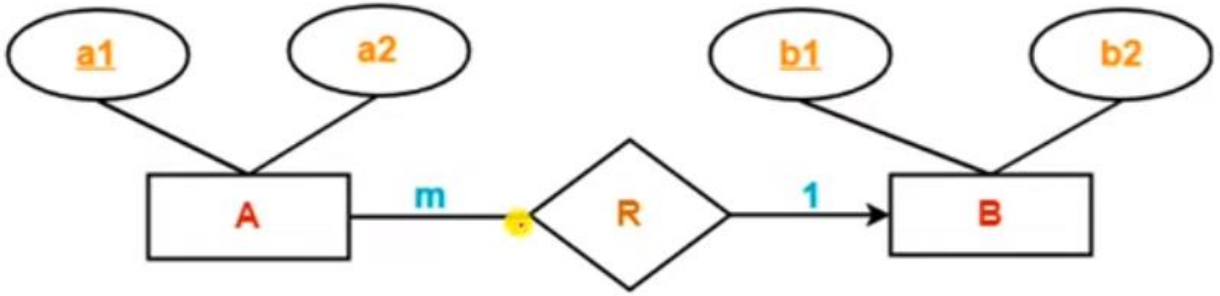
In ***One-to-Many relationship***, **two tables** will be required-

1. **A (a1 , a2)**
2. **BR (b1 , b2, a1)**

NOTE - Here, combined table will be drawn for the entity set B and relationship set R.

Mapping Cardinality Constraints

Case-3: For Binary Relationship with Cardinality Ratio m:1



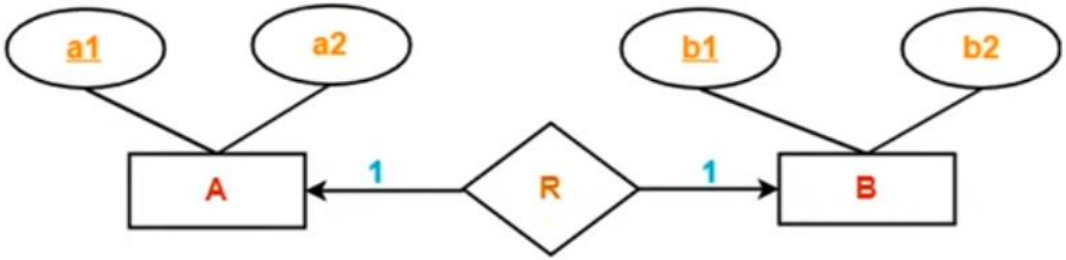
In ***Many-to-One relationship***, **two tables** will be required-

1. **AR (a1 , a2 , b1)**
2. **B (b1 , b2)**

NOTE - Here, combined table will be drawn for the entity set A and relationship set R.

Mapping Cardinality Constraints

Case-4: For Binary Relationship with Cardinality Ratio 1:1



In ***One-to-One relationship***, **two tables** will be required. Either combine 'R' with 'A' or 'B'

Way-01:

1. AR (a1 , a2 , b1)
2. B (b1 , b2)

Way-02:

1. A (a1 , a2)
2. BR (b1 , b2, a1)

Mapping Cardinalities

One-to-One Relationship:

1. **Passport - Person Relationship**:

   Entity 1: **Passport**

   Entity 2: **Person**

   Mapping: One person has exactly one passport, and each passport belongs to only one person.

```plaintext
Passport ———————— Person
```

2. **University - President Relationship**:

   Entity 1: **University**

   Entity 2: **President**

   Mapping: Each university has only one president, and a president leads only one university.

```plaintext
University ———————— President
```

Mapping Cardinalities

One-to-Many Relationship:

1. **Department - Employee Relationship**:

   Entity 1: **Department**

   Entity 2: **Employee**

   Mapping: A department can have many employees, but each employee works in only one department.

```
plaintext                                                    Copy code

Department ——————→ Employee
```

2. **Country - Cities Relationship**:

   Entity 1: **Country**

   Entity 2: **Cities**

   Mapping: A country can have multiple cities, but each city belongs to only one country.

```
plaintext                                                    Copy code

Country ——————→ Cities
```

Mapping Cardinalities

Many-to-One Relationship:

1. **Students - University Relationship**:

   Entity 1: **Students**

   Entity 2: **University**

   Mapping: Many students attend one university, but each student attends only one university.

```plaintext
Students  ←——  University
```

2. **Employees - Department Relationship**:

   Entity 1: **Employees**

   Entity 2: **Department**

   Mapping: Multiple employees work in one department, but each employee works in only one department.

```plaintext
Employees  ←——  Department
```

Mapping Cardinalities

Many-to-Many Relationship:

1. **Authors - Books Relationship**:

   Entity 1: **Authors**

   Entity 2: **Books**

   Mapping: Many authors can write many books, and a book can have multiple authors.

```
plaintext                                                    Copy code

Authors ———— Books
```

2. **Courses - Students Relationship**:

   Entity 1: **Courses**

   Entity 2: **Students**

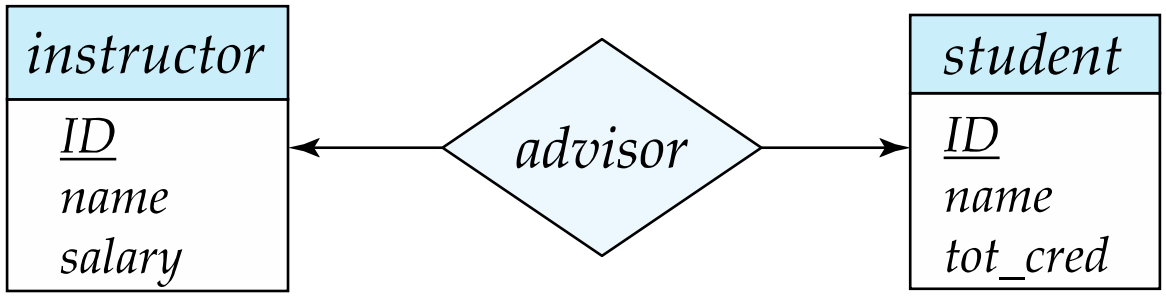
   Mapping: Many students can enroll in many courses, and a course can have multiple students.

```
plaintext                                                    Copy code

Courses ———— Students
```

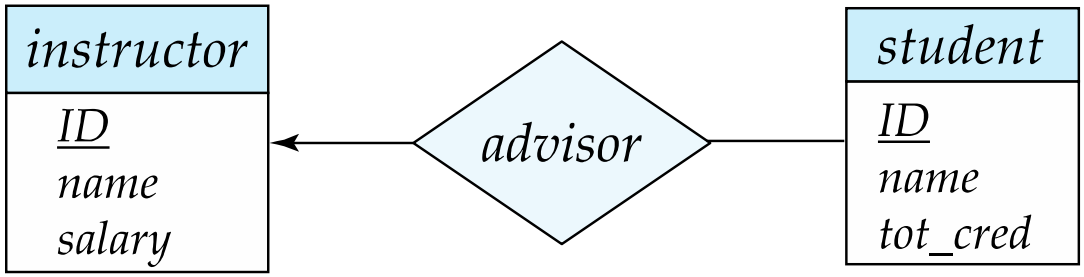
Representing Cardinality Constraints in ER Diagram

- 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.
- One-to-one relationship between an *instructor* and a *student* :
  - 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*



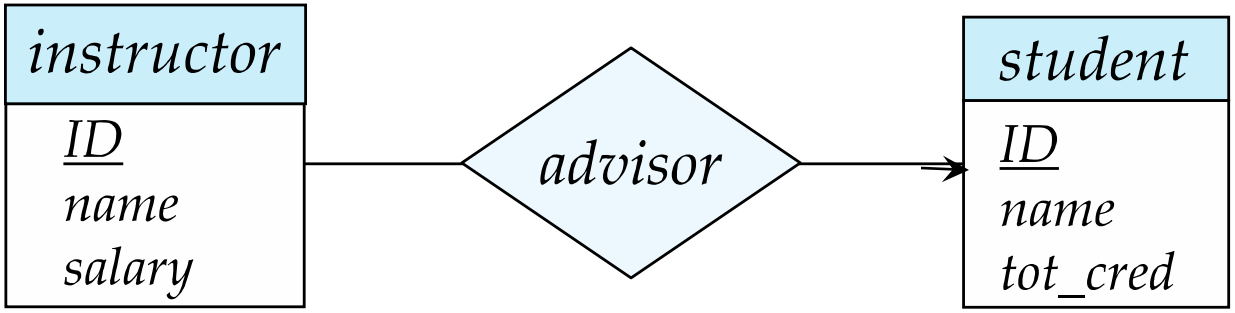
One-to-Many Relationship

- 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,



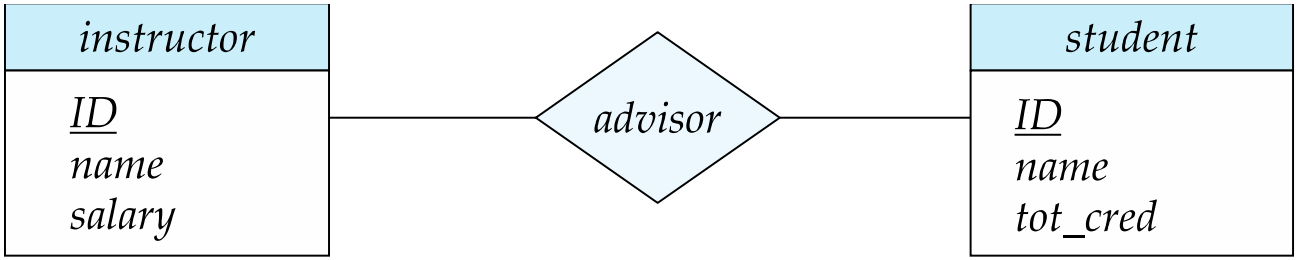
Many-to-One Relationships

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



Many-to-Many Relationship

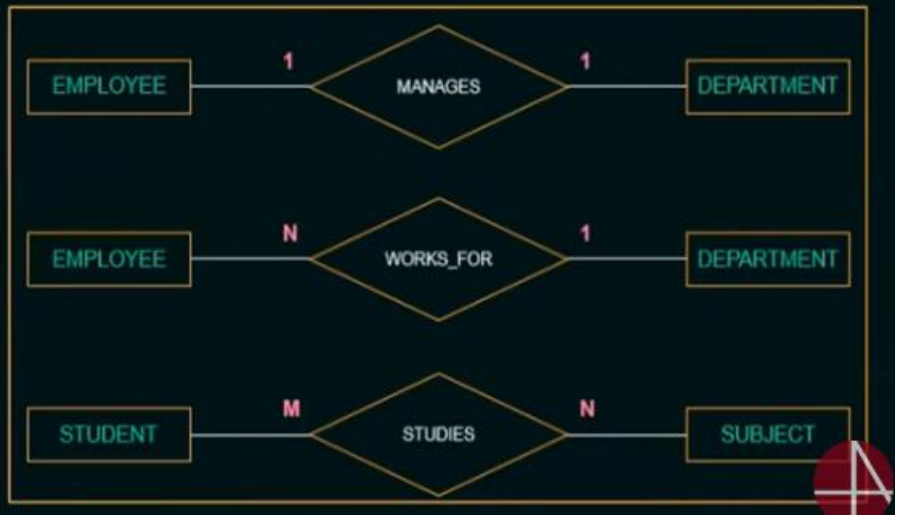
- An instructor is associated with several (possibly 0) students via *advisor*
- A student is associated with several (possibly 0) instructors via *advisor*



Relationships

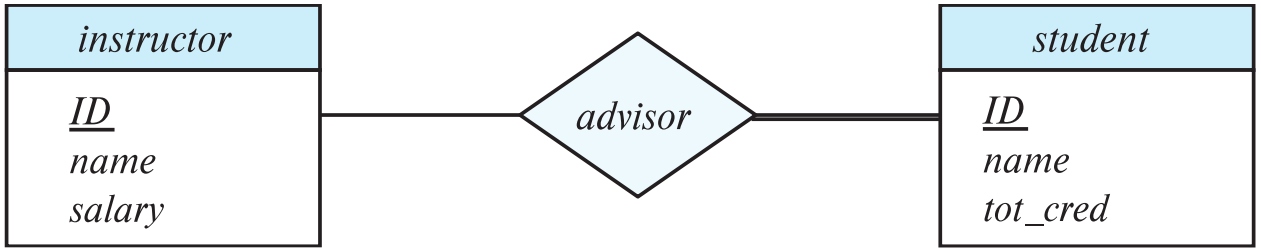
Relationship Constraints:

1. Cardinality Ratio
   - Maximum number of relationship instances that an entity can participate in.
   - Possible cardinality ratios for binary relationship → 1:1, 1:N, N:1, M:N.



