SYLLABUS

Database Design and ER Diagrams: Attributes and Entity Sets, Relationships and Relationship Sets, Constraints, Keys, Design Issues, Entity-Relationship Diagram, Extended E-R Features, Database Design with ER model, Database Design for a schema.

Database Design and ER Diagrams

Introduction

- Database design is a collection of tasks or processes that enhance the designing, development, implementation, and maintenance of enterprise data management system.
- The main objectives of database designing are to produce physical and logical design models of the proposed database system.
- ER model stands for an Entity-Relationship model. It is a high-level data model used to define the data elements and relationship for a specified system.
- It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.
- In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.
- an E-R diagram can express the overall logical structure of a database graphically

Attributes and Entity Sets

Entity:

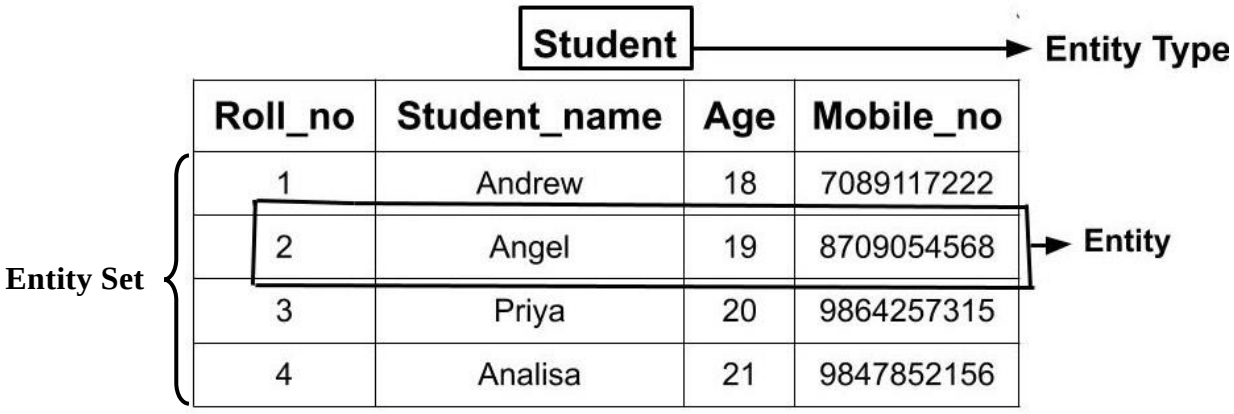
- An entity can be a real-world object that is distinguishable from all other objects.
- If we cannot distinguish it from others then it is an object but not entity.
- For example, human, car, mobile, bank account etc.
- Entities are two types: **tangible** and **intangible**.
- In the ER diagram, we represent the **entity** by a **rectangle shape**.

Entity Type:

- The entity type is a collection of the entities having similar attributes.
- In the ER diagram, we represent the **entity type** by its **name**.

Entity Set:

- An entity set is a collection of similar types of entities that share the same properties, or attributes.
- Entity sets need not be disjoint.
- For example, a Students set may contain all the students of a college.
- In the ER diagram, we represent the **entity set** by a **rectangle shape**.



Strong Entity:

- A strong entity is an entity that can be uniquely identified by its attributes alone.
- A strong entity is not dependent on any other entity. It has a primary key.
- In the ER diagram, we represent the strong **entity** by a **rectangle shape**.

Weak Entity:

- A weak entity is an entity that cannot be uniquely identified by its attributes alone.
- It must depend on strong entity. The foreign key is typically a primary key of an entity it is related to.
- In the ER diagram, we represent the **weak entity** by a **double line rectangle shape**.

Attributes:

- The properties of an entity are called attributes.

(Or)

- Entities are represented by means of their properties, called **attributes**.
- Each entity has a value for each of its attributes.
- In the ER diagram, we represent the **attribute** by an **oval shape**.
- For example,
  - ✓ A **human** entity having **attributes** like name, gender, age, color, height etc
  - ✓ A **mobile** entity having **attributes** like brand, model, prize, color, os etc.
- **Types of attributes are**
  - ✓ **Simple vs composite attributes**
  - ✓ **Single vs multi valued attributes**
  - ✓ **Derived vs complex attributes**
  - ✓ **Key attributes**
  - ➢ **Simple Attribute:**
    - When an attribute cannot be divided further, then it is called a simple attribute.
    - It is also known as **atomic attributes**.
    - Example**:** The roll number of a student, age of person etc.
  - ➢ **Composite Attribute:**
    - When any attribute can be divided further into more sub-attributes, then that attribute is called a composite attribute.
    - Composite attributes are those that are made up of the composition of more than one attribute.
    - Example**:** The address can be further split into house number, street number, city, state, country, and pin code. The name can also be split into first name middle name, and last name.
  - ➢ **Single-valued Attribute:**
    - Those attributes which can have exactly one value are known as single valued attributes.
    - They contain singular values, so more than one value is not allowed.
    - For example, the DOB of a student can be a single valued attribute. Another example is gender because one person can have only one gender.
  - ➢ **Multi-valued Attribute:**
    - Those attributes which can have more than one entry or which contain more than one value are called multi valued attributes.
    - For example, one person can have more than one phone number, so that it would be a multi valued attribute. Another example is the hobbies of a person because one can have more than one hobby.
    - In the Entity Relationship (ER) diagram, we represent the multi valued attribute by double oval representation.
  - ➢ **Derived Attribute:**
    - Derived attributes are also called stored attributes. When one attribute can be derived from the other attribute, then it is called a derived attribute. We can do some calculations on normal attributes and create derived attributes.

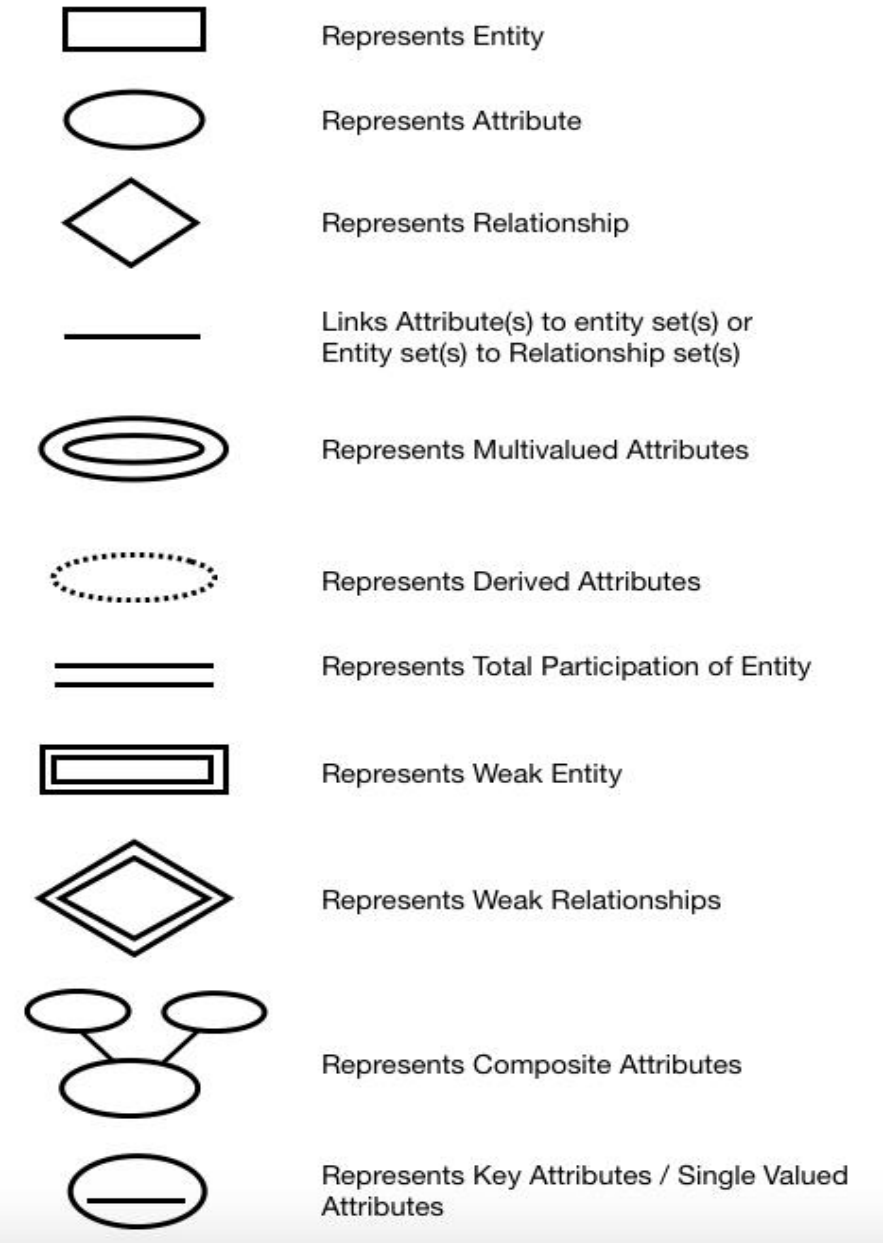
- For example, the age of a student can be a derived attribute because we can get it by the DOB of the student.
- Another example total and average mark of a student.

