Tribhuvan University

B.Sc. CSIT

Butwal Multiple Campus

Database Management System
CSC260
Unit 3: Data Modelling Using the ER Model

Er. Nirmal Thapa

Email: nirmalthapa562@gmail.com

Butwal Multiple Campus

Tribhuvan University

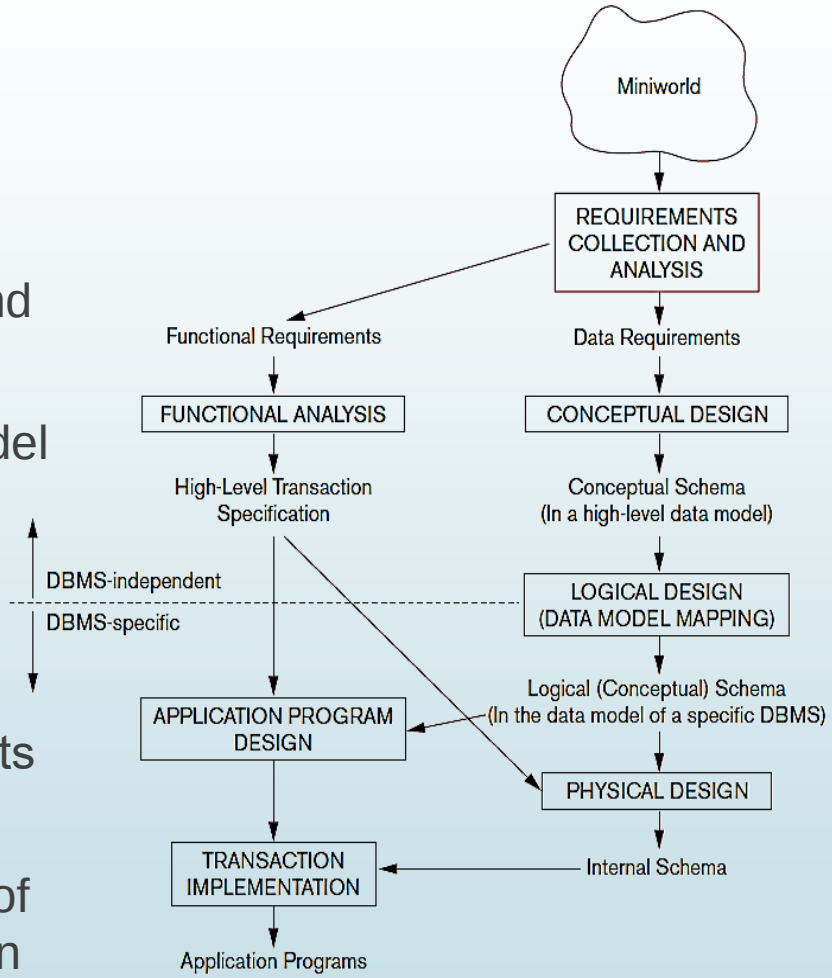
Concept of Conceptual Design

Using High-Level Conceptual Data Models for Database Design:

- **Requirements Collection and Analysis:**
  - Database designers understand and document the **data requirements** of the database users
  - Result of this step is a concisely written set of users' requirements
  - These requirements should be specified in as detailed and complete as possible
- **Functional Requirements:**
  - Consist of the user defined operations (or transactions) that will be applied to the database, including both retrievals and updates.
  - In software design, it is common to use data flow diagrams, sequence diagrams, etc. to specify functional requirements.

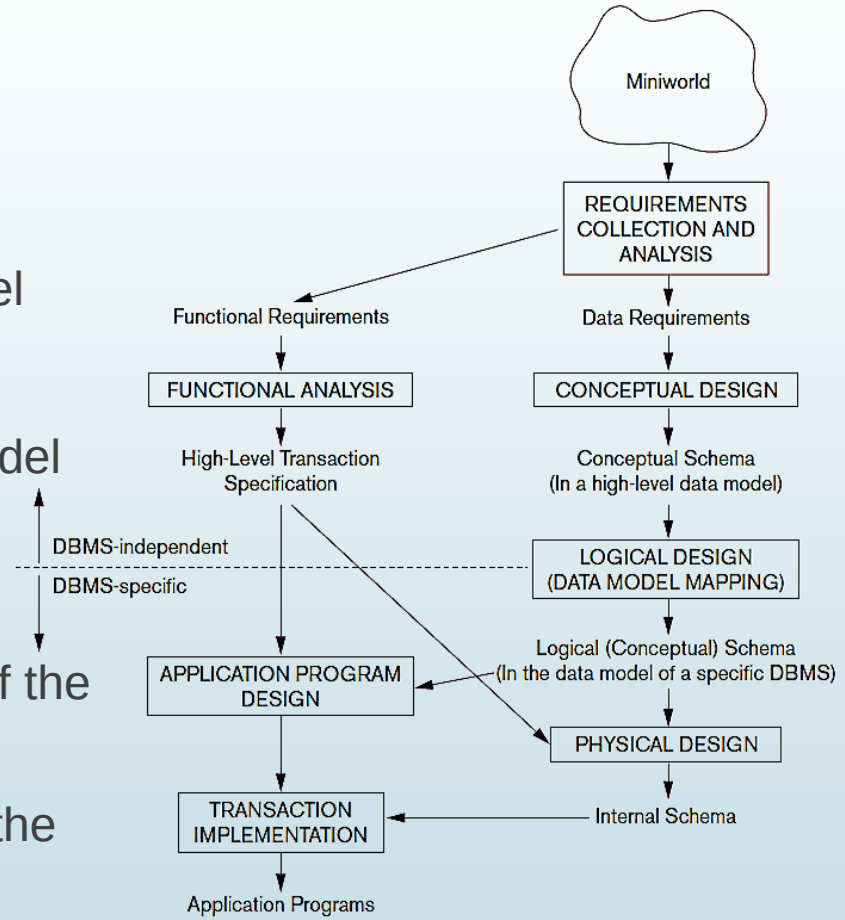
Concept of Conceptual Design

- **Conceptual Design:**
  - Using a high level conceptual data model
  - A concise description of the data requirements of the users
  - Includes detailed descriptions of the entity types, relationships, and constraints
  - Expressed using the concepts provided by the high-level data model
  - No implementation details
  - Ease of understanding. Used to communicate with non-technical users
  - Also used as a reference to ensure that all users' data requirements are met and that the requirements do not conflict
  - Enables db designers to concentrate on specifying the properties of the data, without being concerned with storage and implementation details
  - This makes it is easier to create a good conceptual database design



Concept of Conceptual Design

- **Logical Design**
  - Actual implementation of the database using commercial DBMS
  - The conceptual database schema
  - Most current commercial DBMSs use an implementation data model
  - E.g. Relational, O-O, O-R, Network, hierarchical
  - The conceptual schema is transformed from the high-level data model into the implementation data model
  - This step is called logical design or data model mapping
  - Its result is a database schema in the implementation data model of the DBMS
  - Data model mapping is often automated or semi-automated within the database design tools.