Total and Partial Participation

- **Total participation** (indicated by double line): every entity in the entity set participates in at least one relationship in the relationship set



participation of *student* in *advisor* relation is total

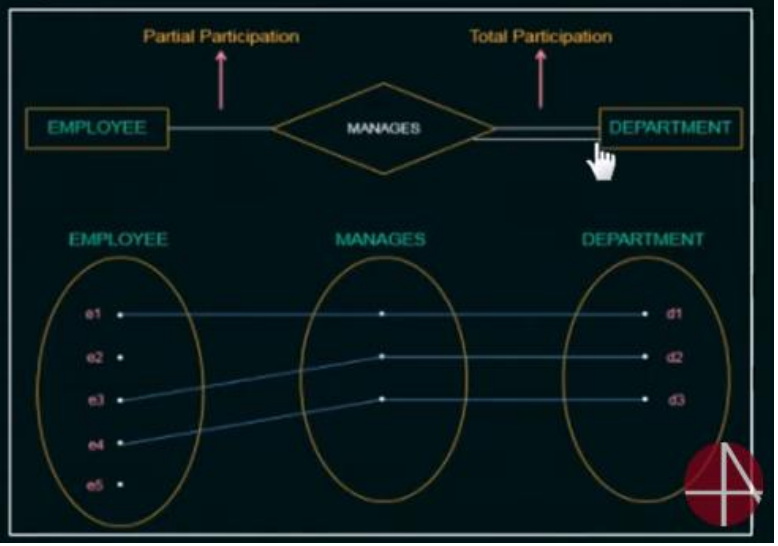
  - every *student* must have an associated instructor

- **Partial participation**: some entities may not participate in any relationship in the relationship set
  - Example: participation of *instructor* in *advisor* is partial

Relationships

Relationship Constraints:

2. Participation Constraints
   - Specifies whether existence of an entity depends on its being related to another entity.
   - 2 types: Total participation & Partial participation.



nesoacademy.org

Total and Partial Participation

Total Participation Example:

Relationship: Employee - Department

Entities:

- **Employee**
- **Department**

Description:

- Total participation for employees in departments signifies that every employee must belong to at least one department.

Representation:

plaintext Copy code

```
Employee ═══► Department
```

In this case, the double line (═══►) indicates that every employee (in the entity set) must be associated with at least one department via the relationship set.

Total and Partial Participation

Partial Participation Example:

Relationship: Manager - Project

Entities:

- **Manager**
- **Project**

Description:

- Partial participation for managers in projects means that not every manager is required to be assigned to a project.

Representation:

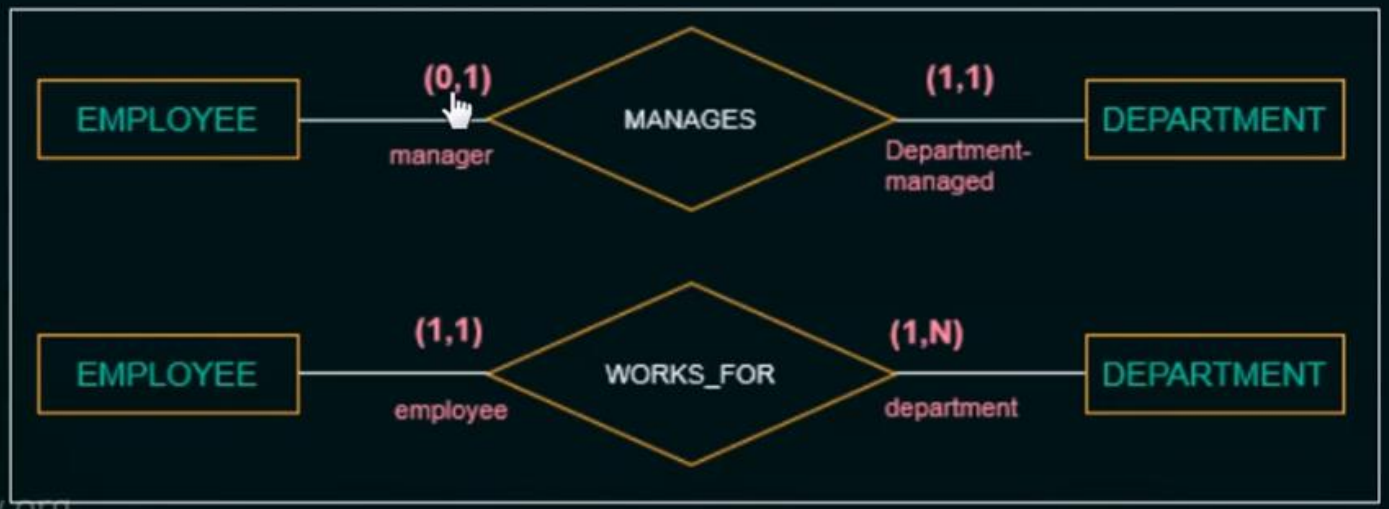
plaintext Copy code

```
Manager ——————► Project
```

Here, the single line (——————►) denotes that some managers may not be associated with any project, indicating partial participation where managers might not be involved in any project in the relationship set.

Alternative Notations for ER Diagrams

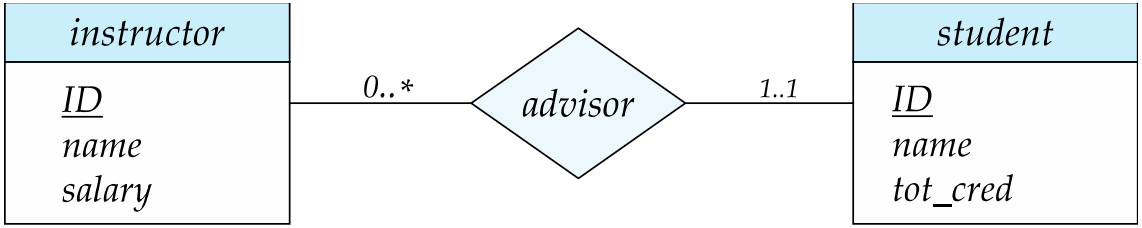
- Associates a pair of integer numbers (min, max) with each participation of an entity type in a relationship type, where $0 \leq min \leq max$ and $max \geq 1$.



nesoacademy.org

Notation for Expressing More Complex Constraints

- A line may have an associated minimum and maximum cardinality, shown in the form *l..h*, where *l* is the minimum and *h* the maximum cardinality
  - A minimum value of 1 indicates total participation.
  - A maximum value of 1 indicates that the entity participates in at most one relationship
  - A maximum value of * indicates no limit.
- Example



  - Instructor can advise 0 or more students. A student must have 1 advisor; cannot have multiple advisors

Cardinality Constraints on Ternary Relationship

- We allow at most one arrow out of a ternary (or greater degree) relationship to indicate a cardinality constraint
- For example, an arrow from *proj_guide* to *instructor* indicates each student has at most one guide for a project
- If there is more than one arrow, there are two ways of defining the meaning.
  - For example, a ternary relationship *R* between *A*, *B* and *C* with arrows to *B* and *C* could mean
    1. Each *A* entity is associated with a unique entity from B and *C* or
    2. Each pair of entities from (*A, B*) is associated with a unique *C* entity, and each pair (*A, C*) is associated with a unique *B*
  - Each alternative has been used in different formalisms
  - To avoid confusion we outlaw more than one arrow

Primary Key

- Primary keys provide a way to specify how entities and relations are distinguished. We will consider:
  - Entity sets
  - Relationship sets.
  - Weak entity sets

Primary key for Entity Sets

- By definition, individual entities are distinct.
- From database perspective, the differences among them must be expressed in terms of their attributes.
- The values of the attribute values of an entity must be such that they can uniquely identify the entity.
  - No two entities in an entity set are allowed to have exactly the same value for all attributes.
- A key for an entity is a set of attributes that suffice to distinguish entities from each other

Primary Key for Relationship Sets

- To distinguish among the various relationships of a relationship set we use the individual primary keys of the entities in the relationship set.
  - Let *R* be a relationship set involving entity sets E1, E2, .. En
  - The primary key for R is consists of the union of the primary keys of entity sets E1, E2, ..En
  - If the relationship set *R* has attributes a1, a2, .., am associated with it, then the primary key of *R* also includes the attributes a1, a2, .., am
- Example: relationship set “advisor”.
  - The primary key consists of *instructor.ID* and *student.ID*
- The choice of the primary key for a relationship set depends on the mapping cardinality of the relationship set.

Choice of Primary key for Binary Relationship

- Many-to-Many relationships. The preceding union of the primary keys is a minimal superkey and is chosen as the primary key.
- One-to-Many relationships . The primary key of the “Many” side is a minimal superkey and is used as the primary key.
- Many-to-one relationships. The primary key of the “Many” side is a minimal superkey and is used as the primary key.
- One-to-one relationships. The primary key of either one of the participating entity sets forms a minimal superkey, and either one can be chosen as the primary key.

Weak Entity Sets

- Consider a *section* entity, which is uniquely identified by a *course_id*, *semester, year*, and *sec_id*.
- Clearly, section entities are related to course entities. Suppose we create a relationship set *sec_course* between entity sets *section* and *course*.
- Note that the information in *sec_course* is redundant, since *section* already has an attribute *course_id*, which identifies the course with which the section is related.
- One option to deal with this redundancy is to get rid of the relationship *sec_course*; however, by doing so the relationship between *section* and *course* becomes implicit in an attribute, which is not desirable.

Weak Entity Sets (Cont.)

- An alternative way to deal with this redundancy is to not store the attribute *course_id* in the *section* entity and to only store the remaining attributes *section_id*, *year*, and *semester*.
  - However, the entity set *section* then does not have enough attributes to identify a particular *section* entity uniquely
- To deal with this problem, we treat the relationship *sec_course* as a special relationship that provides extra information, in this case, the *course_id*, required to identify *section* entities uniquely.
- A **weak entity set** is one whose existence is dependent on another entity, called its **identifying entity**
- Instead of associating a primary key with a weak entity, we use the identifying entity, along with extra attributes called **discriminator** to uniquely identify a weak entity.

Weak Entity Sets (Cont.)

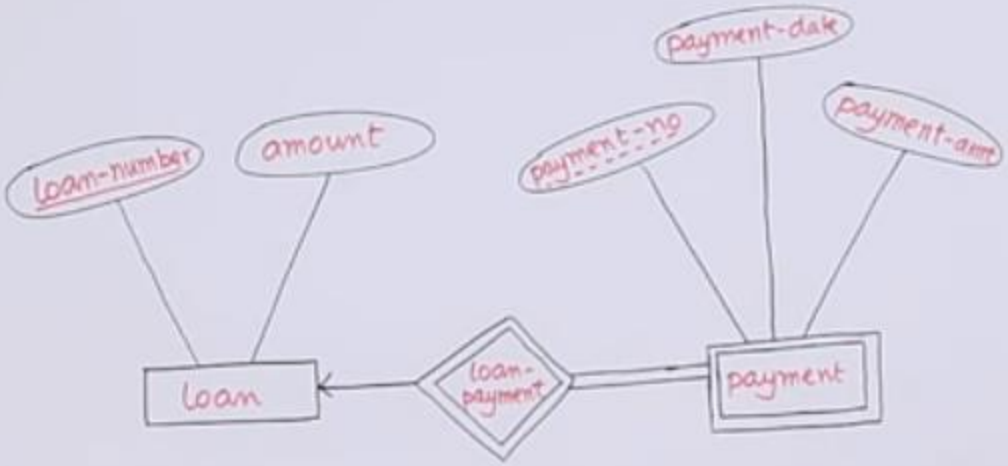
- An entity set that is not a weak entity set is termed a **strong entity set**.
- Every weak entity must be associated with an identifying entity; that is, the weak entity set is said to be **existence dependent** on the identifying entity set.
- The identifying entity set is said to **own** the weak entity set that it identifies.
- The relationship associating the weak entity set with the identifying entity set is called the **identifying relationship**.
- Note that the relational schema we eventually create from the entity set *section* does have the attribute *course_id*, for reasons that will become clear later, even though we have dropped the attribute *course_id* from the entity set *section.*

Weak Entity Sets (Cont.)

loan (loan-number, amount)

loan-payment (loan-number, payment-no, payment-date, payment-amt)

Weak Entity set with Total Participation



L1	1,00,000	20	24/7	5000
L2	2,00,000	20	24/7	5000
L3	1,50,000	20	24/7	5000

Weak Entity Sets (Cont.)