➢ **Complex Attribute:**

- If any attribute has the combining property of multi values and composite attributes, then it is called a complex attribute. It means if one attribute is made up of more than one attribute and each attribute can have more than one value, then it is called a complex attribute.
- For example, if a person has more than one office and each office has an address made from a street number and city. So the address is a composite attribute, and offices are multi valued attributes, so combing them is called complex attributes.

➢ **Key Attribute:**

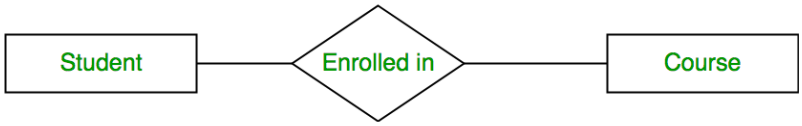
- Those attributes which can be identified uniquely in the relational table are called key attributes.
- For example, a student id is a unique attribute.



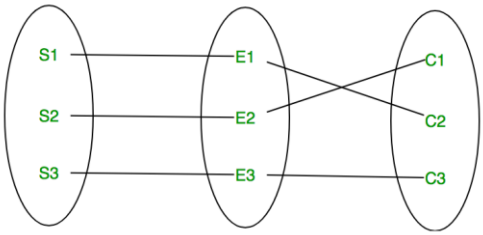
ER Model Digramatic Represenation

Relationship and Relationship Sets

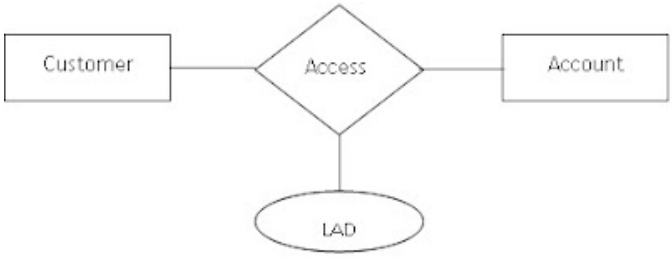
- A relationship is an association among several entities.
- For example, '**Enrolled in'** is a relationship type that exists between entity type Student and Course.
- In ER diagram, the relationship type is represented by a diamond and connecting the entities with lines.



- A **relationship set** is a set of relationships of the same type.
- The following relationship set depicts S1 as enrolled in C2, S2 is enrolled in C1, and S3 is enrolled in C3.

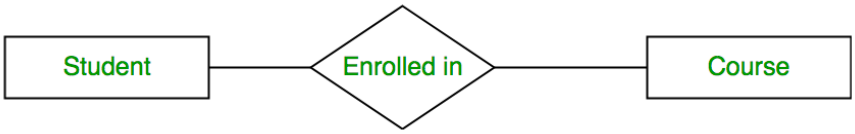


- Like entities, a relationship too can have attributes. These attributes are called **descriptive or relationship attributes**.
- Descriptive attributes are used to record information about the relationship, rather than about any one of the participating entities.

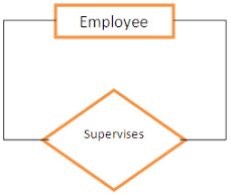


Degree of Relationship:

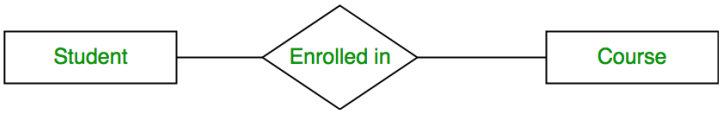
- A degree of relationship represents the number of entity types that associate in a relationship.
- Degree refers to the number of attributes/columns in a table.



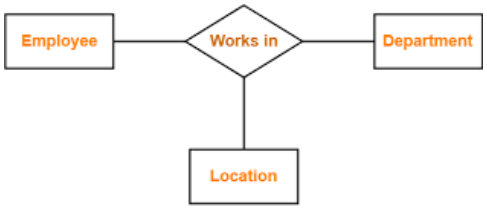
- For example, for above relationship degree is 2 because two entity types (student, course) are associated with relationship (enrolled in).
- Base on degree of relationships, there exist three types of relationships −
  1. **Unary relationship:** An association between two same entity types, it is known as a unary or recursive relationship.



2. **Binary relationship:** An association between two entity types, it is known as a binary relationship.



3. **Ternary relationship:** An association between three entity types, it is known as a ternary relationship



Ternary Relationship Set

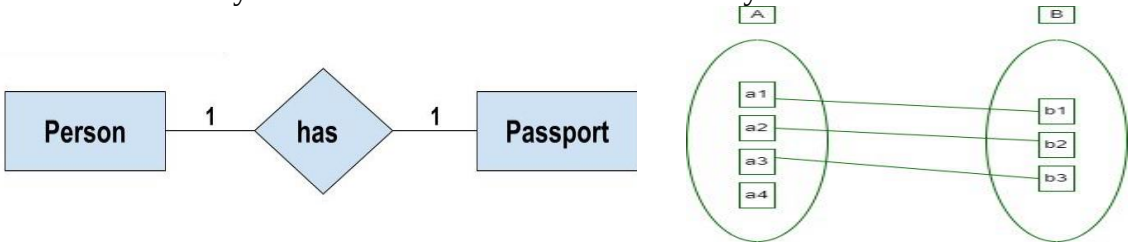
Cardinality: The number of entities to which another entity can be linked via a given relation set is called cardinality.

(Or)

The cardinality of a relationship is the number of tuples (rows) in a relationship.

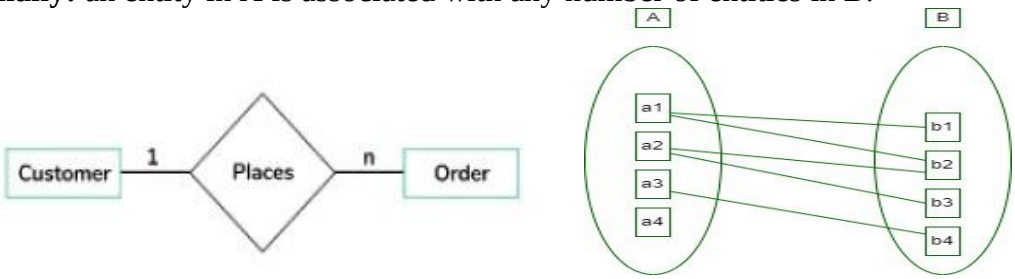
- Types of cardinalities in between tables are:
  1. one-to-one
  2. one-to-many
  3. many-to-one
  4. many-to-many

1. **One-to-one:** an entity in A is connected to at most one entity in B.