Concept of Conceptual Design

Logical Design

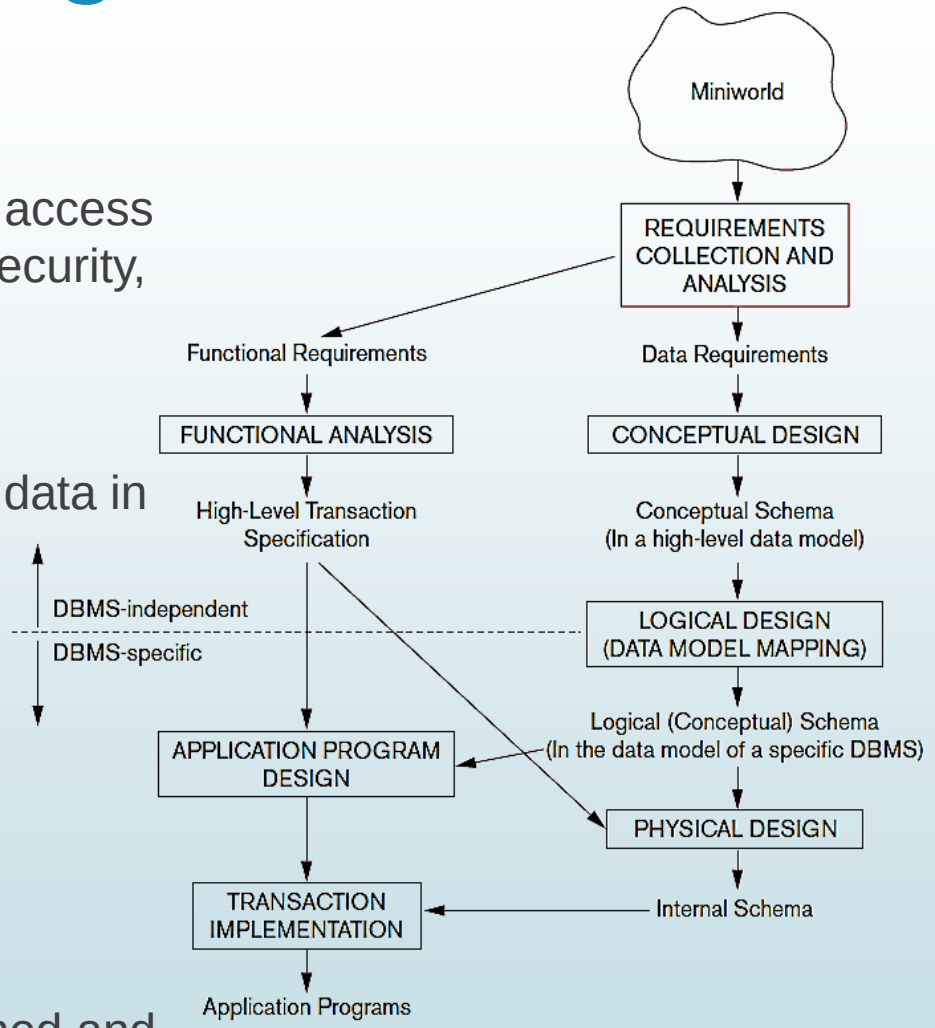
- The logical design is generally performed in four steps, which are as follows.

1. Map conceptual model to logical model components
   - Map strong/weak entities and higher degree relationship
2. Validate logical model using normalization
3. Validate logical model integrity constraints
   - Eg. CLASS_CODE is a valid class code.
     Type: numeric
     Range: low value = 1000 high value = 9999
     Display format: 9999
     Length: 4
4. Validate logical model against user requirements
   - validate all logical model definitions against all end-user data, transaction, and security requirements.

Concept of Conceptual Design

Physical Design

- Process of determining the data storage organization and data access characteristics of the database in order to ensure its integrity, security, and performance
- This is the last stage in the database design process
- A very technical job that affects not only the accessibility of the data in the storage device(s) but also the performance of the system.
- Composed of the following steps.
  1. Define data storage organization
  2. Define integrity and security measures
  3. Determine performance measurements
- In parallel with these activities, application programs are designed and implemented



Sample Database Design

Example : Let us see the example db application, called **COMPANY**

Requirement gathered:

- The company is organized into departments. Each department has a unique name, a unique number, and a particular employee who manages the department. We keep track of the start date when that employee began managing the department. A department may have several locations.

- A department controls a number of projects, each of which has a unique name, a unique number, and a single location.

Sample Database Design

Example : Let us see the example db application, called **COMPANY**

Requirement gathered:

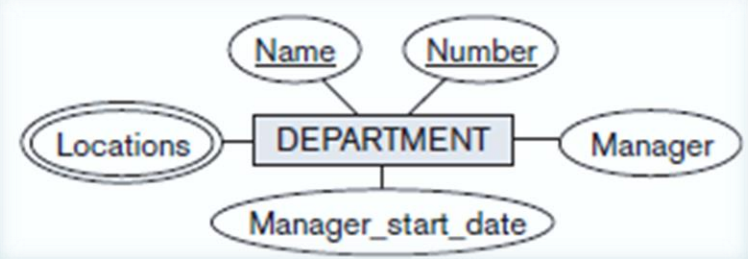
- Employee details: name, SSN, address, salary, gender . An employee is assigned to one department, but may work on several projects
- It is required to keep track of the current number of hours per week that an employee works on each project, as well as the direct supervisor of each employee (who is another employee).
- The database will keep track of the dependents of each employee for insurance purposes, including each details

Initial Conceptual Design of COMPANY Database

We can identify 4 entity types based on the requirements:

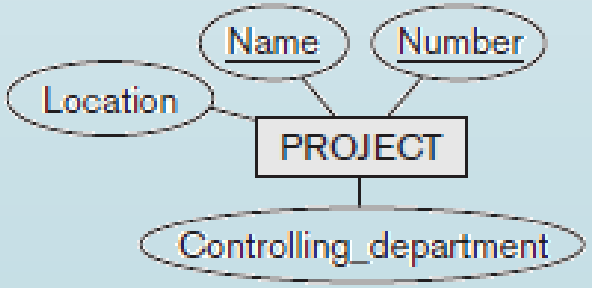
DEPARTMENT :

- Attributes: Name, Number, Locations, Manager, and Manager_start_date, Locations is the only multivalued attribute.



PROJECT:

- Attributes : Name, Number, Location, and Controlling_department.

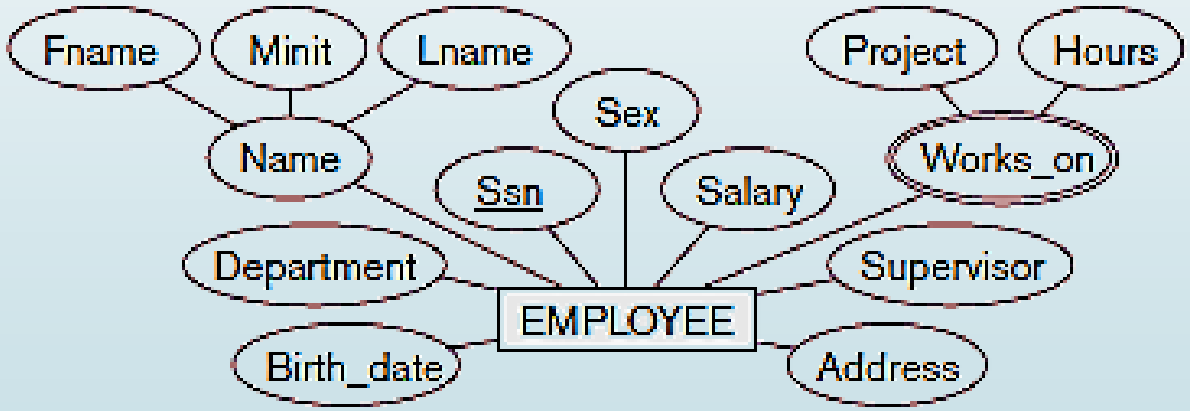


Initial Conceptual Design of COMPANY Database

We can identify 4 entity types based on the requirements:

EMPLOYEE :