- An **Entity** is a **"thing" or "object"** in the real world that is distinguishable from other objects. In ER Diagram, an **entity** is represented using rectangles.
  - Consider an example of an Organisation- Employee, Manager, Department, Product can be taken as entities in an Organisation.

- Two types of **Entities** in ER Diagram:

1. **Strong entity**
2. **Weak entity**

Weak Entity Sets (Cont.)

Strong Entity (Strong Entity Set)

- The **strong entity** always have a **primary key.**
- Its existence is not dependent on any other entity i.e. it is independent of other entity.
- A set of strong entities is known as ***strong entity set.***
- Strong Entity is represented by a *single rectangle*

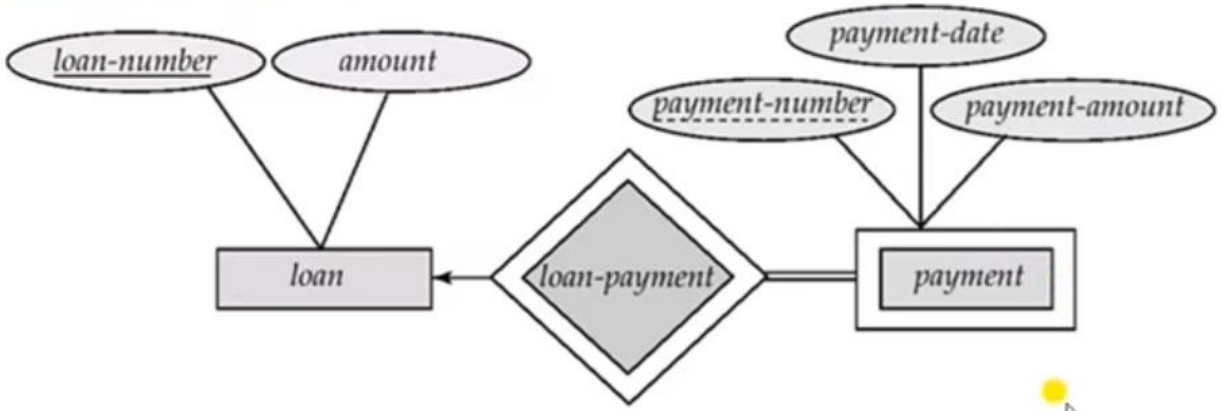
Weak Entity Sets (Cont.)

Weak Entity (Weak Entity Set)

- The **weak entity** does not have a sufficient attributes to form a primary key i.e. Weak entity do not have a primary key.
- A **weak entity** is dependent on a strong entity to ensure the its existence.
- A set of weak entities is known as *weak entity set.*
- Weak Entity is represented by double rectangle:

Weak Entity Sets (Cont.)

Example 1

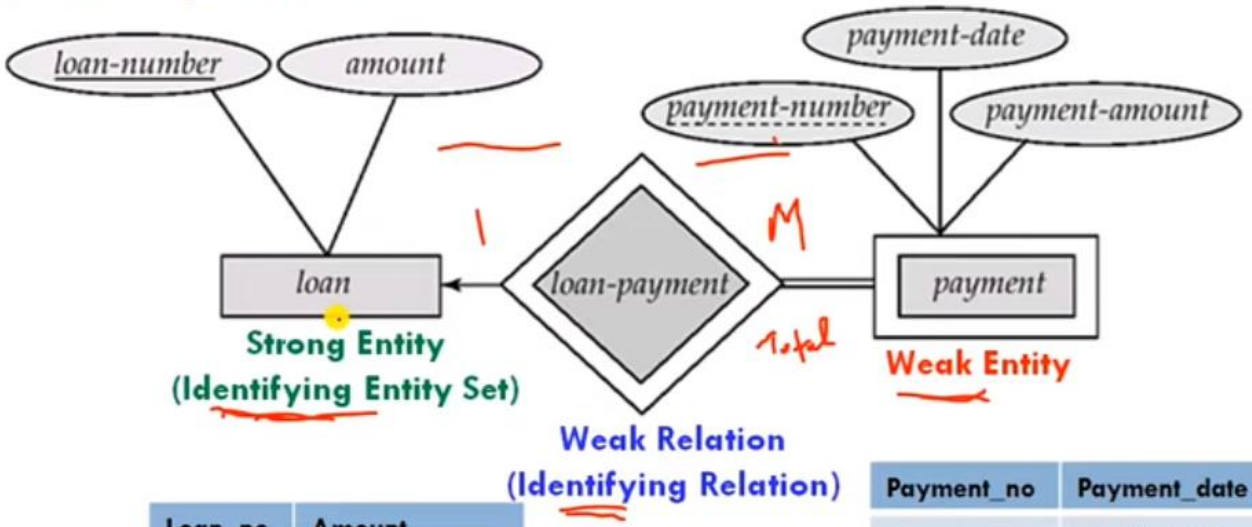


Loan_no	Amount
L1	1,00,000
L2	2,00,000
L3	3,00,000

Payment_no	Payment_date	Payment_amount
1	05-06-2020	5000
1	08-07-2020	10000
1	10-08-2020	15000
2	05-07-2020	5000
2	08-08-2020	10000
2	10-09-2020	15000

Weak Entity Sets (Cont.)

Example 1



Loan_no	Amount
L1	1,00,000
L2	2,00,000
L3	3,00,000

Payment_no	Payment_date	Payment_amount
1	05-06-2020	5000
1	08-07-2020	10000
1	10-08-2020	15000
2	05-07-2020	5000
2	08-08-2020	10000
2	10-09-2020	15000

Weak Entity Sets (Cont.)

Weak Entity Set

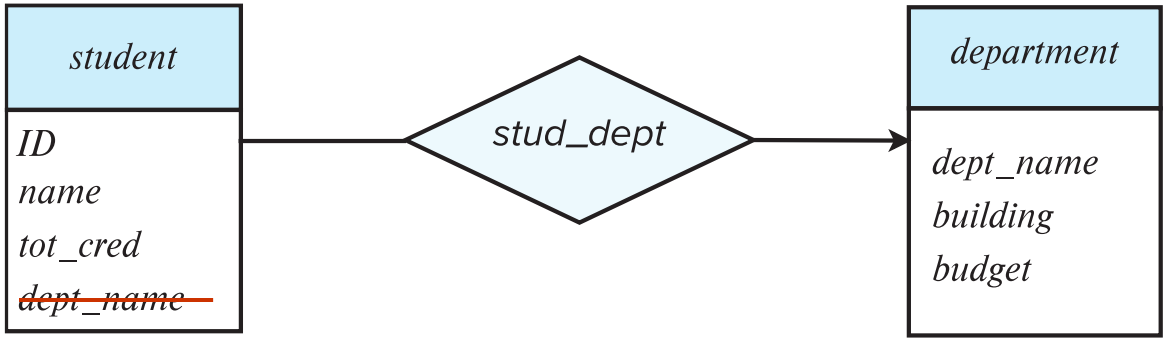
- The **existence of a weak entity set depends** on the existence of a **strong entity set** (called ***identifying entity set***)
- *The **relationship** associating the weak entity set with the strong (or identifying) entity set is called **identifying relationship***
  - Identifying relationship depicted using a **double diamond**
- The **participation of Weak entity set** from the identifying relation set is always **Total**
- The Identifying relationship is **One-to-Many Relationship** from the ***identifying entity set to the weak entity set***
  - ***1-M** relationship from **Loan to Payment***
- The ***discriminator (or partial key)*** of a weak entity set is the set of attributes that distinguishes among all the entities of a weak entity set.
  - The discriminator of a weak entity set us underline with a **dashed line**
- The **primary key of a weak entity set** is formed by the **primary key of the strong entity set** on which the weak entity set is existence dependent, **plus** the **weak entity set's discriminator**.
  - **Primary Key** of Weak Entity **Payment { loan_no, Payment_no }**

Expressing Weak Entity Sets

- In E-R diagrams, a weak entity set is depicted via a double rectangle.
- We underline the discriminator of a weak entity set with a dashed line.
- The relationship set connecting the weak entity set to the identifying strong entity set is depicted by a double diamond.
- Primary key for *section* – (*course_id, sec_id, semester, year*)

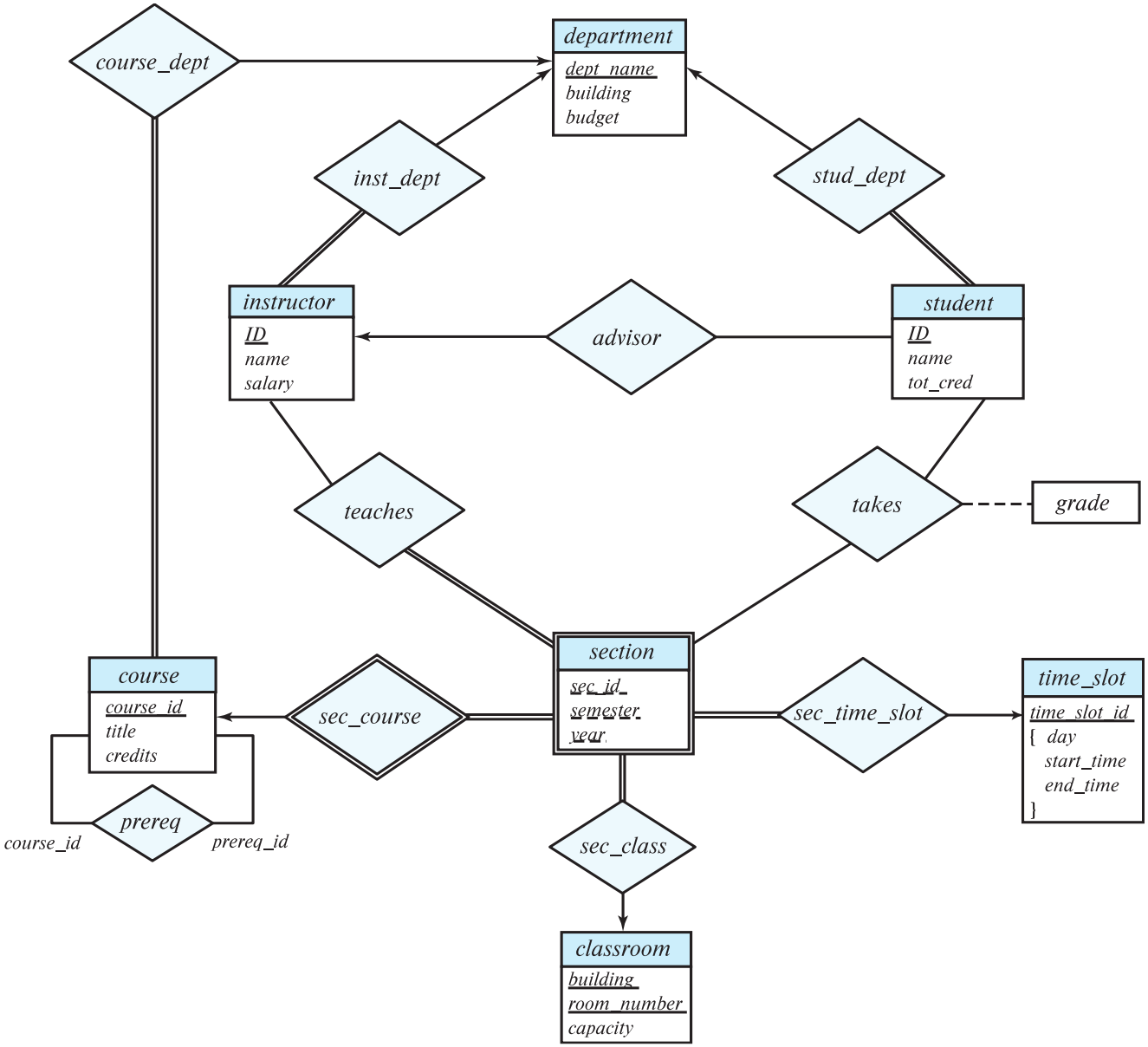
Redundant Attributes

- Suppose we have entity sets:
  - *student*, with attributes: *ID*, *name, tot_cred*, *dept_name*
  - *department,* with attributes: *dept_name, building, budget*
- We model the fact that each student has an associated department using a relationship set *stud_dept*
- The attribute *dept_name* in *student* below replicates information present in the relationship and is therefore redundant
  - and needs to be removed.
- BUT: when converting back to tables, in some cases the attribute gets reintroduced, as we will see later.