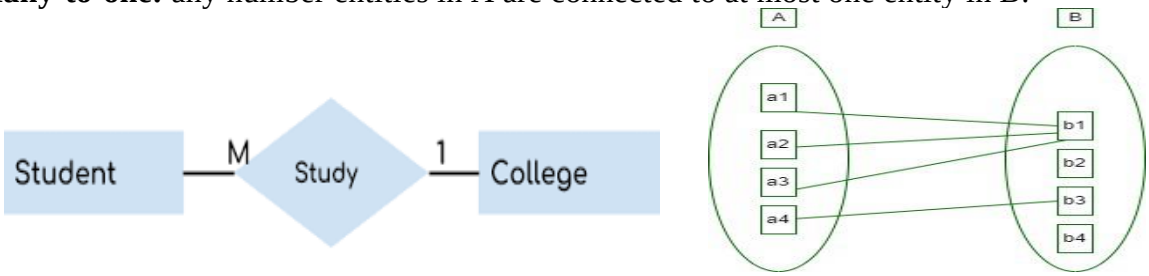
- **Example:** One person has one passport. They both serve one-to-one relationships.

2. **One-to-many:** an entity in A is associated with any number of entities in B.



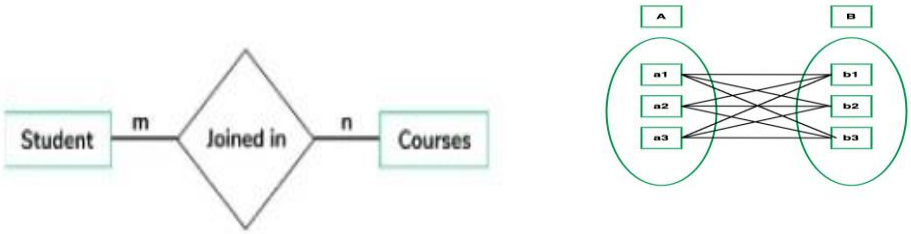
- **Example:** A customer can place multiples orders. They serve one-to-many relationships.

3. **Many-to-one:** any number entities in A are connected to at most one entity in B.



- **Example:** Multiple students may study in single college. Such a type of relationship is known as a many-to-one relationship.

4. **Many-to-many:** In this type of cardinality mapping, an entity in A is associated with any number of entities in B, and an entity in B is associated with any number of entities in A.



- **Example:** In a college, multiple students can join in multiple courses. They serve many-to-many relationships.

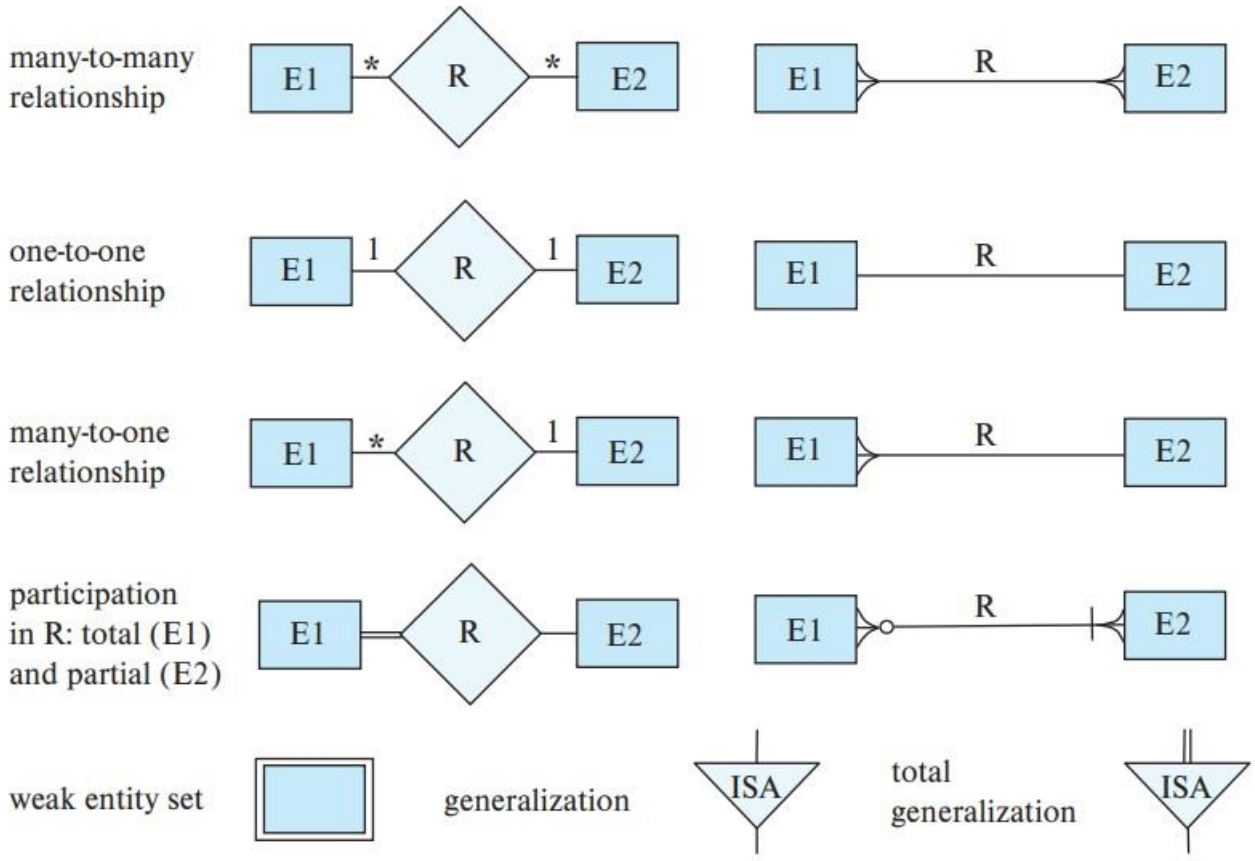


Figure 6.27 Alternative E-R notations.