Attributes: Name, Ssn, Sex, Address, Salary, Birth_date, Department, and Supervisor. Both Name and Address may be composite attributes

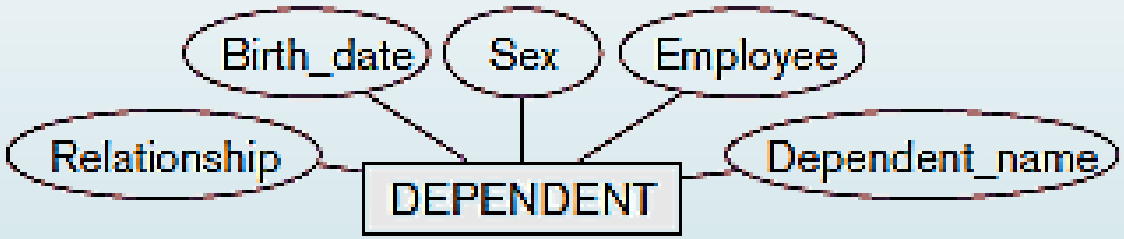


Initial Conceptual Design of COMPANY Database

We can identify 4 entity types based on the requirements:

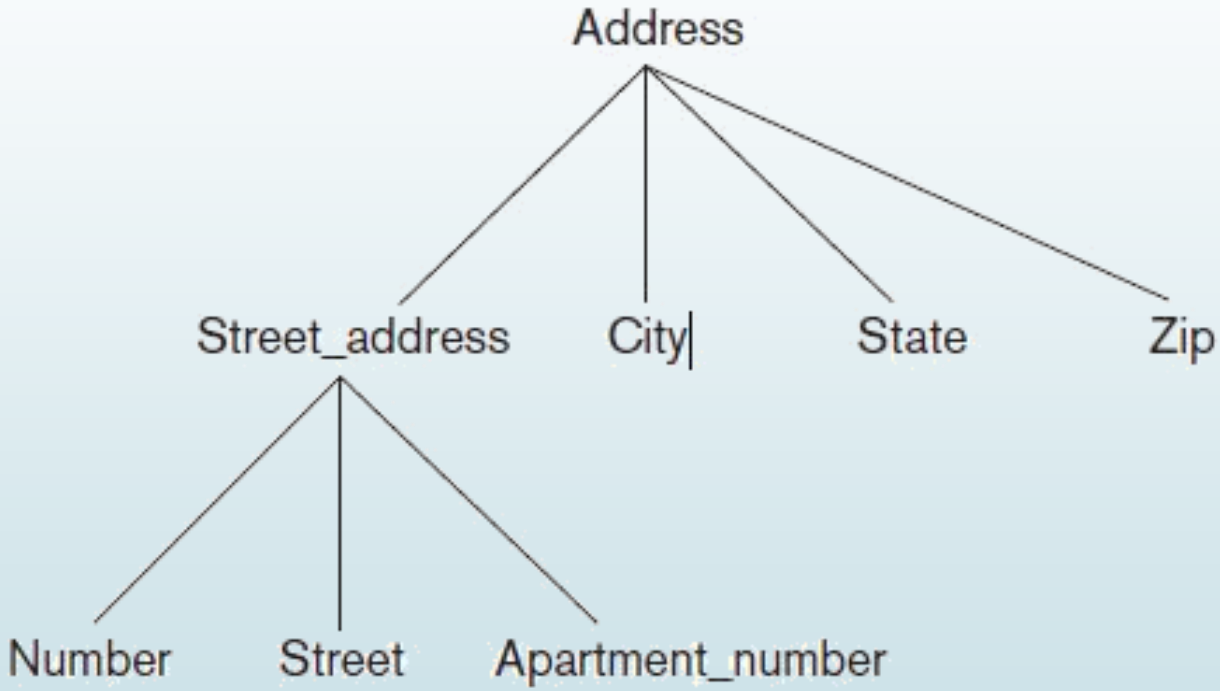
DEPENDENT

Attributes Employee, Dependent_name, Sex, Birth_date, and Relationship (to the employee).



Entity Types, Entity Sets, Attributes, and Keys

Composite versus Simple (Atomic) Attributes:



Entity Types, Entity Sets, Attributes, and Keys

Entity and attributes:

- The basic concept that the ER model represents is an entity
- A thing or object in the real world with an independent existence.
- An entity may be an object with a physical existence (for example, a particular person, car, house, or employee)
- It may be an object with a conceptual existence (for instance, a company, a job, or a university course)

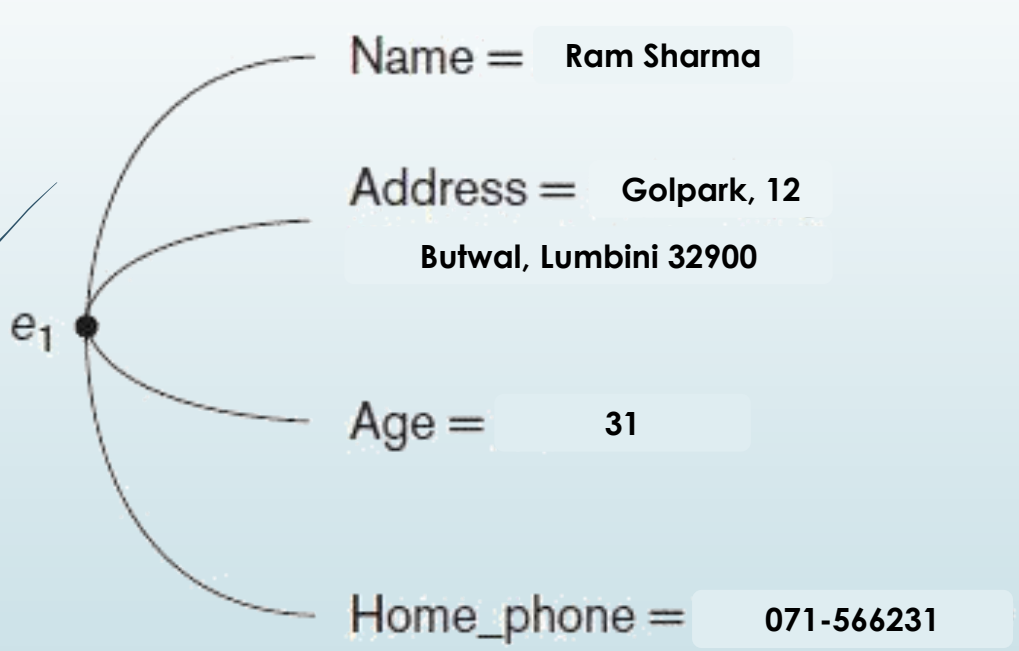
Entity Types, Entity Sets, Attributes, and Keys