(a) Incorrect use of attribute

E-R Diagram for a University Enterprise



Reduction to Relation Schemas

Reduction to Relation Schemas

- Entity sets and relationship sets can be expressed uniformly as *relation schemas* that represent the contents of the database.
- A database which conforms to an E-R diagram can be represented by a collection of schemas.
- For each entity set and relationship set there is a unique schema that is assigned the name of the corresponding entity set or relationship set.
- Each schema has a number of columns (generally corresponding to attributes), which have unique names.

Representing Entity Sets

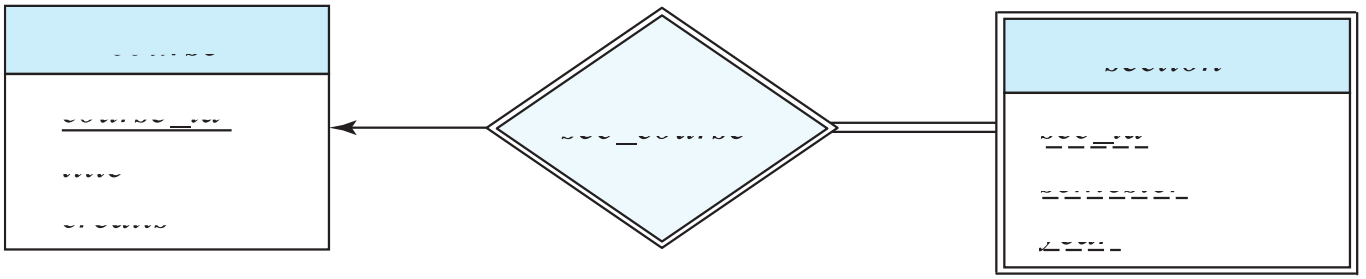
- A strong entity set reduces to a schema with the same attributes

  student(ID, name, tot_cred)

- A weak entity set becomes a table that includes a column for the primary key of the identifying strong entity set

  section (course_id, sec_id, sem, year)

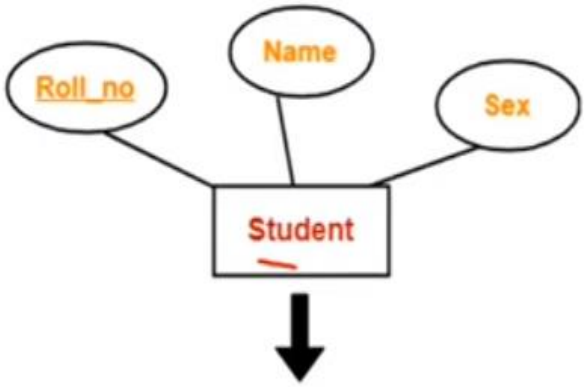
- Example



Representing Entity Sets

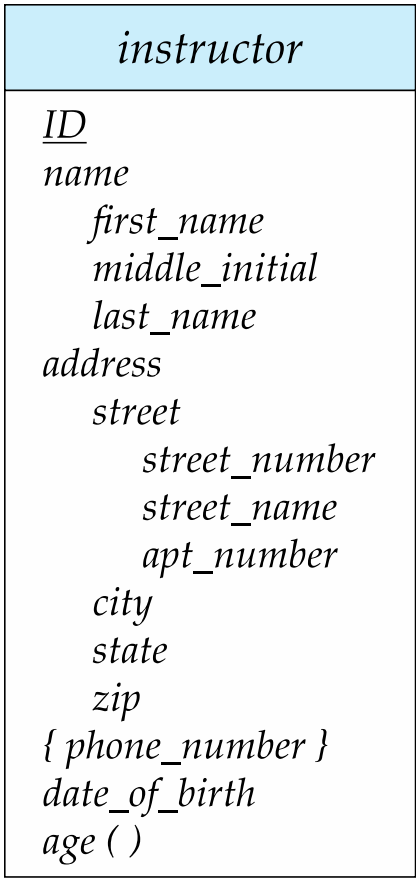
Rule1: Strong Entity Set With Only Simple Attributes

- A strong entity set with only ***simple attributes*** will require only **one table** in relational model.
  - Attributes of the table will be the attributes of the entity set.
  - The primary key of the table will be the key attribute of the entity set.



Roll_no	Name	Sex

Representation of Entity Sets with Composite Attributes

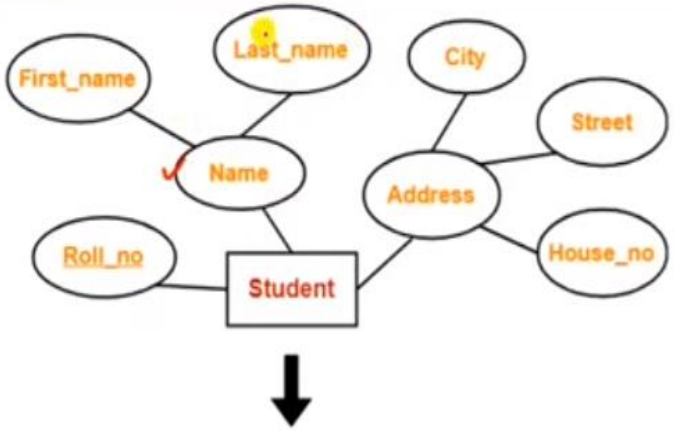


- Composite attributes are flattened out by creating a separate attribute for each component attribute
  - Example: given entity set *instructor* with composite attribute *name* with component attributes *first_name* and *last_name* the schema corresponding to the entity set has two attributes *name_first_name* and *name_last_name*
    - Prefix omitted if there is no ambiguity (*name_first_name* could be *first_name)*
- Ignoring multivalued attributes, extended instructor schema is
  - *instructor(ID, first_name, middle_initial, last_name, street_number, street_name, apt_number, city, state, zip_code, date_of_birth)*

Representing Entity Sets

Rule 2: Strong Entity Set With Composite Attributes

- A strong entity set with any number of ***composite attributes*** will require **only one table** in relational model.
- While conversion, simple attributes of the composite attributes are taken into account and not the composite attribute itself.



Roll_no	First_name	Last_name	House_no	Street	City

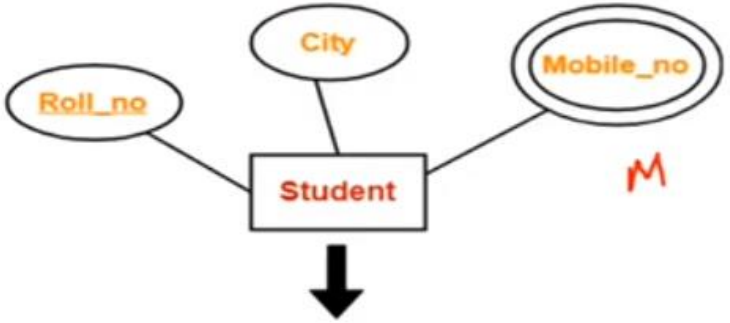
Representation of Entity Sets with Multivalued Attributes

- A multivalued attribute *M* of an entity *E* is represented by a separate schema *EM*
- Schema *EM* has attributes corresponding to the primary key of *E* and an attribute corresponding to multivalued attribute *M*
- Example: Multivalued attribute *phone_number* of *instructor* is represented by a schema:
  inst_phone= (*ID*, *phone_number*)
- Each value of the multivalued attribute maps to a separate tuple of the relation on schema *EM*
  - For example, an *instructor* entity with primary key 22222 and phone numbers 456-7890 and 123-4567 maps to two tuples:
    (22222, 456-7890) and (22222, 123-4567)

Representing Entity Sets

Rule 3: Strong Entity Set With Multi Valued Attributes

- A strong entity set with any number of ***multi-valued attributes*** will require **two tables** in relational model.
  - One table will contain all the simple attributes with the primary key.
  - Other table will contain the primary key and all the multi valued attributes.

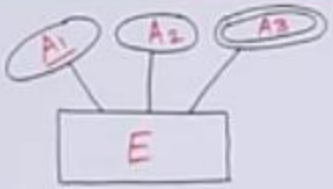


Roll_no	City

Roll_no	Mobile_no

Representing Entity Sets

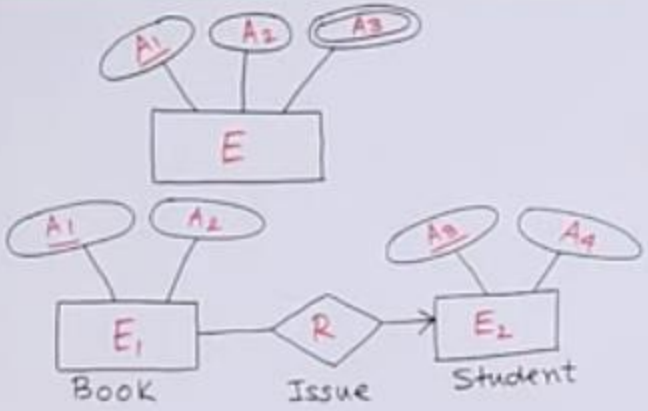
Forming schemas from ER-Diagram



Resultant schemas

$E_1(\underline{A_1}, A_2), E_2(\underline{A_1}, A_3)$

Forming schemas from ER-Diagram



Resultant schemas

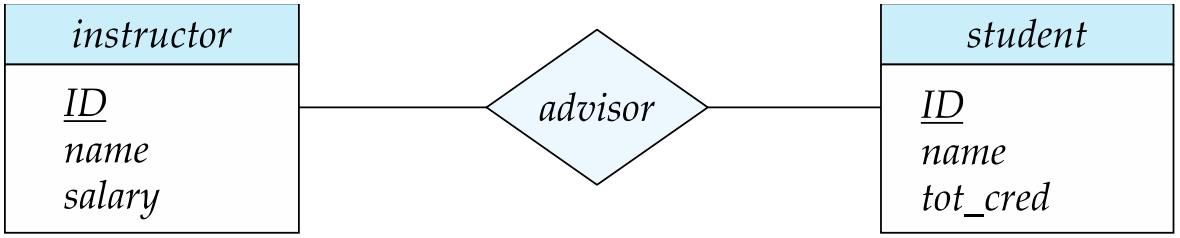
$E_1(\underline{A_1}, A_2), E_2(\underline{A_1}, A_3)$

Existence dependency	Cardinality E1 : E2	Resultant schemas
X	N : 1	$E_1(\underline{A_1}, A_2), E_2(\underline{A_3}, A_4), R(\underline{A_1}, \underline{A_3})$

Representing Relationship Sets

- A many-to-many relationship set is represented as a schema with attributes for the primary keys of the two participating entity sets, and any descriptive attributes of the relationship set.
- Example: schema for relationship set *advisor*

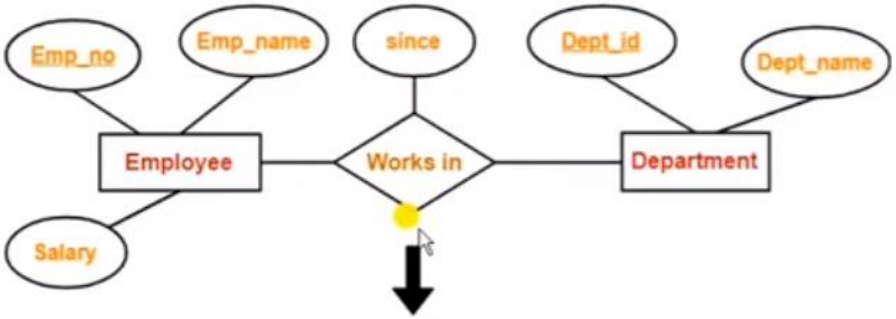
advisor = (*s_id, i_id*)



Representing Relationship Sets

Rule 4: Translating Relationship Set into a Table

- A ***relationship set*** will require **one table** in the relational model.
- Attributes of the table are:
  - Primary key attributes of the participating entity sets
  - Its own descriptive attributes if any.
- Set of non-descriptive attributes will be the primary key
- For given ER diagram, three tables will be required in relational model-
  - One table for the entity set "Employee"
  - One table for the entity set "Department"
  - One table for the relationship set "Works in"