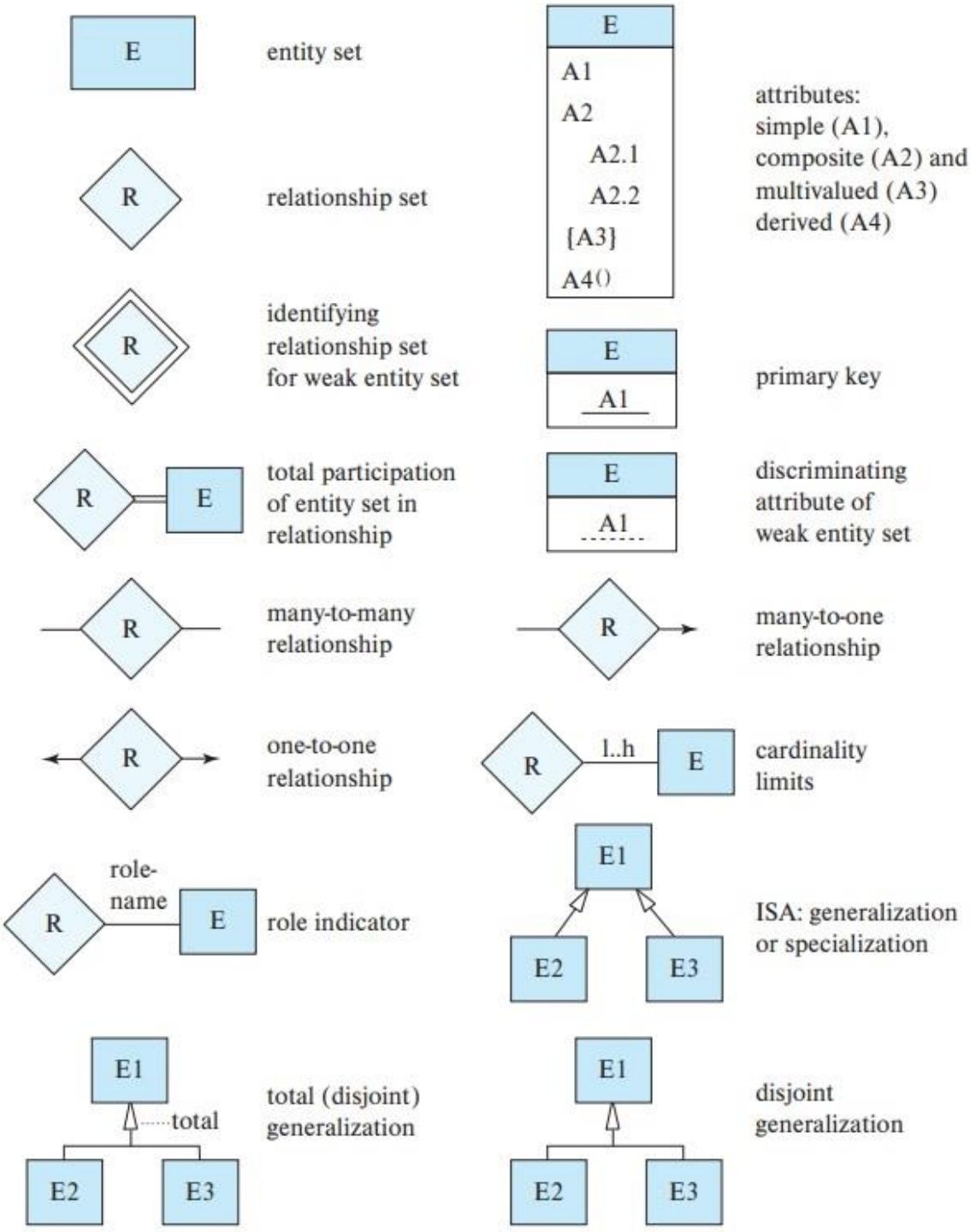


Figure 6.26 Symbols used in the E-R notation.

Constraints:

- Constraints are used for modelling limitations on the relations between entities.
- There are two types of constraints on the Entity Relationship (ER) model −
  - Mapping cardinality or cardinality ratio.
  - Participation constraints.

Mapping Cardinality

- It is expressed as the number of entities to which another entity can be associated via a relationship set.
- For the binary relationship set there are entity set A and B then the mapping cardinality can be one of the following −
  - One-to-one
  - One-to-many
  - Many-to-one
  - Many-to-many

One-to-one relationship

An entity set A is associated with at most one entity in B and an entity in B is associated with at most one entity in A.

One-to-many relationship

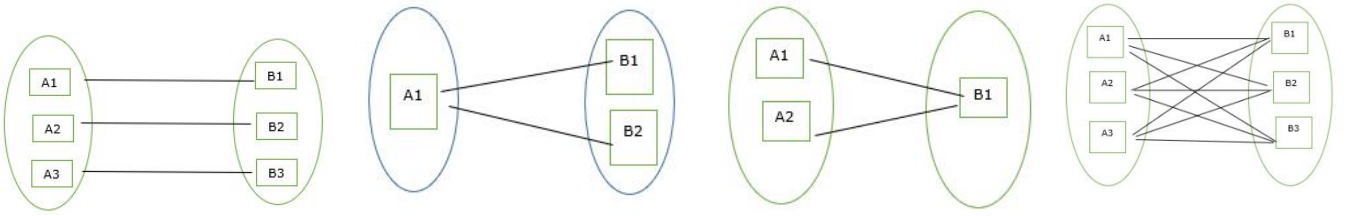
An entity set A is associated with any number of entities in B with a possibility of zero and an entity in B is associated with at most one entity in A.

Many-to-one relationship

An entity set A is associated with at most one entity in B and an entity set in B can be associated with any number of entities in A with a possibility of zero.

Many-to-many relationship

An entity set A is associated with any number of entities in B with a possibility of zero and an entity in B is associated with any number of entities in A with a possibility of zero.



One-to-One **One-to-Many** **Many-to-One** **Many-to-Many**

Participation constraint:

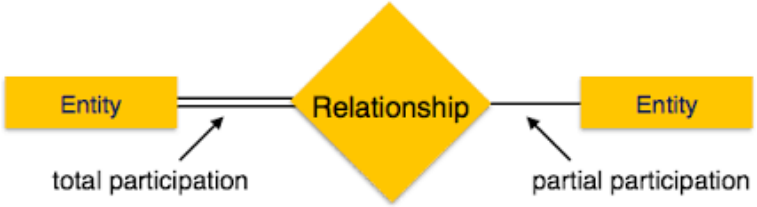
- Participation constraint specifies the existence of an entity when it is related to another entity in a relationship type.
- It is also called minimum cardinality constraint.

Total Participation

- Each entity in the entity set is involved in at least one relationship in a relationship set.

Partial Participation

- Each entity in entity set may or may not occur in at least one relationship in a relationship set.



Keys:

- Key is an attribute or set of attributes used to identify the tuples (rows) uniquely in a relation.
- Example: RollNo, Aadhar_No etc.
- Different Types of Keys in the Relational Model
  1. Super key
  2. Candidate keys
  3. Primary key
  4. Foreign key
  5. Composite key

1. Super key

- A super key is a set of one or more attributes that allow us to identify uniquely a tuple in the relation.
- For Example, STUD_NO, (STUD_NO, STUD_NAME), etc.
- A candidate key is a super key but vice versa is not true.
- It supports NULL values.

2. Candidate Key

- The minimal set of attributes that can uniquely identify a tuple is known as a candidate key.
- For Example, STUD_NO in STUDENT relation.
- There can be more than one candidate key in a relationship.

3. Primary key

- It is the first key used to identify one and only one instance of an entity uniquely.
- There can be more than one candidate key in relation out of which one can be chosen as the primary key.
- All other keys are alternate keys.
- It has no duplicate values; it has unique values.
- It cannot be NULL.

4. Foreign keys

- If an attribute can only take the values which are present as values of some other attribute, it will be a foreign key to the attribute to which it refers.
- It is a key it acts as a primary key in one table and it acts as secondary key in another table.
- It combines two or more relations (tables) at a time.
- They act as a cross-reference between the tables.

5. Composite Key

- Whenever a key consists of more than one attribute, it is known as a composite key.
- This key is also known as Compound or Concatenated Key.
- Different combinations of attributes may give different accuracy in terms of identifying the rows uniquely.

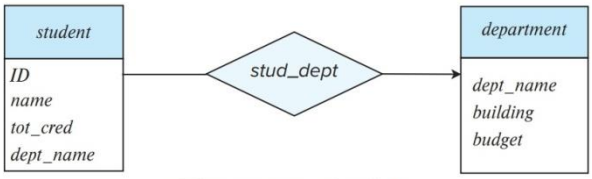
E-R Design Issues

We examine basic issues in the design of an E-R database schema

1. Common Mistakes in E-R Diagrams
2. Use of Entity Sets versus Attributes
3. Use of Entity Sets versus Relationship Sets
4. Binary versus n-ary Relationship Sets

1. Common Mistakes in E-R Diagrams

- A common mistake when creating E-R models is the use of the primary key of an entity set as an attribute of another entity set, instead of using a relationship.
- For example, in our university E-R model, it is incorrect to have dept name as an attribute of student, as depicted in Figure

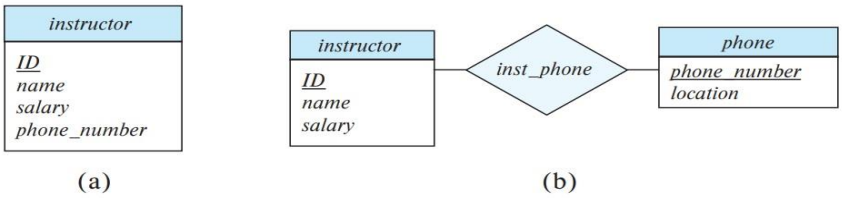


(a) Incorrect use of attribute

- Another related mistake that people sometimes make is to designate the primary key attributes of the related entity sets as attributes of the relationship set.
- For example, ID (the primary-key attributes of student) and ID (the primary key of instructor) should not appear as attributes of the relationship advisor.
- This should not be done since the primary-key attributes are already implicit in the relationship set.