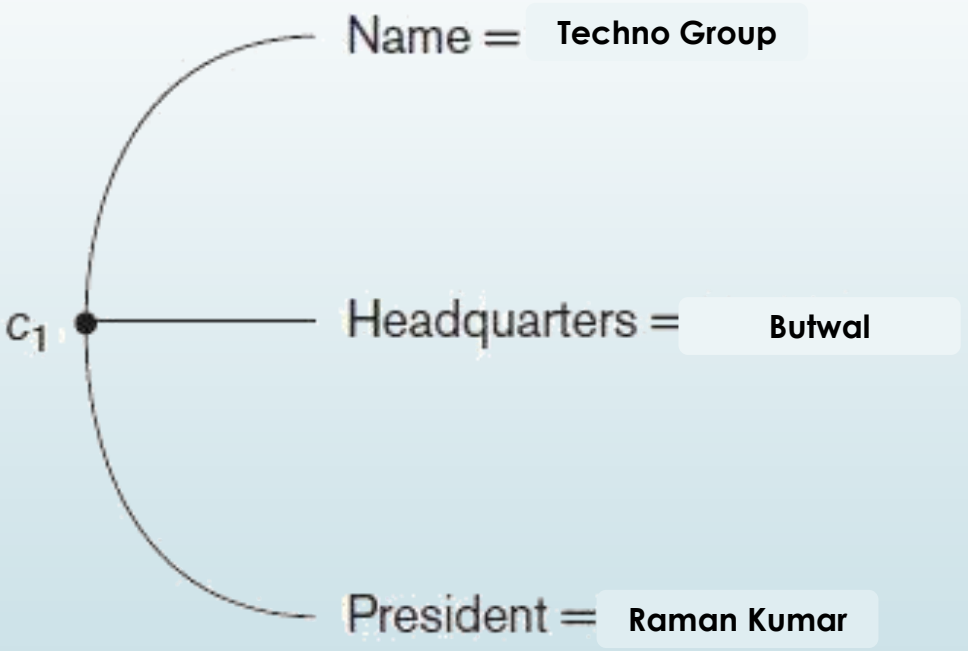
Entity and attributes:

- Each entity has attributes
  - Particular properties that describe it
  - For example, an EMPLOYEE entity may be described by the employee's name, age, address, salary, and job.
  - A particular entity will have a value for each of its attributes.
  - The attribute values that describe each entity become a major part of the data stored in the database.

Entity Types, Entity Sets, Attributes, and Keys

Entity and attributes:





Entity Types, Entity Sets, Attributes, and Keys

Several types of attributes occur in the ER model:

- Simple (Atomic) versus composite
- Singlevalued versus multivalued
- Stored versus derived

Entity Types, Entity Sets, Attributes, and Keys

Composite Attributes

- ★ Can be divided into further parts.
- ★ Ex: Name → First Name, Middle Name, Last Name

Simple Attributes

- ★ Cannot be divided further.
- ★ Ex: Weight → cannot be further divided.

Entity Types, Entity Sets, Attributes, and Keys

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 person.

Entity Types, Entity Sets, Attributes, and Keys

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

Entity Types, Entity Sets, Attributes, and Keys

Complex Attributes:

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

Entity Types, Entity Sets, Attributes, and Keys

Null Values:

- In some cases, a particular entity may not have an applicable value for an attribute
- Eg. the **Apartment_number** attribute of an address applies only to addresses that are in apartment
- Similarly, a **College_degrees** attribute
- For such situations, a special value called **NULL** is created.
- NULL can also be used if we do not know the value of an attribute for a particular entity
- Eg. if we do not know the home phone number of person
- The meaning of the former type of **NULL** is not applicable, whereas the meaning of the latter is unknown.
- The unknown category of **NULL** can be further classified into two cases.
  1. When it is known that the attribute value exists but is missing eg. **height**
  2. when it is not known whether the attribute value exists eg. **home_phone**

Entity Types, Entity Sets, Attributes, and Keys

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.

Entity Types, Entity Sets, Attributes, and Keys

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.

Relationship Degree, Role names and Recursive Relationship

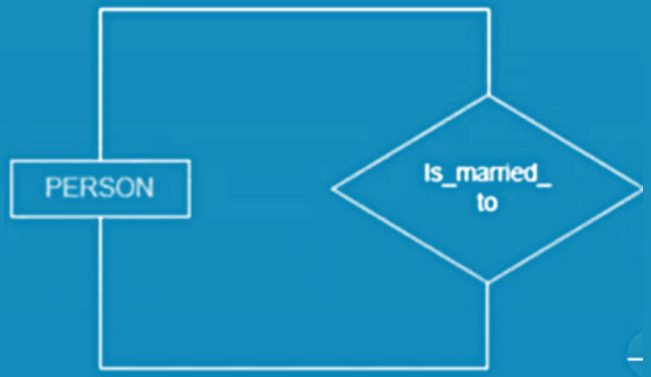
- Relationship → association among 2 or more entities.
- Ex: teacher teaches student

  ↓

  relationship

★ Degree of Relationship: Denotes the number of entity types that participate in a relationship.

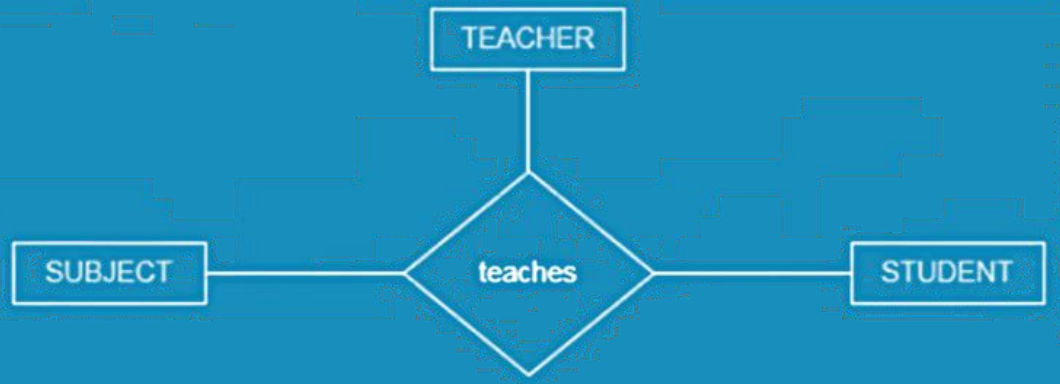
1. Unary relationship:
   - ➢ Exists when there is an association with only one entity.



Relationship Degree, Role names and Recursive Relationship

Degree of Relationship:

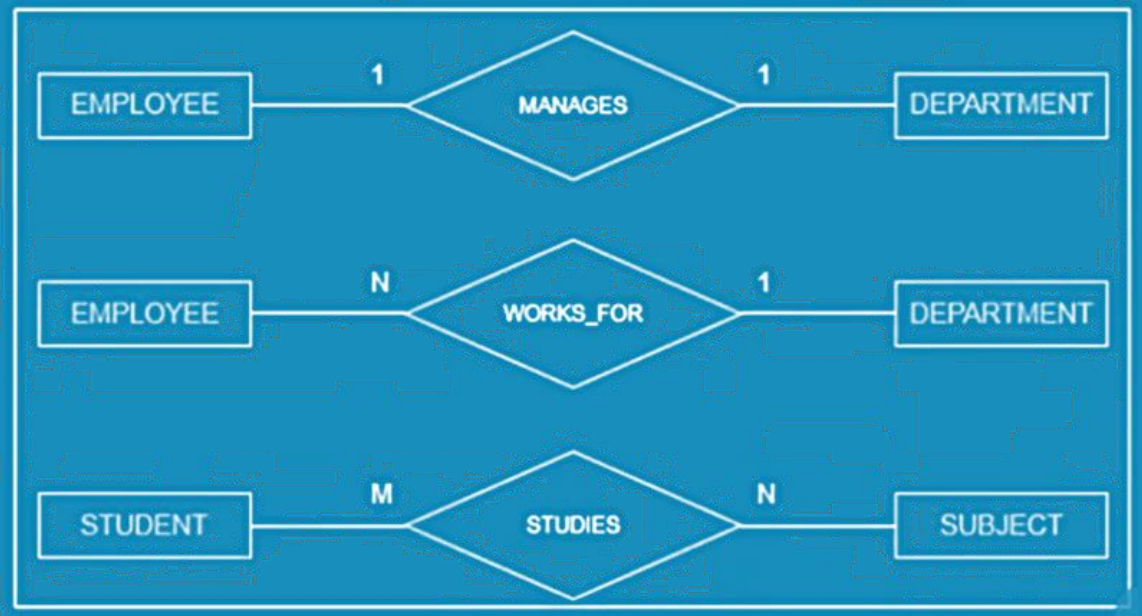
2. Binary relationship:
   - Exists when there is an association among two entities.
3. Ternary relationship:
   - Exists when there is an association among three entities.