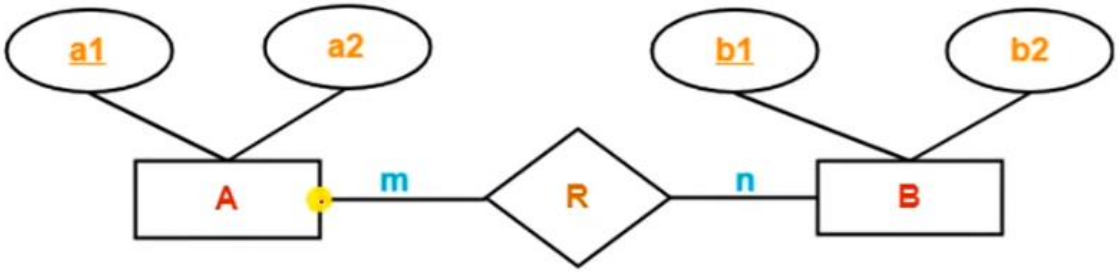
Emp_no	Dept_id	since

Schema : Works in (Emp_no , Dept_id , since)

Rule 5: For Binary Relationships With Cardinality Ratios

- The following four cases are possible-
  - **Case-1:** Binary relationship with cardinality ratio m:n
  - **Case-2:** Binary relationship with cardinality ratio 1:n
  - **Case-3:** Binary relationship with cardinality ratio m:1
  - **Case-4:** Binary relationship with cardinality ratio 1:1

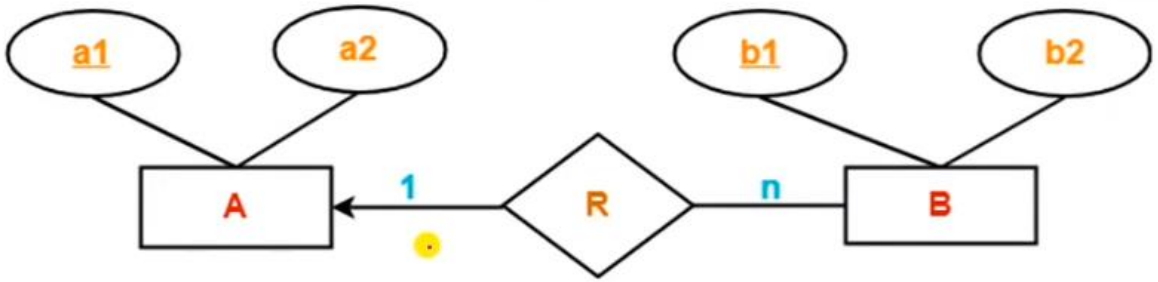
Case-1: For Binary Relationship with Cardinality Ratio m:n



In ***Many-to-Many relationship***, **three tables** will be required-

1. **A (a1 , a2)**
2. **R (a1 , b1)**
3. **B (b1 , b2)**

Case-2: For Binary Relationship with Cardinality Ratio 1:n

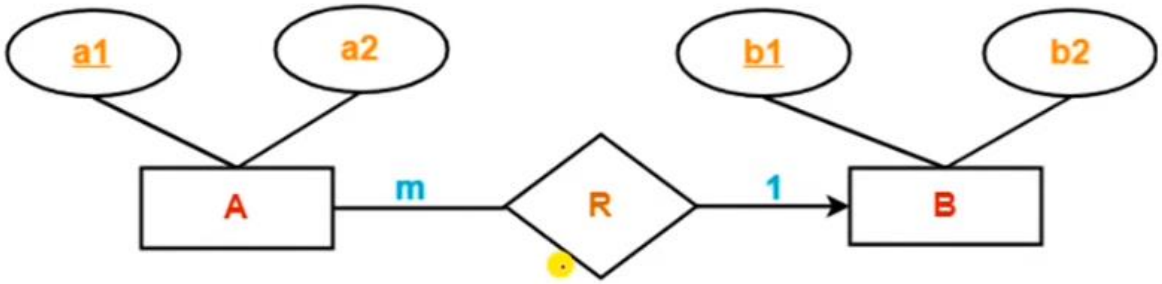


In ***One-to-Many relationship***, **two tables** will be required-

1. **A (a1 , a2)**
2. **BR (b1 , b2, a1)**

NOTE - Here, combined table will be drawn for the entity set B and relationship set R.

Case-3: For Binary Relationship with Cardinality Ratio m:1

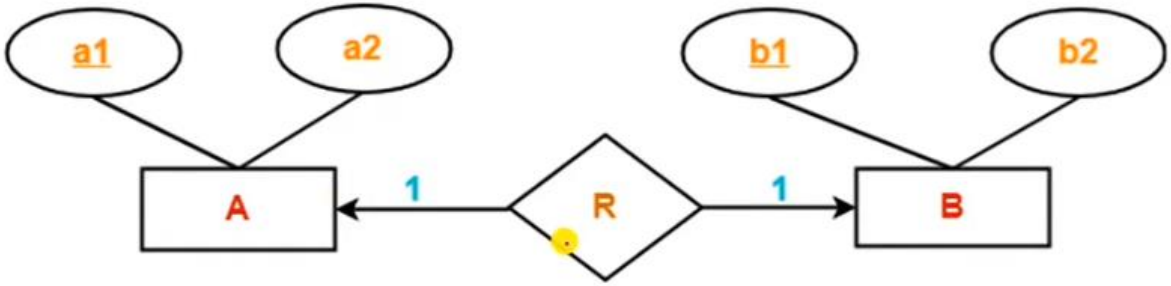


In ***Many-to-One relationship***, **two tables** will be required-

1. **AR (a1 , a2 , b1)**
2. **B (b1 , b2)**

NOTE - Here, combined table will be drawn for the entity set A and relationship set R.

Case-4: For Binary Relationship with Cardinality Ratio 1:1



In ***One-to-One relationship***, **two tables** will be required. Either combine 'R' with 'A' or 'B'

Way-01:

1. AR (a1 , a2 , b1)
2. B (b1 , b2)

Way-02:

1. A (a1 , a2)
2. BR (b1 , b2, a1)

Thumb Rules to Remember:

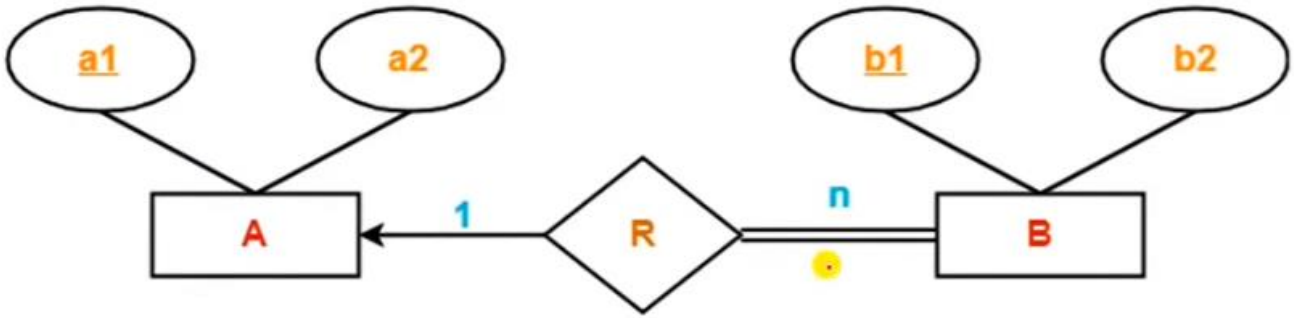
(For determining minimum number of tables)

- **While determining the minimum number of tables required for binary relationships with given cardinality ratios**, following thumb rules must be kept in mind-
  - For binary relationship with cardinality ration **m : n** ,
    - *Separate and individual tables* will be drawn for each entity set and relationship (i.e. **three tables** will be required).
  - For binary relationship with cardinality ratio either **m : 1 or 1 : n** ,
    - Always remember "many side will consume the relationship" i.e. a combined table will be drawn for many side entity set and relationship set (i.e. **Two tables** will be required).
  - For binary relationship with cardinality ratio **1 : 1** ,
    - **Two tables** will be required. You can combine the relationship set with any one of the entity sets.

Rule-6: For Binary Relationship with Both Cardinality Constraints and Participation Constraints

- Cardinality constraints will be implemented as discussed in Rule-5.
- Because of the *total participation constraint*, foreign key acquires **NOT NULL** constraint i.e. now foreign key can not be null.
- Two Cases:
  - **Case-1:** For Binary Relationship with Cardinality Constraint and Total Participation Constraint from One Side
  - **Case-2:** For Binary Relationship with Cardinality Constraint and Total Participation Constraint from Both Sides

Case-1: For Binary Relationship with Cardinality Constraint and Total Participation Constraint from One Side

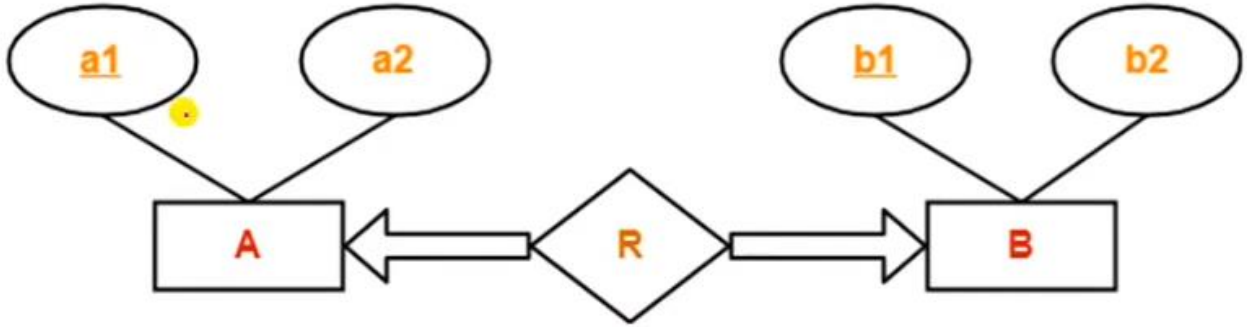


- Because cardinality ratio = 1 : n , we will combine the entity set B and relationship set R.
- Then, **two tables** will be required-
  1. **A (a1 , a2)**
  2. **BR (b1 , b2 , a1)**

Because of total participation, ***foreign key a1*** has acquired ***NOT NULL constraint***, so it can't be null now.

Case-2: For Binary Relationship with Cardinality Constraint and Total Participation Constraint from Both Sides

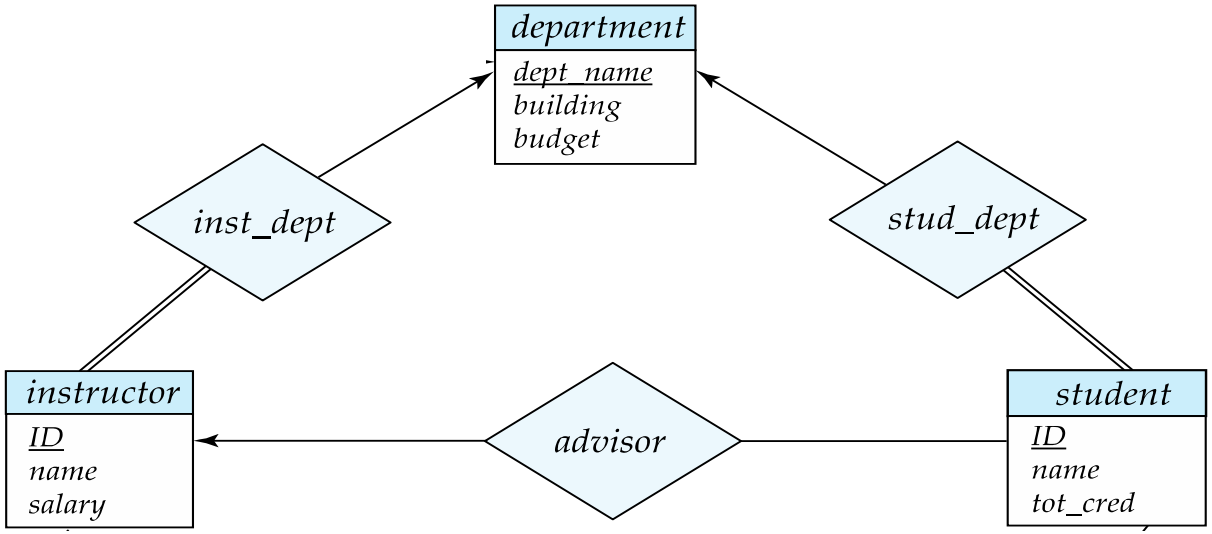
- If there is a key constraint from both the sides of an entity set with total participation, then that binary relationship is represented using **only single table.**



- Here, Only **one table** is required.
  1. **ARB (a1 , a2 , b1 , b2)**

Redundancy of Schemas

- 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
- Example: Instead of creating a schema for relationship set *inst_dept*, add an attribute *dept_name* to the schema arising from entity set *instructor*
- Example

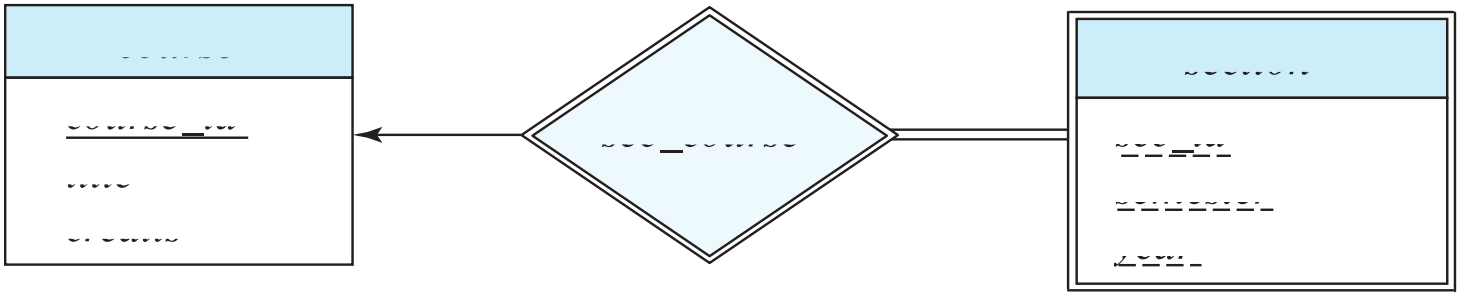


Redundancy of Schemas (Cont.)