2. Use of Entity Sets versus Attributes

- In the real-world situations, sometimes it is difficult to select the property as an attribute or an entity set.
- Consider the entity set instructor with the additional attribute phone number (Figure 6.23a.) It can be argued that a phone is an entity in its own right with attributes phone number and location; the location may be the office or home where the phone is located, with mobile (cell) phones perhaps represented by the value "mobile."
- If we take this point of view, we do not add the attribute phone number to the instructor. Rather, we create:
  - A phone entity set with attributes phone number and location.
  - A relationship set **inst_phone**, denoting the association between instructors and the phones that they have.

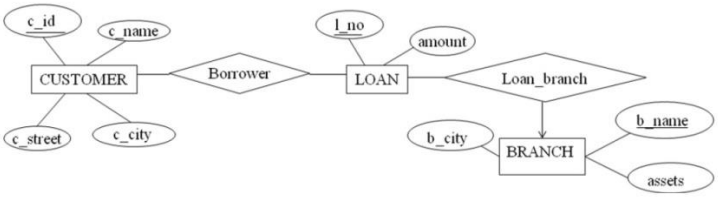


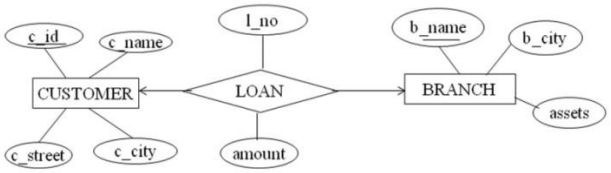
(a) (b)

Figure 6.23 Alternatives for adding *phone* to the *instructor* entity set.

3. Use of Entity Sets versus Relationship Sets

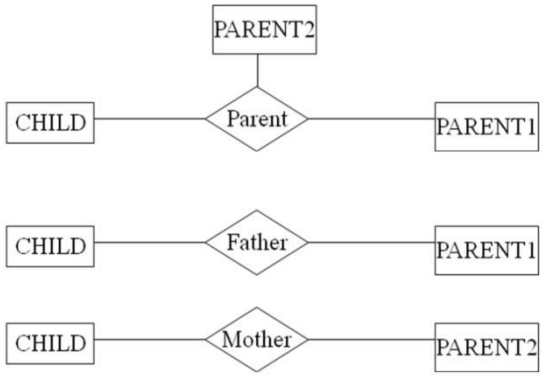
- Sometimes, an entity set can be better expressed in relationship set. Thus, it is not always clear whether an object is best expressed by an entity set or a relationship set





4. Binary versus n-ary Relationship Sets

- Relationships in databases are often binary. Some relationships that appear to be non-binary could actually be better represented by several binary relationships



In fact, it is always possible to replace a nonbinary (n-ary, for n > 2) relationship set by a number of distinct binary relationship sets. For simplicity, consider the abstract ternary (n = 3) relationship set R, relating entity sets A, B, and C. We replace the relationship set R with an entity set E, and we create three relationship sets as shown in Figure 6.25:

- ✓ R_A, a many-to-one relationship set from E to A.
- ✓ R_B, a many-to-one relationship set from E to B.
  R_C, a many-to-one relationship set from E to C.

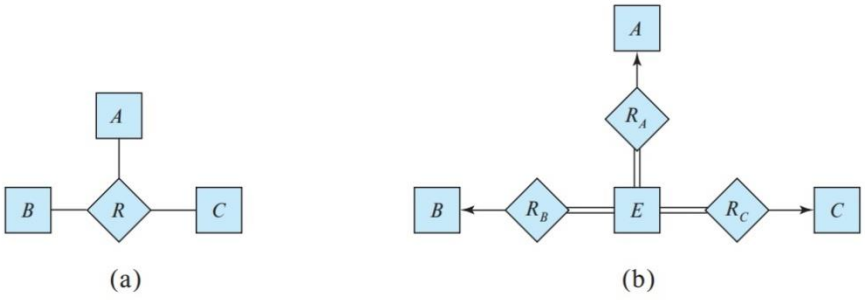
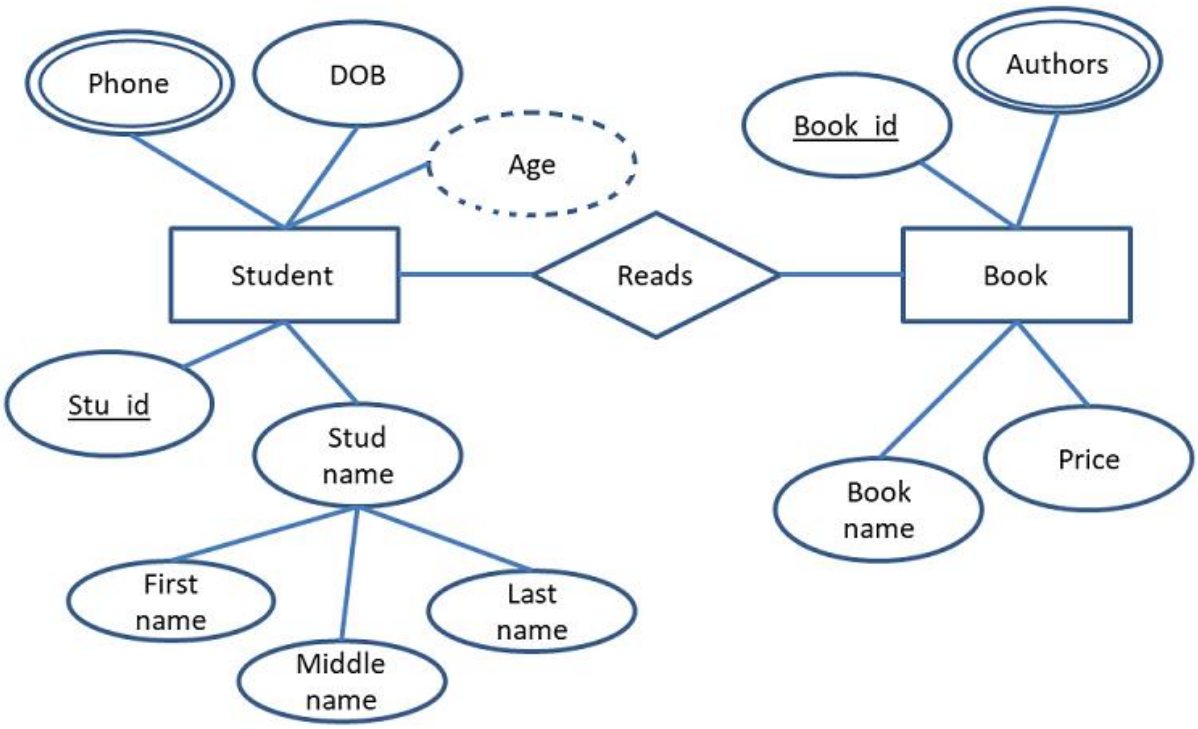


Figure 6.25 Ternary relationship versus three binary relationships.