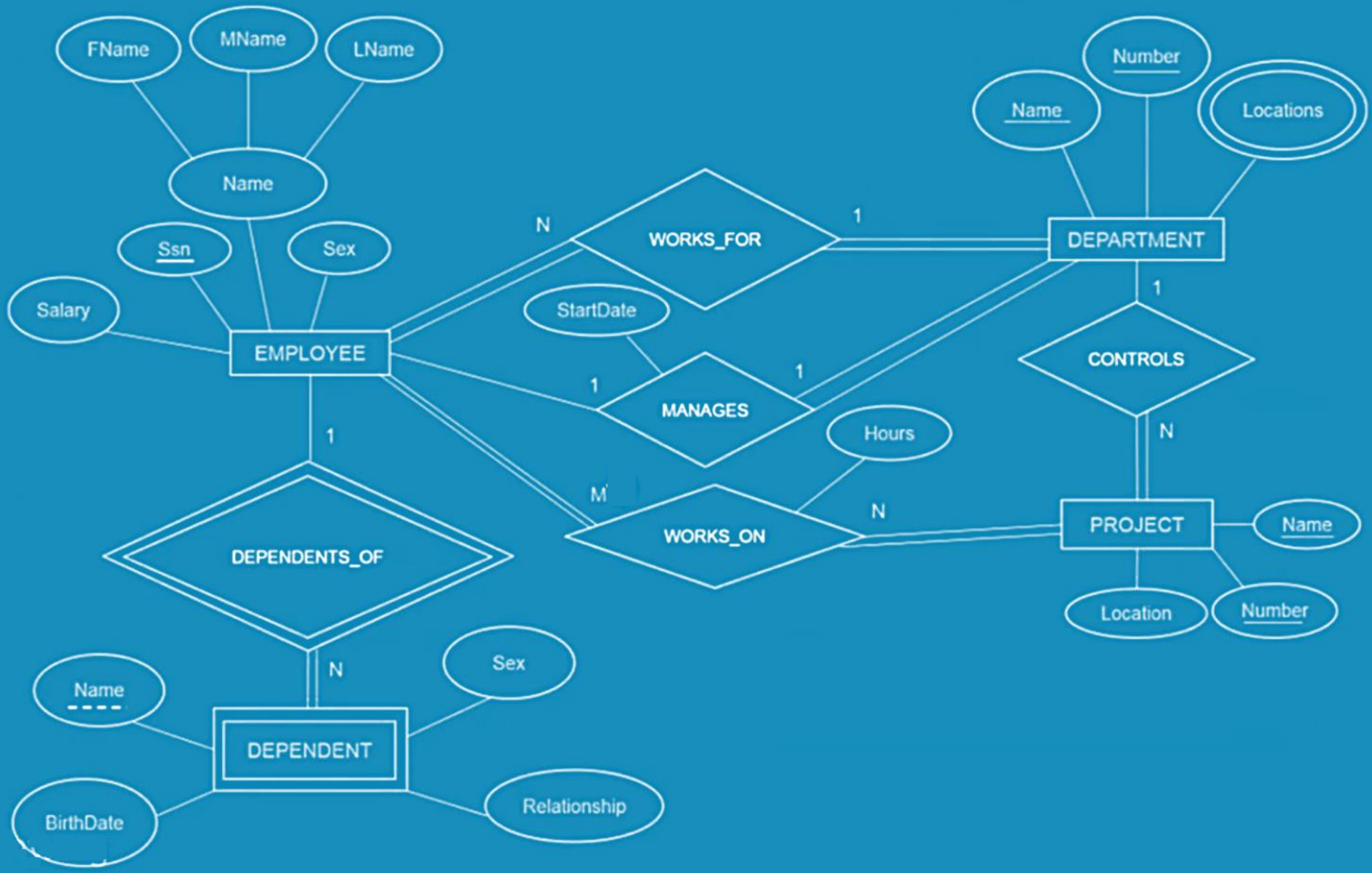
Relationship Degree, Role names and Recursive Relationship

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.



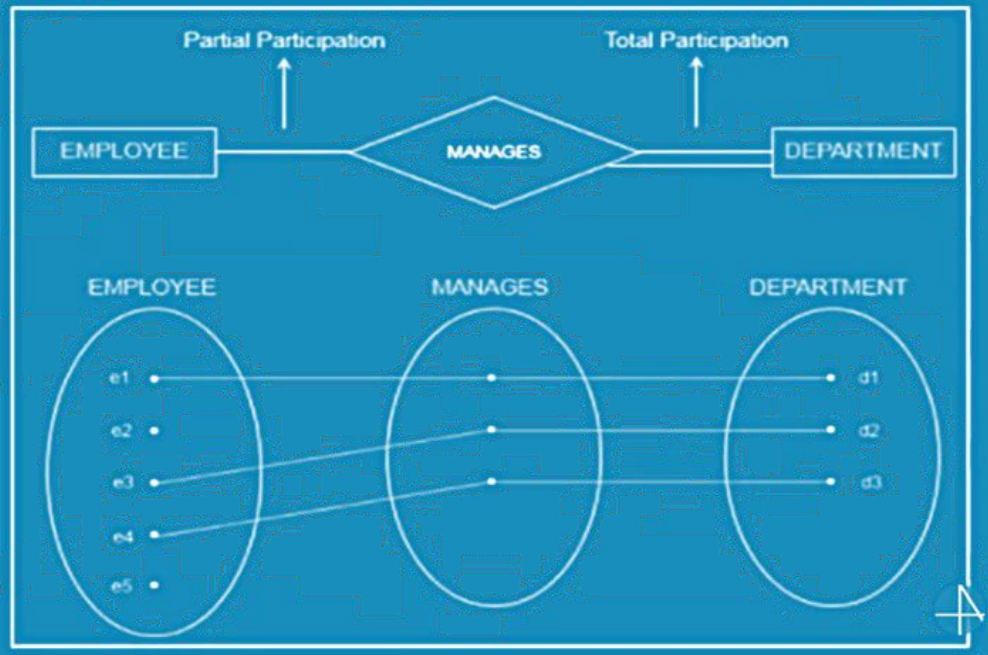
Relationship Degree, Role names and Recursive Relationship



Relationship Degree, Role names and Recursive Relationship

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.