- For one-to-one relationship sets, either side can be chosen to act as the "many" side
  - That is, an extra attribute can be added to either of the tables corresponding to the two entity sets
- If participation is *partial* on the "many" side, replacing a schema by an extra attribute in the schema corresponding to the "many" side could result in null values

Redundancy of Schemas (Cont.)

- The schema corresponding to a relationship set linking a weak entity set to its identifying strong entity set is redundant.
- Example: The *section* schema already contains the attributes that would appear in the *sec_course* schema

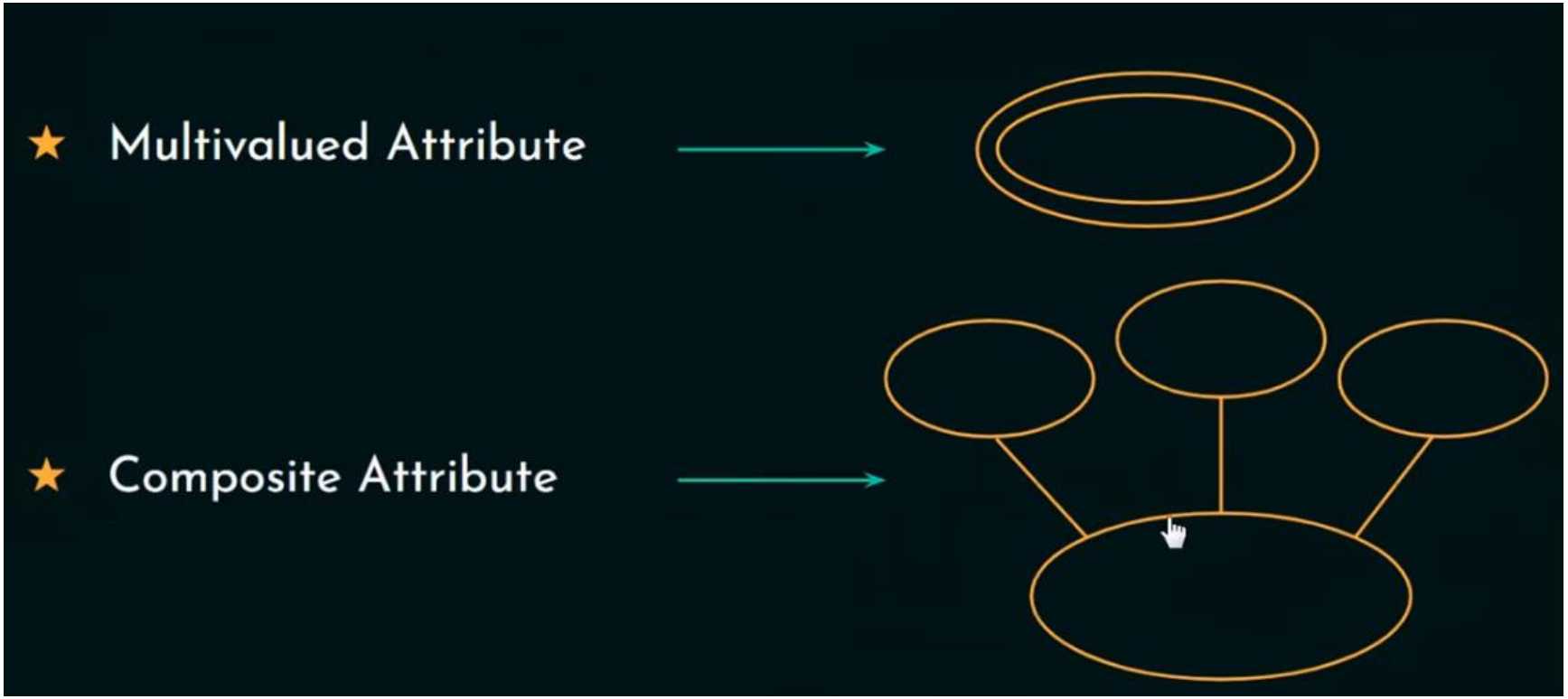


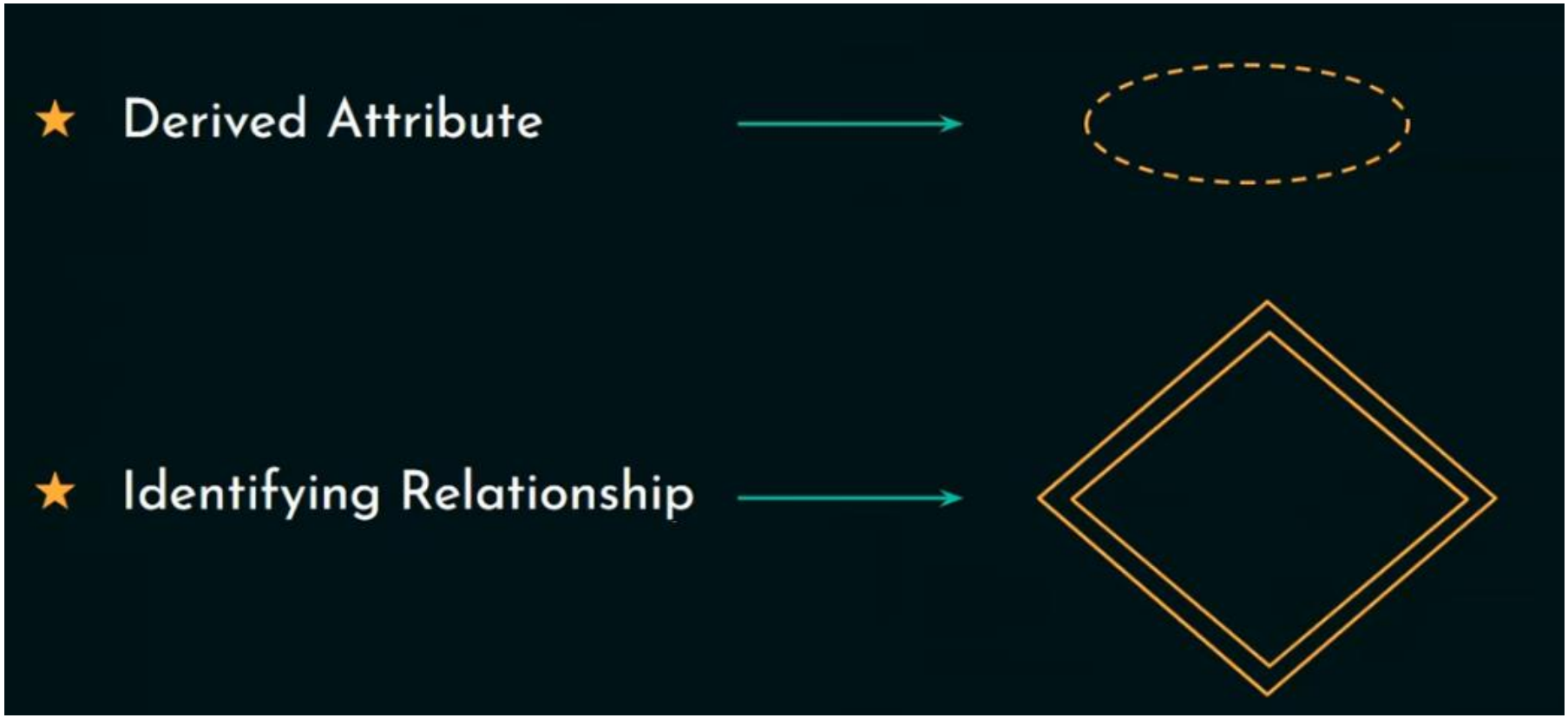
Redundancy of Schemas (Cont.)

Redundancy of Schemas (Cont.)

- The schema corresponding to a relationship set linking a weak entity set to its identifying strong entity set is redundant.
- Example: The *section* schema already contains the attributes that would appear in the *sec_course* schema
  - Thus, we can eliminate *sec_course*


Entity
Weak Entity
Attribute
Key Attribute


Multivalued Attribute
Composite Attribute


Derived Attribute
Identifying Relationship

❖ Sample Database Application

★ Let us see an example database application, called COMPANY.

★ Requirements gathered:

- Company is organized into departments. Each department has a unique name, unique number & a particular employee who manages the department. We also keep track of the start date of the manager. A department may have several locations.

★ Requirements gathered:

- A department controls number of projects, each of which has a unique name, unique number and a single location.
- Employee details → name, SSN, sex, salary. We keep track of number of hours per week on each project.
- Keep track of each employee's dependents (first name, sex, relationship to the employee).

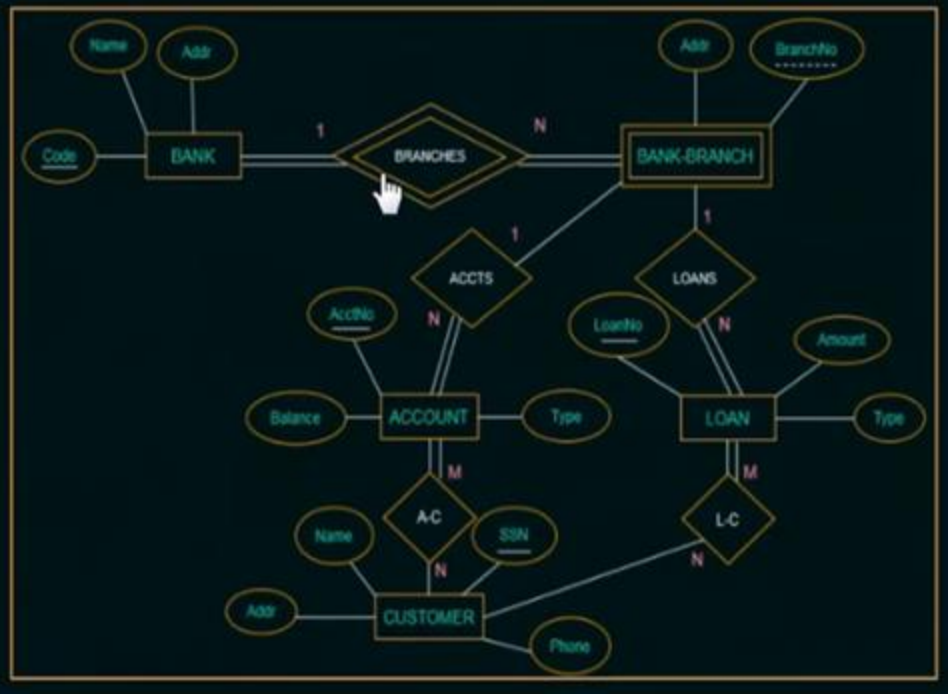
Exercises

1. Consider the ER diagram shown in the figure for part of a BANK database. Each bank can have multiple branches, and each branch can have multiple accounts and loans.

a. List the (nonweak) entity types in the ER diagram.