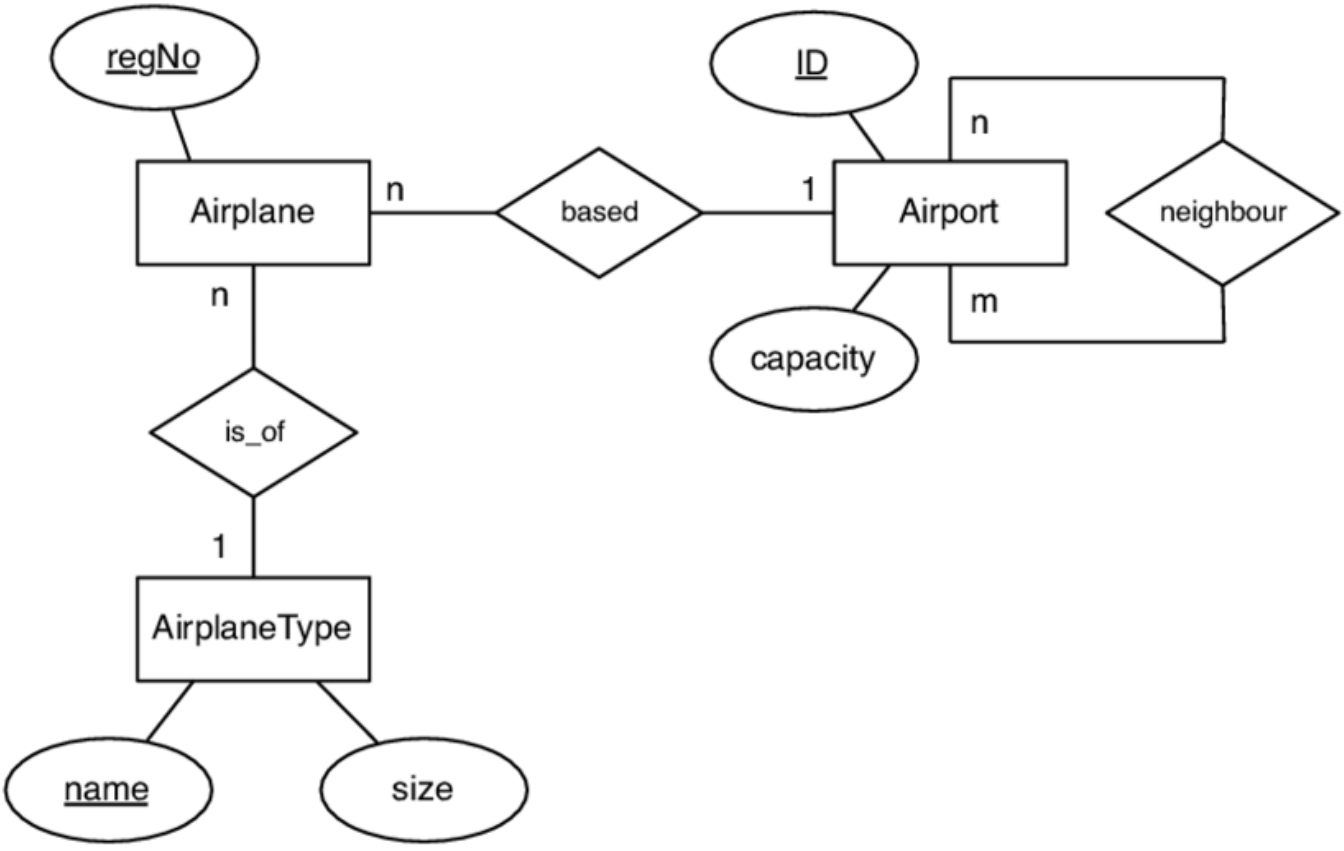
The guidelines that should be followed while designing an ER diagram are discussed below:

- ✓ Recognize entity sets
- ✓ Recognize relationship sets and participating entity sets
- ✓ Recognize attributes of entity sets and attributes of relationship sets
- ✓ Define binary relationship types and existence dependencies
- ✓ Define general cardinality, constraints, keys, and discriminators
- ✓ Design diagram

Entity-Relationship Diagram



Sample E-R Diagram 1



Sample E-R Diagram 2

Extended E-R Features:

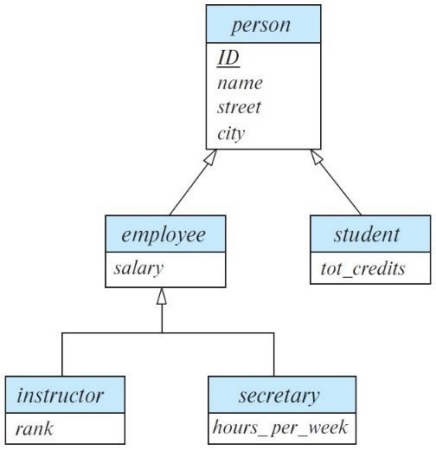
- The extended E-R features are **specialization**, **generalization**, **attribute inheritance**, and **aggregation**.

Specialization

- The process of designating subgroupings within an entity set is called specialization.
- Example, the entity set ***person*** may be further classified as one of the following: ***employee***, ***student***.
- Each of these person types is described by a set of attributes that includes all the attributes of entity set *person* plus possibly additional attributes.
- For example, *employee* entities may be described further by the attribute *salary*, whereas *student* entities may be described further by the attribute *tot cred*.
- The specialization of *person* allows us to distinguish among person entities according to whether they correspond to employees or students: in general, a person could be an employee, a student, both, or neither.
- It is a bottom-up approach and we can apply specialization repeatedly to refine a design.
- An entity set may be specialized by more than one distinguishing feature i.e., a person could be an employee, a student, both, or neither.
- In E-R diagram, specialization is depicted by a hollow arrow-head pointing from the specialized entity to the other entity.

Generalization:

- Generalization is a process in which a new entity is formed using the common attributes of two or more entities.
- It is a bottom-up approach in which two or more entities can be generalized to a higher-level entity if they have some attributes in common.
- Generalization is a containment relationship that exists between a *higher-level* entity set and one or more *lower-level* entity sets.
- In our example, *employee* is the higher-level entity set and *instructor* and *secretary* are lower-level entity sets.
- Higher- and lower-level entity sets also may be designated by the term's superclass and subclass, respectively.
- The *person* entity set is the superclass of the *employee* and *student* subclasses.
- For all practical purposes, generalization is a simple inversion of specialization.
- We apply both processes, in combination, in the course of designing the E-R schema for an enterprise.



Specialization and generalization

Attribute Inheritance

- The attributes of the higher-level entity sets are said to be inherited by the lower-level entity sets.
- For example, *student* and *employee* inherit the attributes of *person*.

Constraints on Generalization/Specializations:

To model an enterprise more accurately, the database designer may choose to place certain constraints on a particular generalization.

I) First constraint involves which entities can be members of a given lower_level entity set. Such membership may be one of the following

- **Condition defined (attribute defined)**
  In condition defined lower-level entity set is evaluated on the basis of whether or not an entity satisfies an explicit condition. Since all the lower-level entities are evaluated based on the same attribute this type of generalization is also called as "attribute **defined generalization.**
  Example: consider a higher-level entity set "Student" has an attribute "student_type". Here all the students are evaluated based on the single attribute "student_type" into undergraduate students and post graduate students.
- **User defined generalization**
  In this type of generalization lower-level entities are not evaluated by an explicit condition or attribute. Here lower-level entities are evaluated based on user choice.
  Example: consider a higher-level entity set "employee" and four lower-level entity sets like team1, team2, team3 and team4. Here In charge will take care about lower-level entity evaluation.