Relationship Degree, Role names and Recursive Relationship

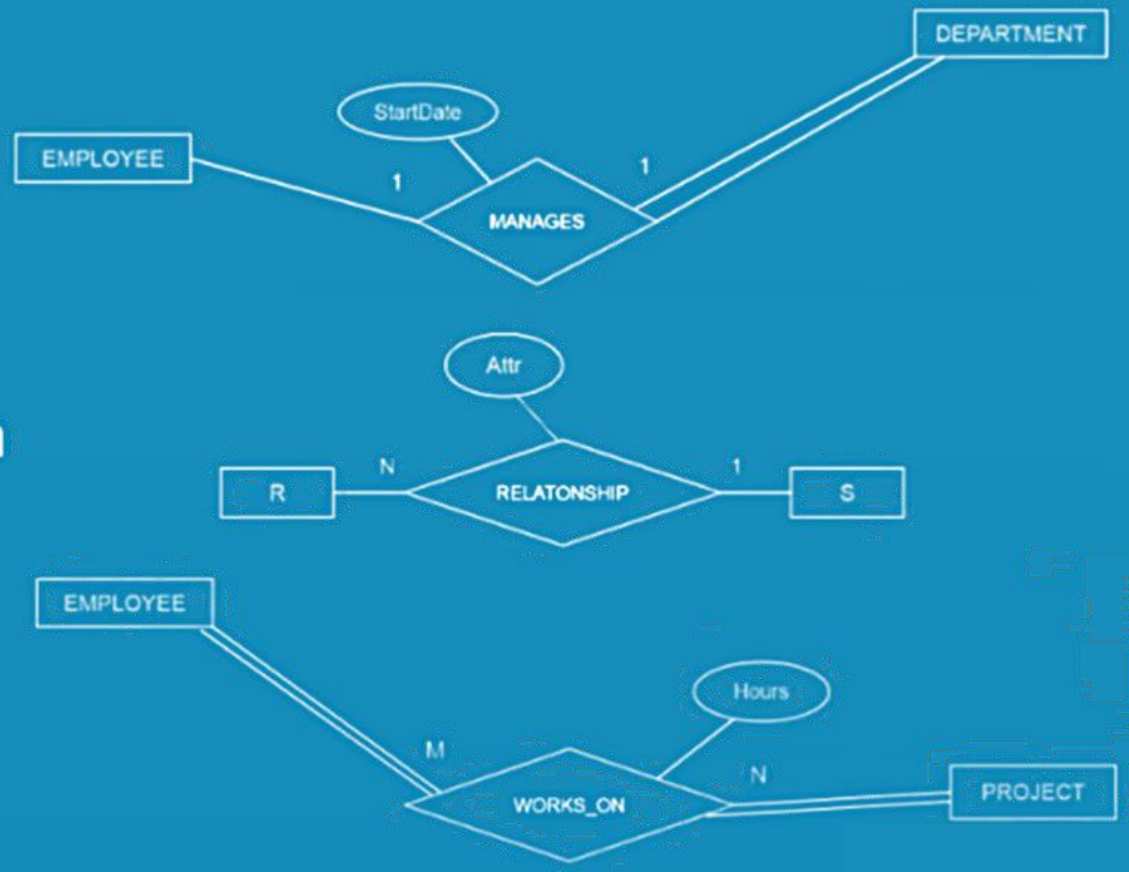
Attributes of Relationship Types

- Attributes of 1:1 or 1:N relationship types can be migrated to one of the participating entity types.
  - ➔ In 1:1 relationship type, attributes can be migrated to either of the entity types.
  - ➔ In 1:N or N:1 relationship type, attributes are migrated only to the entity type on the N-side of the relationship.

Relationship Degree, Role names and Recursive Relationship

Attributes of Relationship Types

- In M:N relationship type, some attributes can be determined by a combination of participating entities.



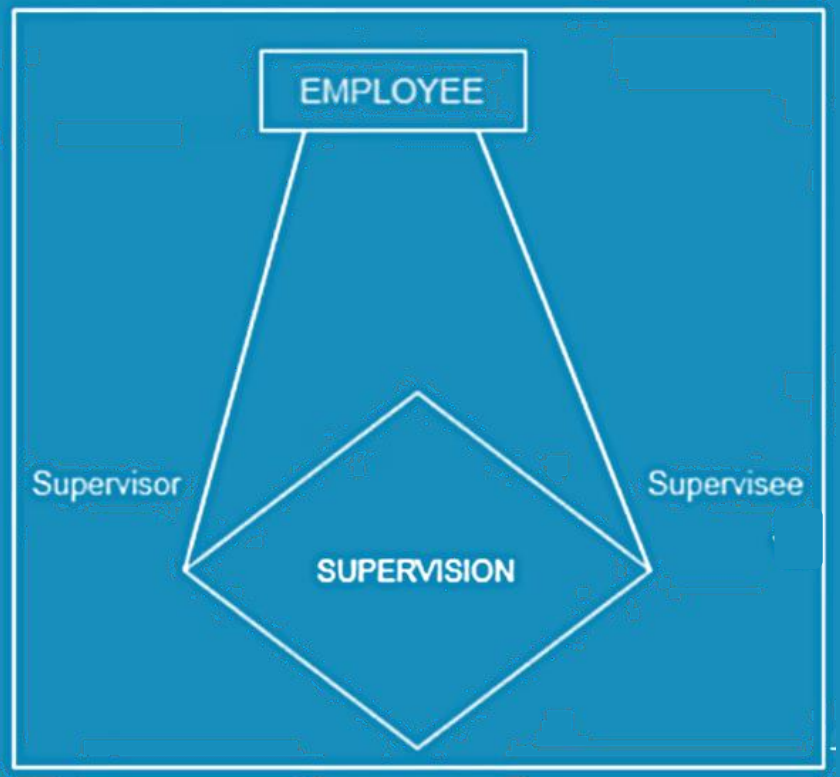
Relationship Degree, Role names and Recursive Relationship

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.