Exercises

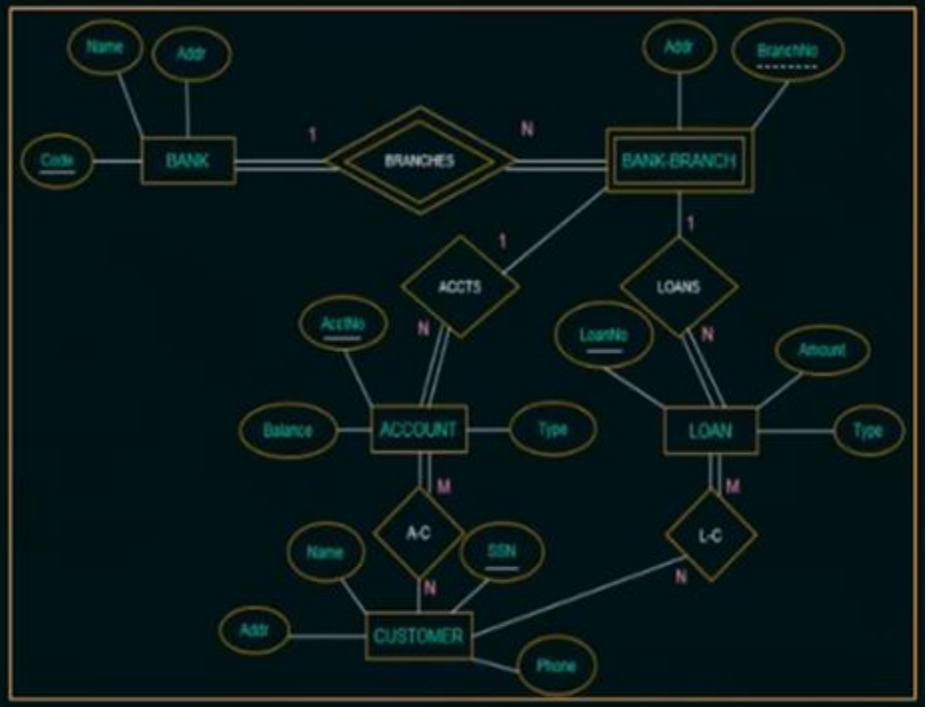
b. Is there a weak entity type? If so, give its name, partial key, and identifying relationship.



c. What constraints do the partial key and the identifying relationship of the weak entity type specify in this diagram?

d. List the names of all relationship types, and specify the (min, max) constraint on each participation of an entity type in a relationship type. Justify your choices.

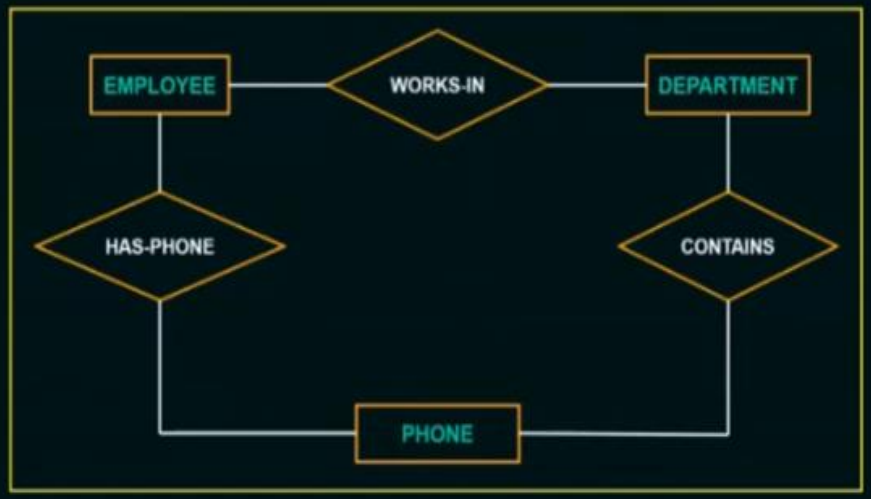
e. List concisely the user requirements that led to this ER schema design.



f. Suppose that every customer must have at least one account but is restricted to at most two loans at a time, and that a bank branch cannot have more than 1000 loans. How does this show up on the (min, max) constraints?

Exercises

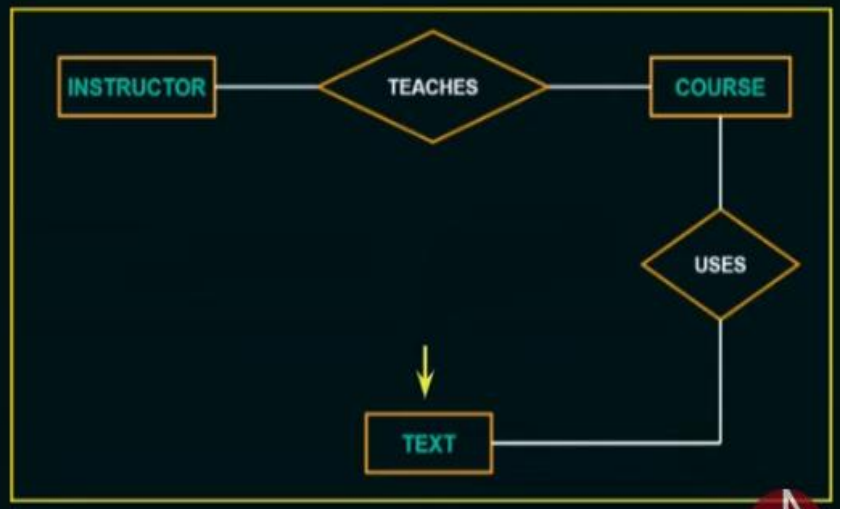
1. Consider the ER diagram in the figure. Assume that an employee may work in upto two departments or may not be assigned to any department. Assume that each department must have one and may have up to three phone numbers. Supply (min, max) constraints on this diagram. State clearly any additional assumptions you make. Under what conditions would the relationship HAS_PHONE be redundant in this example?



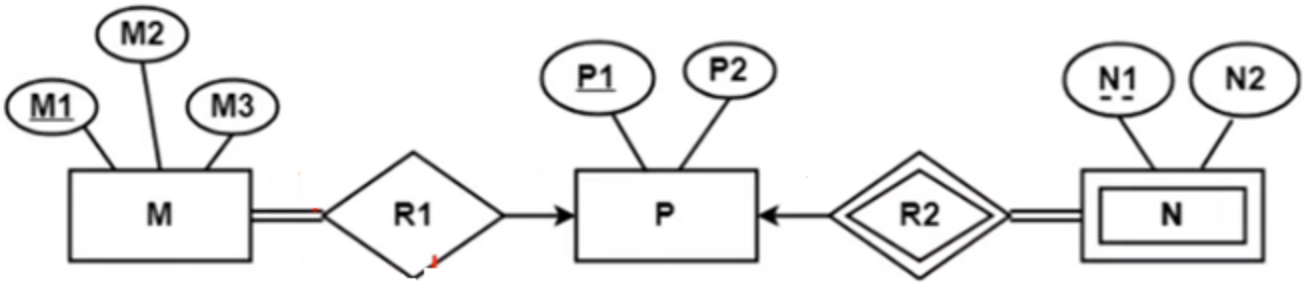
Exercises

2. Consider the ER diagram shown in the figure. Assume that a course may or may not use a textbook, but that a text by definition is a book that is used in some course. A course may not use more than five books. Instructors teach from two to four courses.

Supply (min, max) constraints on this diagram. State clearly any additional assumptions you make. If we add the relationship ADOPTS between INSTRUCTOR and TEXT, what (min, max) constraints would you put on it? Why?



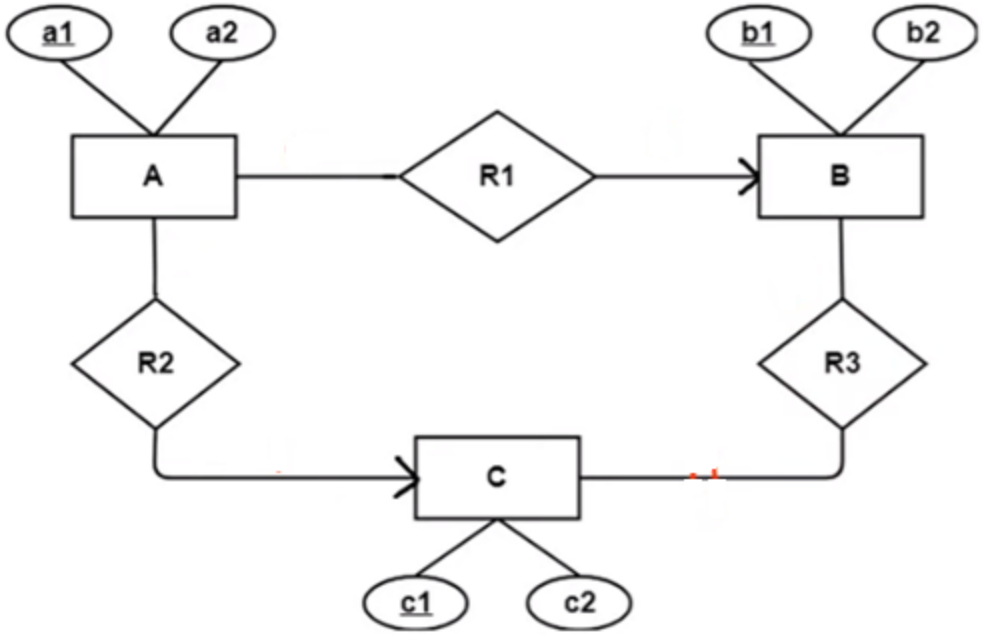
- Find the minimum number of tables required for the following ER diagram in relational model



Solution-

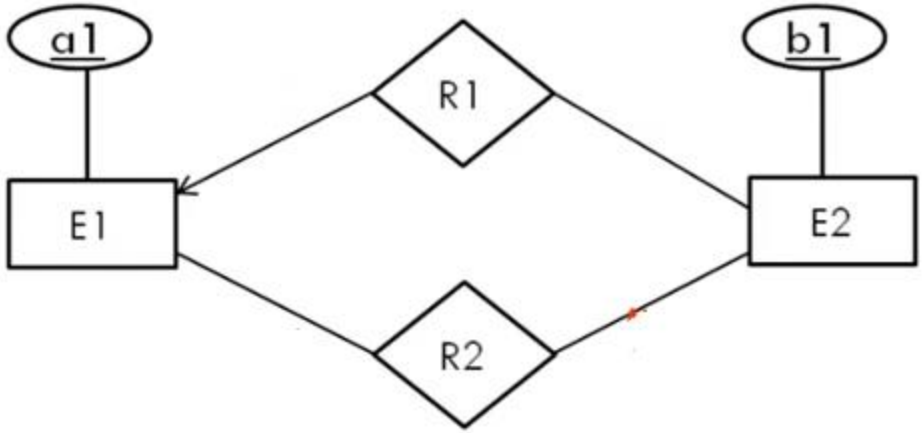
- Find the minimum number of tables required for the following ER diagram in relational model

Solution-



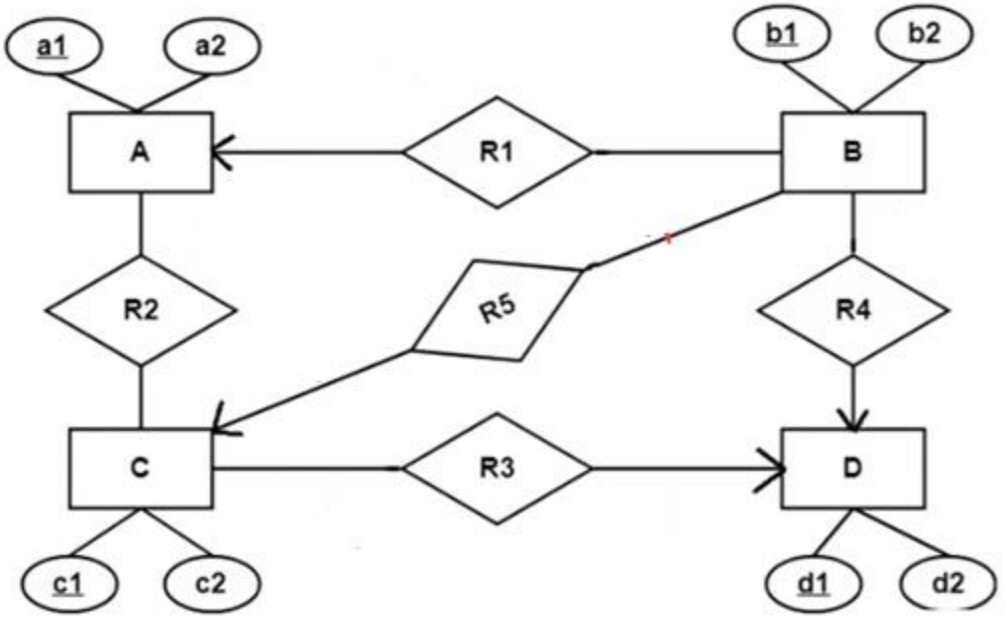
- Find the minimum number of tables required for the following ER diagram in relational model