II) Second constraint relates to whether or not entities may belong to more than one lower-level entity set within a single generalization. The lower-level entity set may be one of the following

- **Disjoint generalization**
  It requires that an entity belong to no more than one lower-level entity.
  Example: consider a higher-level entity set "Student" has an attribute "student_type". Here all the students either belong to undergraduate category or postgraduate category. This is represented with a single arrow in the ER diagram.
- **Overlapping generalization**
  Here the same entity may belong to more than one lower-level entity.
  Example: consider a higher-level entity set "Person" and lower-level entity like "employee" or "student". Here person entity can be person or can be student. Such overlapping generalization is represented with the help of 2 different arrows.

III) Third Constraint on a specialization/ generalization is a completeness constraint, which specifies whether or not an entity in the higher-level entity set must belong to at least one of the lower-level entity sets within the generalization/specialization. This constraint may be one of the following:

- **Total specialization or generalization**: Each higher-level entity must belong to a lower-level entity set.
  Example: consider a higher-level entity set "Student" has an attribute "student_type". Here all the students either belong to undergraduate category or postgraduate category. Here all the students are participating in the process. Therefore, it's a total generalization.

- **Partial specialization or generalization**: Some higher-level entities may not belong to any lower-level entity set.
  Example: consider a higher-level entity set "employee" and four lower level entity sets like team1, team2, team3 and team4. Here few employees those who are probation they will not be assigned to any of the team.

Aggregation:

- Aggregation is an abstraction through which relationships are treated as higher level entities.

- One limitation of the E-R model is that it cannot express relationships among relationships.
- Aggregation is used to represent a relationship between a whole object and its component parts.
- If we need to express a relationship among relationships, then we should use aggregation.
- To illustrate the need for such a construct, consider the ternary relationship *proj_guide*, which we saw earlier, between an *instructor*, *student* and *project.*

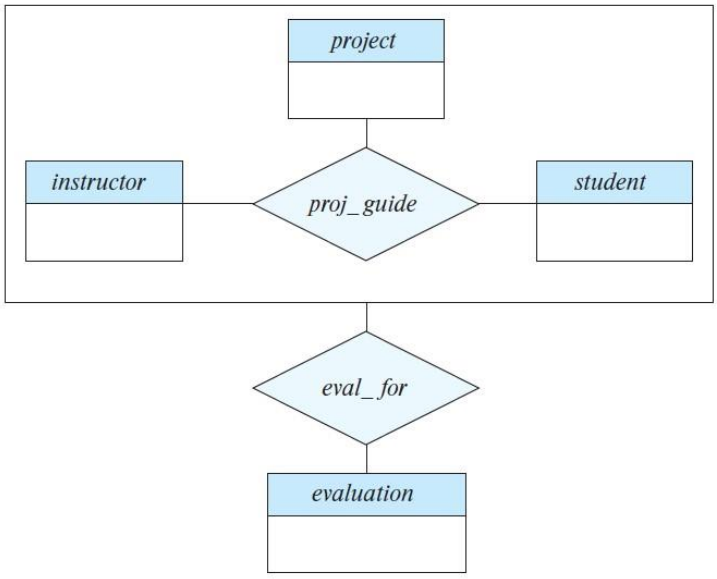
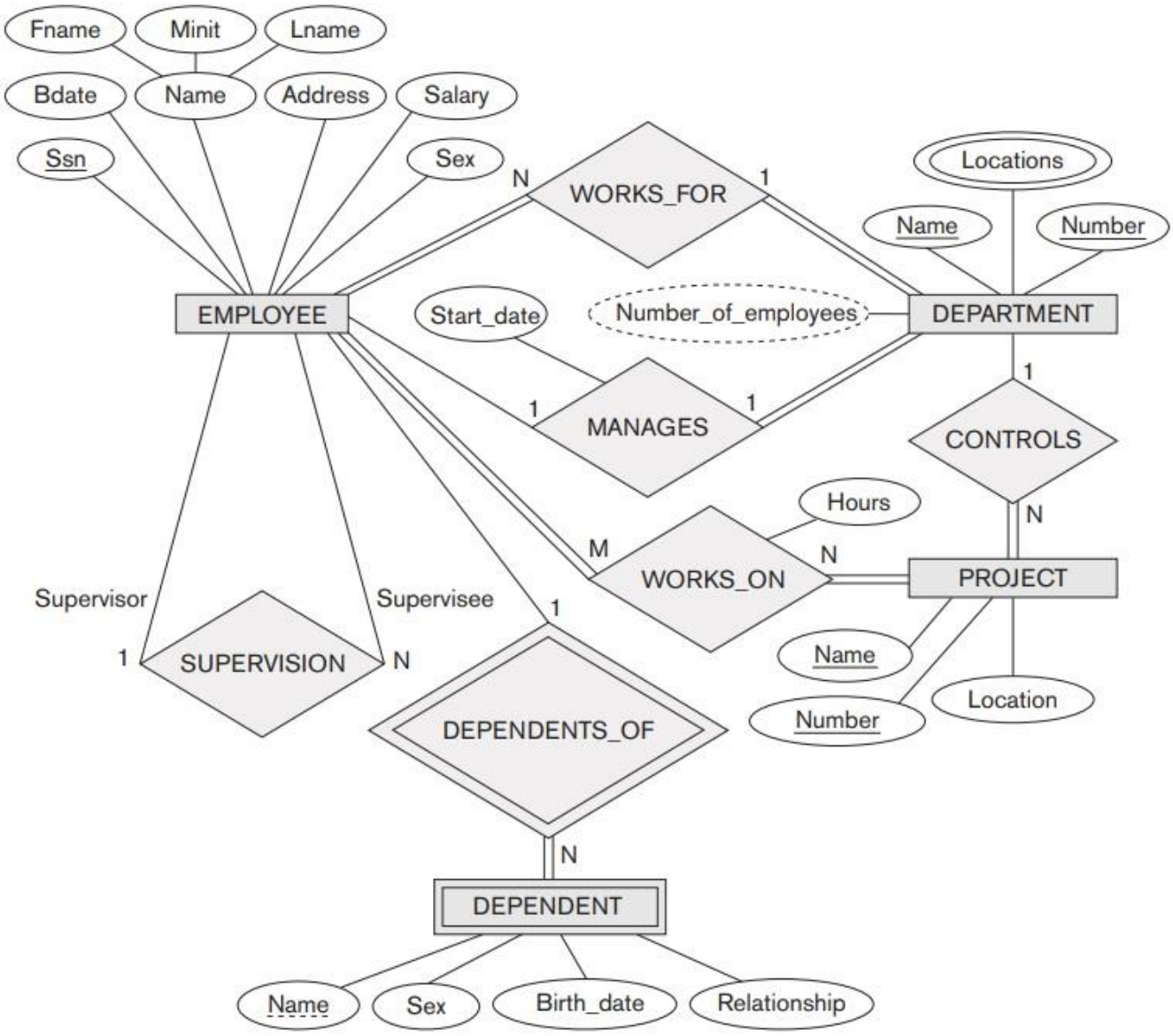


Figure 6.20 E-R diagram with aggregation.