Relationship Degree, Role names and Recursive Relationship

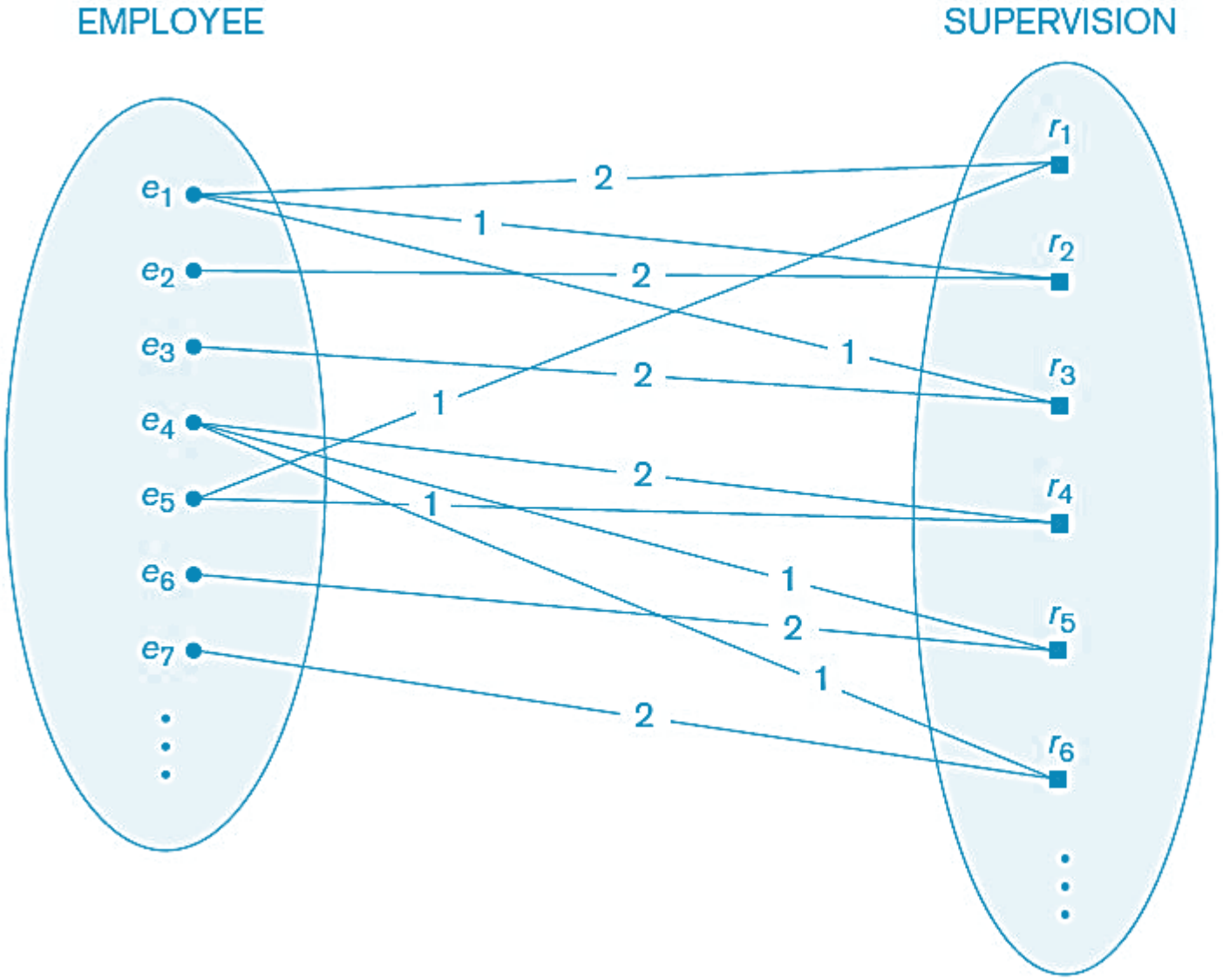


Figure 3.11
A recursive relationship SUPERVISION between EMPLOYEE in the *supervisor* role (1) and EMPLOYEE in the *subordinate* role (2).

Concept of weak Entity Types and partial keys

Strong Entity:

- Strong Entity is independent of any other entity in the schema

Example

A student entity can exist without needing any other entity in the schema or a course entity can exist without needing any other entity in the schema

- A Strong entity is nothing but an entity set having a key attribute or a table that consists of a primary key column

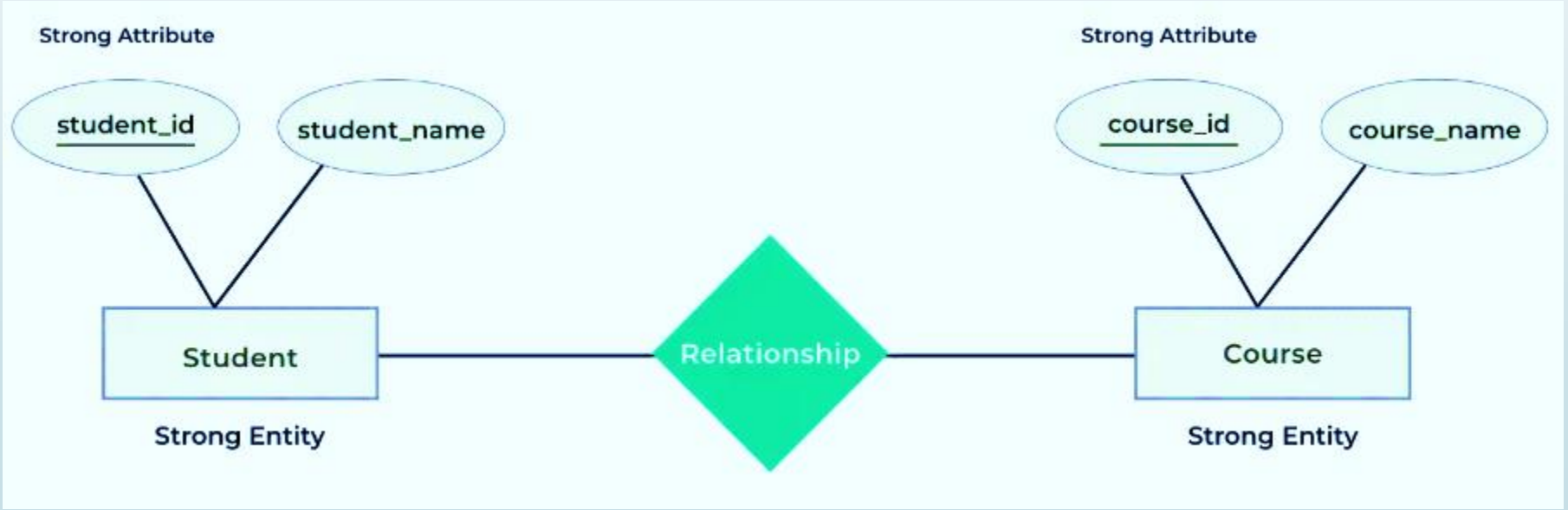
Representation

- The **strong entity** is represented by a **single rectangle**.
- **The relationship between two strong entities** is represented by a **single diamond.**

Concept of weak Entity Types and partial keys

Examples of Strong Entity:

- Consider the ER diagram
- **Student** entity is a strong entity because it consists of a key attributes called student id which is enough for accessing each record uniquely
- In the same way, the **course** entity contains of course ID attribute which is capable of uniquely accessing each row



Concept of weak Entity Types and partial keys

Weak Entity:

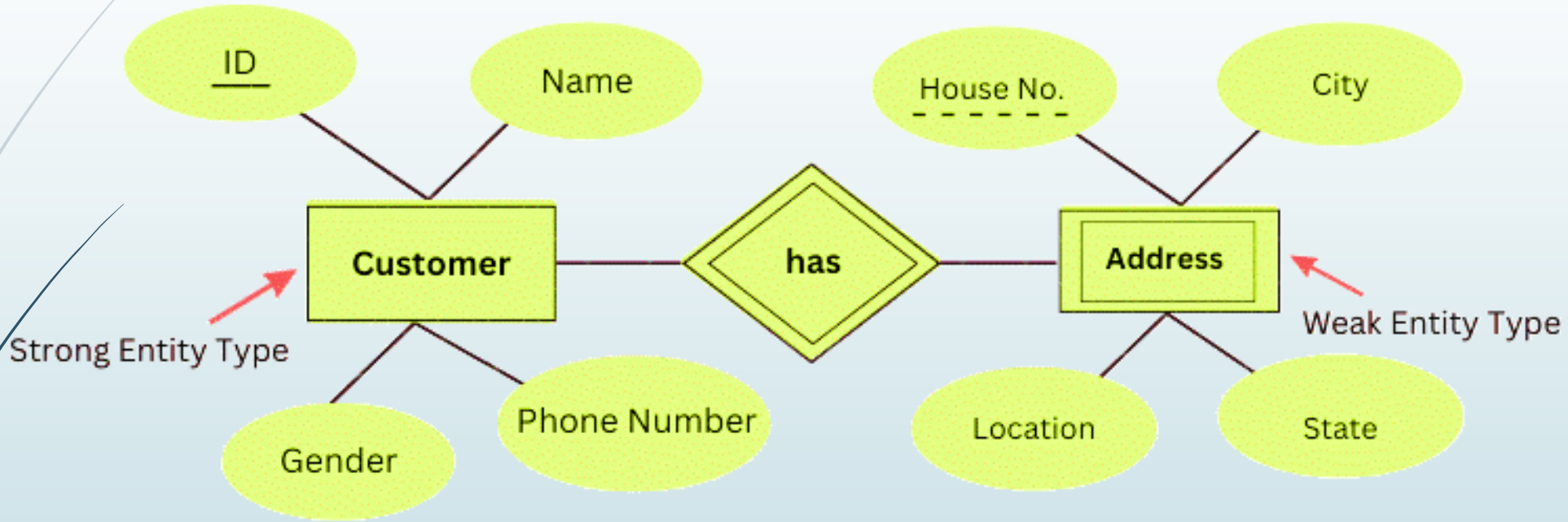
- An entity set that does not have sufficient attributes for Unique Identification of its records.
  - **Eg. 1:** A loan entity can not be created for a customer if the customer doesn't exist
  - **Eg. 2:** A dependents list entity can not be created if the employee doesn't exist
- Weak entity is nothing but an entity that does not have a key attribute
- It contains a partial key called a **discriminator which helps in identifying a group of entities from the entity set**
- A **discriminator** is represented by underlining with a **dashed line**