- **Solution**:

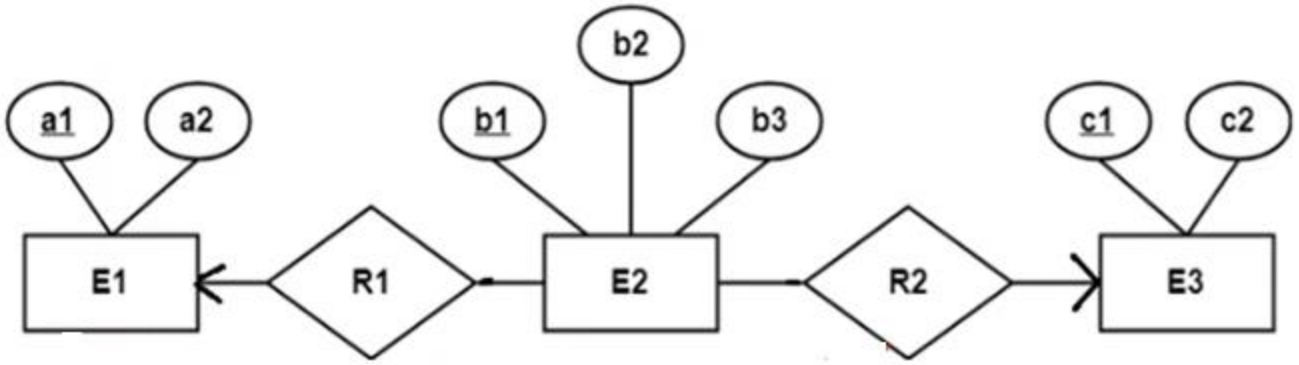


- Find the minimum number of tables required for the following ER diagram in relational model

Solution-

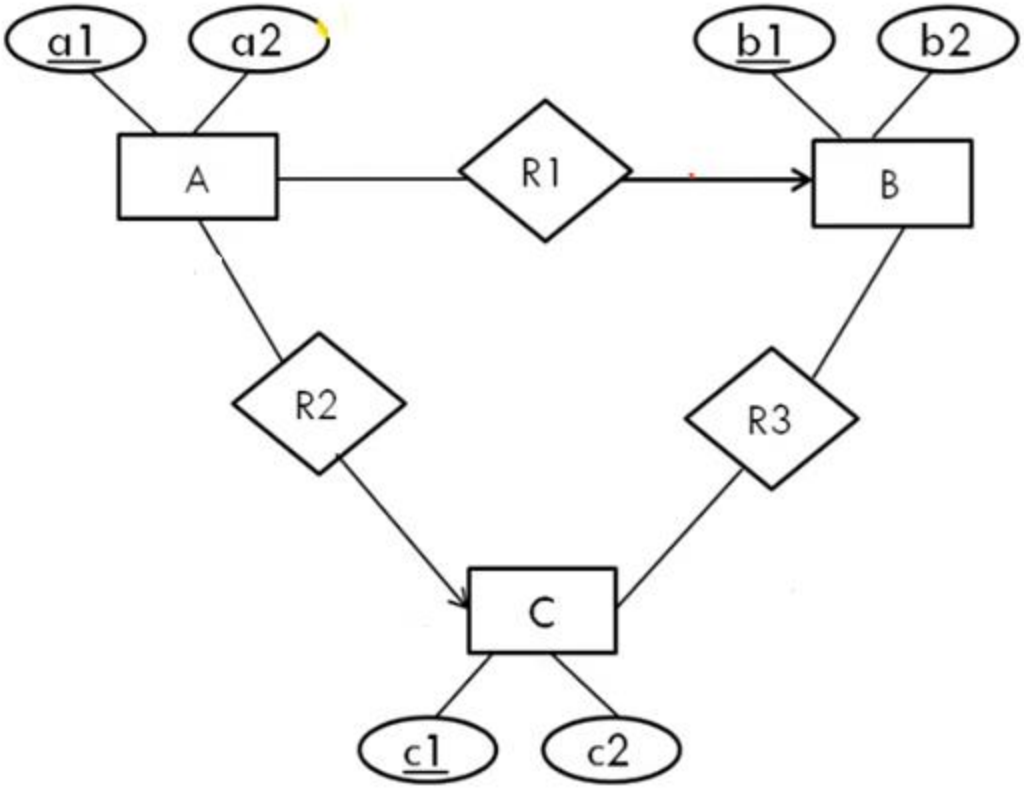


- Find the minimum number of tables required for the following ER diagram in relational model



Solution-

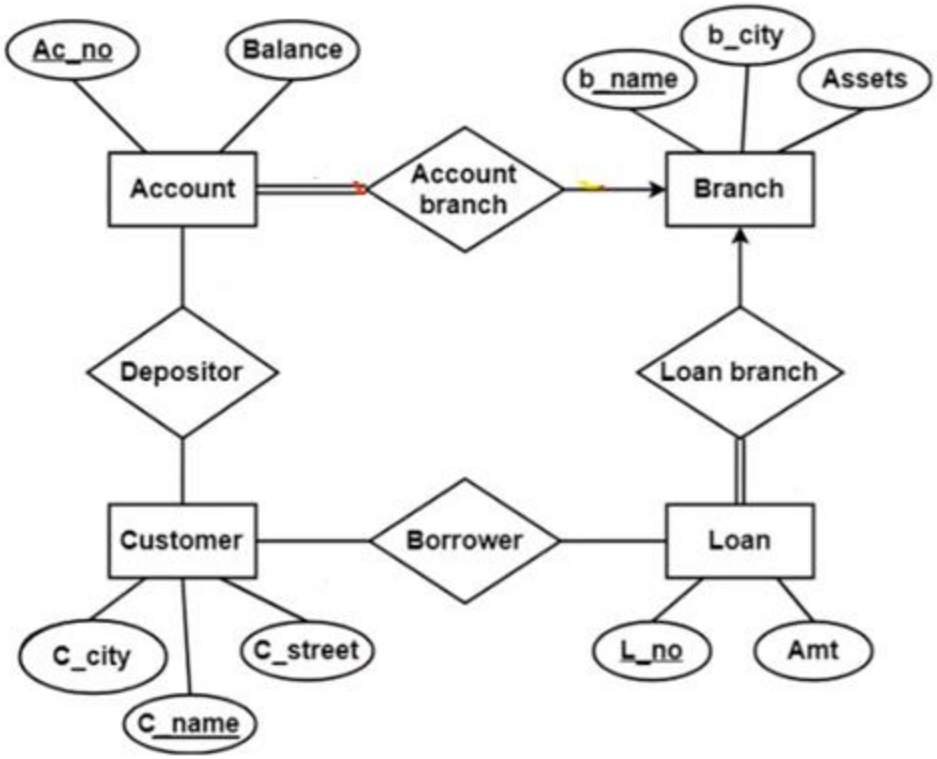
- Find the minimum number of tables required for the following ER diagram in relational model



Solution-

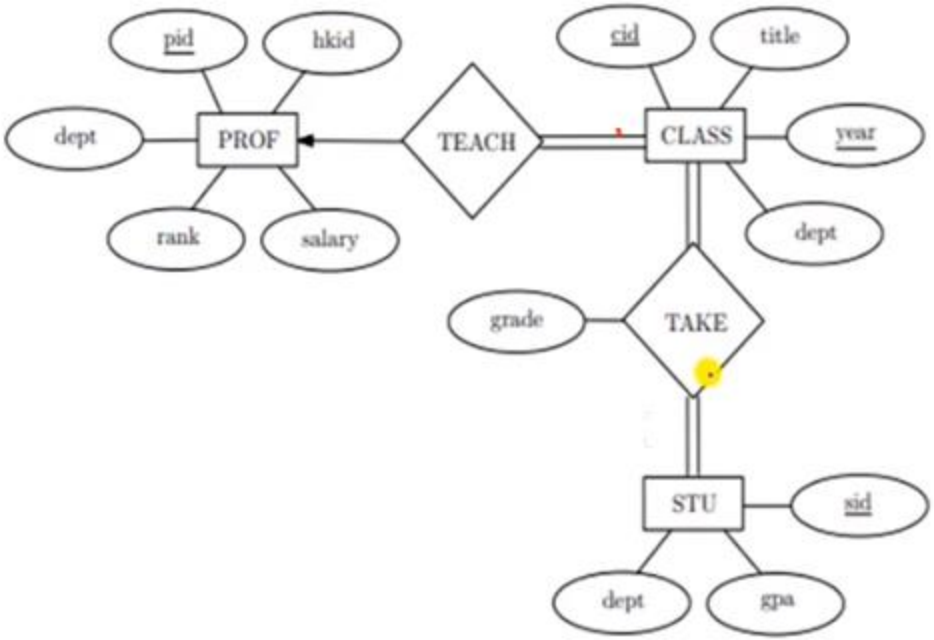
- Find the minimum number of tables required for the following ER diagram in relational model

Solution-

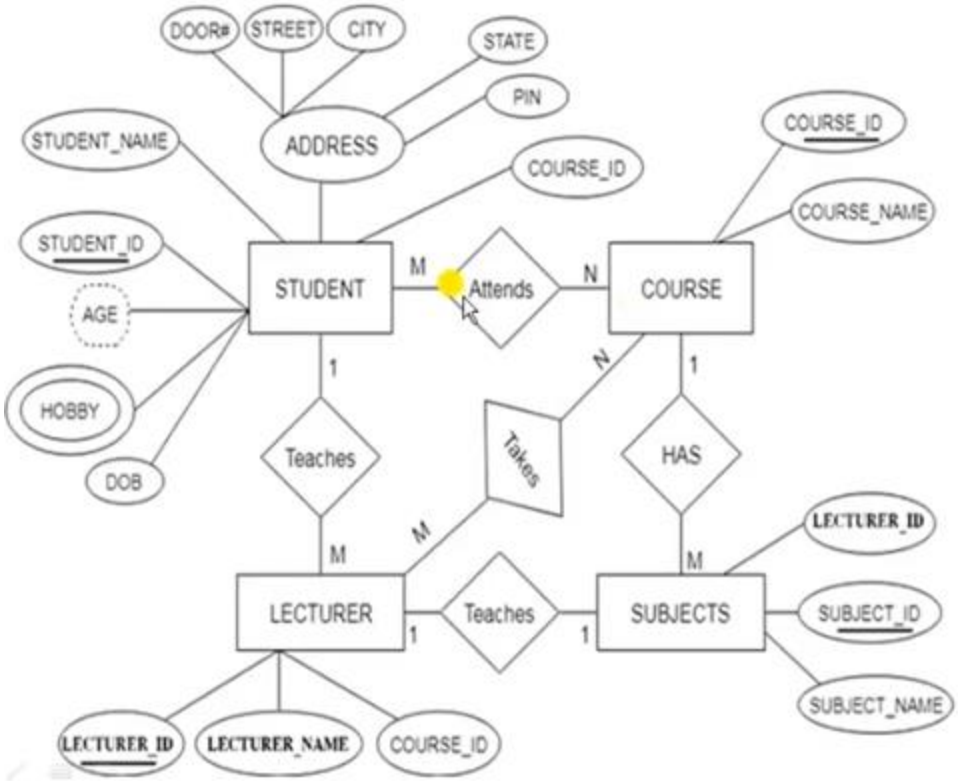


- Find the minimum number of tables required for the following ER diagram in relational model

Solution-



Solution:


DOOR#
STREET
CITY
STATE
PIN
ADDRESS
STUDENT_NAME
COURSE_ID
COURSE_ID
COURSE_NAME
STUDENT_ID
STUDENT
M
Attends
N
COURSE
AGE
HOBBY
DOB
1
N
1
Teaches
Takes
HAS
M
M
M
LECTURER_ID
LECTURER
Teaches
SUBJECTS
SUBJECT_ID
1
1
SUBJECT_NAME
LECTURER_ID
LECTURER_NAME
COURSE_ID