Database Design with ER model:



E-R DIAGRAM FOR COMPANY DATABASE

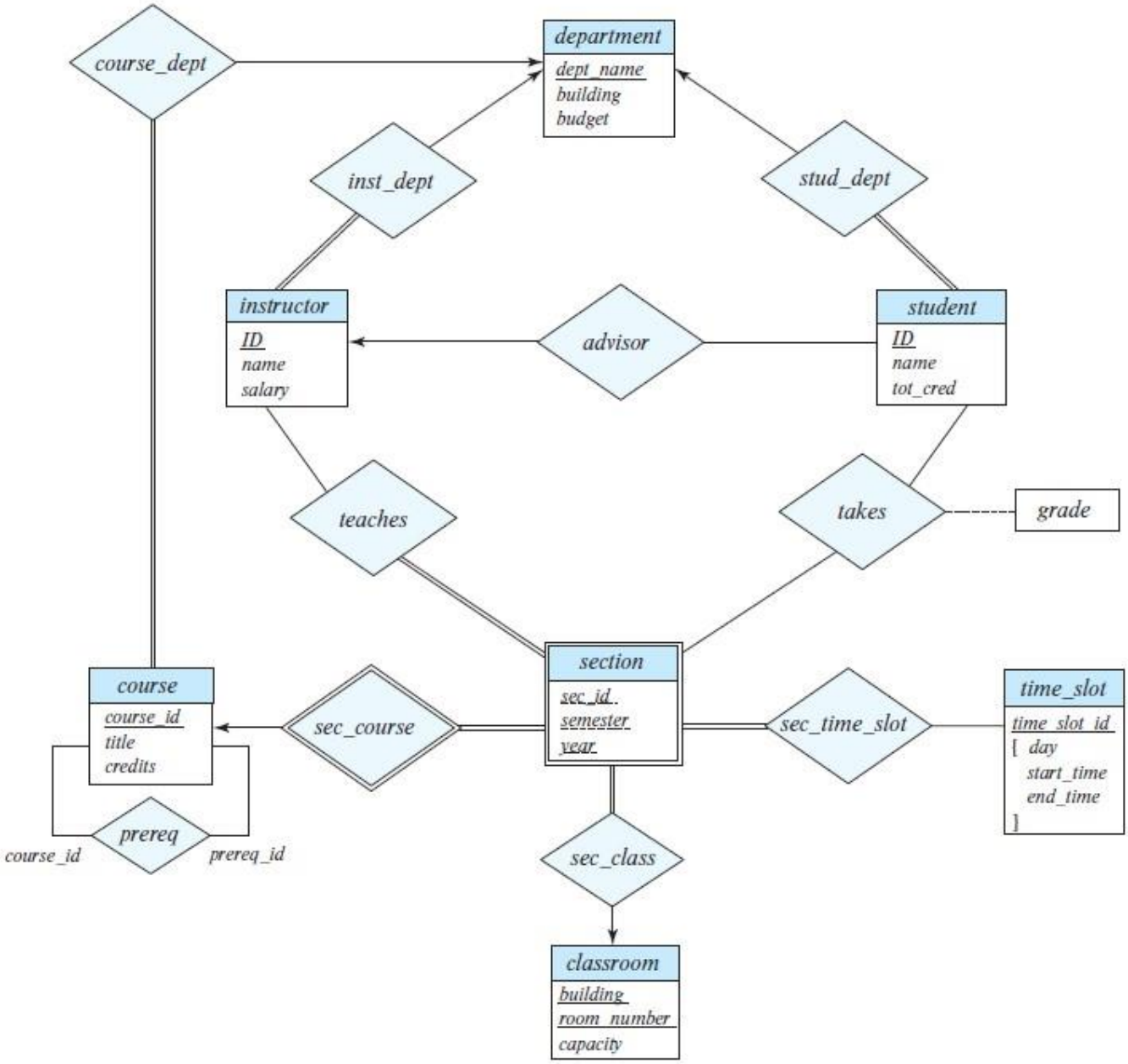
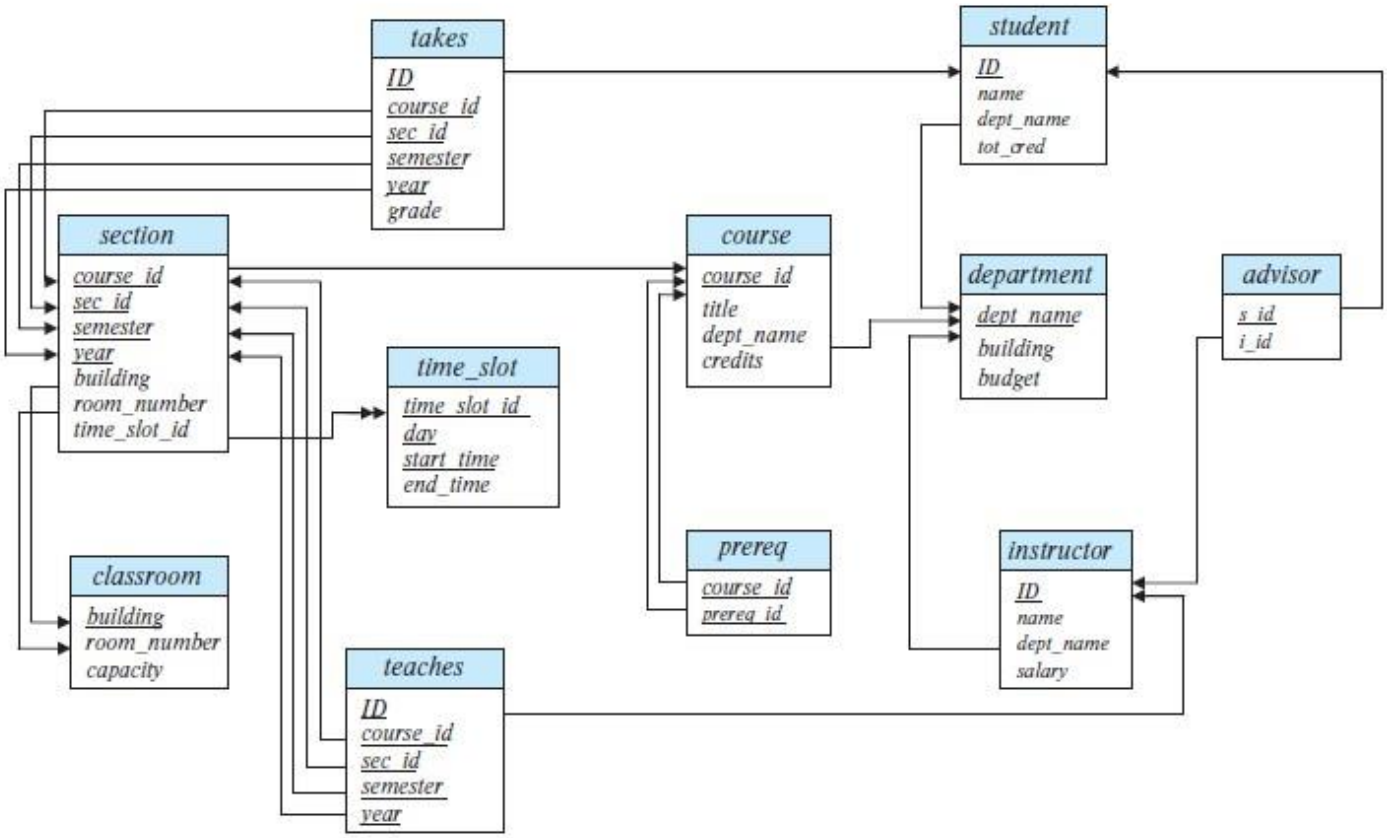


Figure 6.15 E-R diagram for a university enterprise.

Database Design for a schema:

classroom(*building*, *room_number*, *capacity*)
department(*dept_name*, *building*, *budget*)
course(*course_id*, *title*, *dept_name*, *credits*)
instructor(*ID*, *name*, *dept_name*, *salary*)
section(*course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *time_slot_id*)
teaches(*ID*, *course_id*, *sec_id*, *semester*, *year*)
student(*ID*, *name*, *dept_name*, *tot_cred*)
takes(*ID*, *course_id*, *sec_id*, *semester*, *year*, *grade*)
advisor(*s_ID*, *i_ID*)
time_slot(*time_slot_id*, *day*, *start_time*, *end_time*)
prereq(*course_id*, *prereq_id*)

Schema of the university database.



Schema diagram for the university database