Representation

- A **double rectangle** is used for representing a **weak entity set**
- The **double diamond** symbol is used for representing the **relationship between a strong entity and a weak entity** which is known as **identifying relationship**

Concept of weak Entity Types and partial keys

Weak Entity:

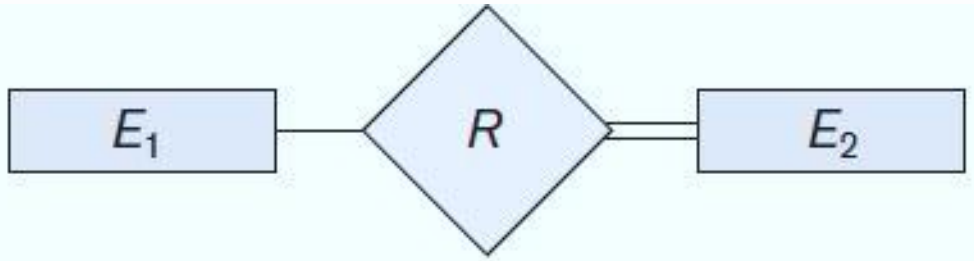


Entity Relationship Diagram between Strong Entity Type & Weak Entity Type

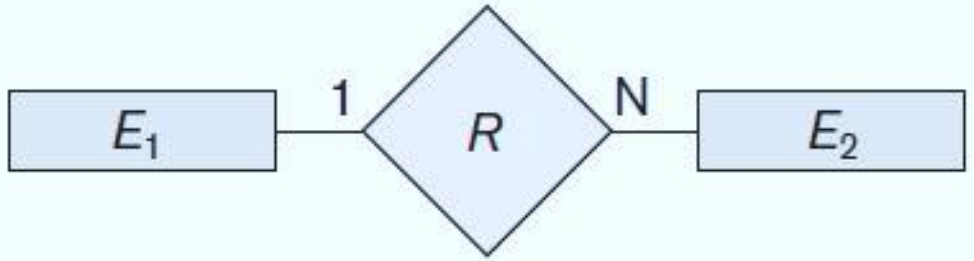
Notation for ER Diagram

Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Indentifying Relationship
	Attribute
	Key Attribute
	Multivalued Attribute
	Composite Attribute
	Derived Attribute

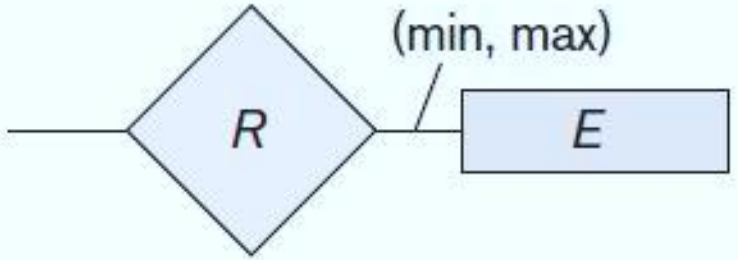
Notations Of ER Diagrams



Total Participation of E_2 in R



Cardinality Ratio 1: N for $E_1 : E_2$ in R



Structural Constraint (min, max) on Participation of E in R

ER Naming Conventions

- When designing a database schema, the choice of names for entity types, attributes, relationship types, and (particularly) roles is not always straightforward
- One should choose names that convey, as much as possible, the meanings attached to the different constructs in the schema
- We choose to use **singular names** for entity types, rather than plural ones
- we will use the convention that **entity type** and **relationship** type names are in **uppercase letters**,

ER Naming Conventions

- **Attribute** names have their **initial letter capitalized**, and role names are in lowercase letters.
- The ***nouns*** appearing in the narrative tend to give rise to **entity type** names, and the ***verbs*** tend to indicate names of **relationship types**.
- **Attribute names** generally arise from additional nouns that describe the nouns corresponding to entity types
- Choosing binary relationship names to make the ER diagram of the schema readable from left to right and from top to bottom.

ER Diagram: Design Issues

- Users often mislead the concept of the elements and the design process of the ER diagram.
- Thus, it leads to a complex structure of the ER diagram and certain issues that does not meet the characteristics of the real-world enterprise model.
- **The basic design issues of an ER database schema in the following points:**

1. **Use of Entity Set vs Attributes**
    - The use of an entity set or attribute depends on the structure of the real-world enterprise
    - It leads to a mistake when the user use the key attributes of an entity set as an attribute of another entity set.
    - Instead, he should use the relationship to do so

ER Diagram: Design Issues

2. Use of Entity Set vs Relationship Sets

- It is difficult to examine if an object can be best expressed by an entity set or relationship set.
- To understand and determine the right use, the user need to designate a relationship set for describing an action that occurs in-between the entities.
- If there is a requirement of representing the object as a relationship set, then its better not to mix it with the entity set.

ER Diagram: Design Issues

3. **Use of Binary vs n-ary Relationship Sets**
   - Relationships described in the databases are binary relationships
   - However, non-binary relationships can be represented by several binary relationships
   - It is possible to represent a non-binary relationship by a set of distinct binary relationships

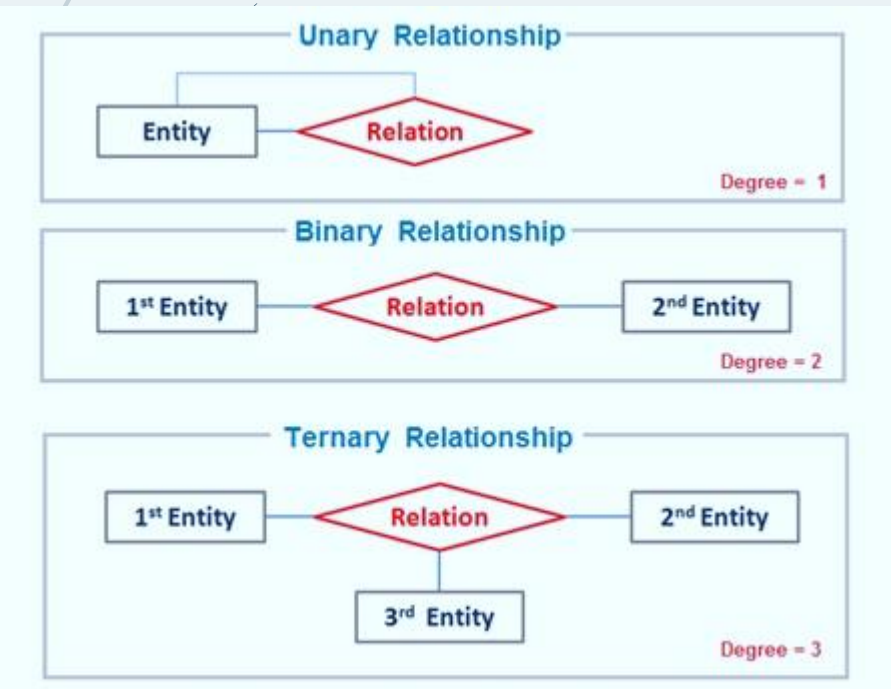
ER Diagram: Design Issues

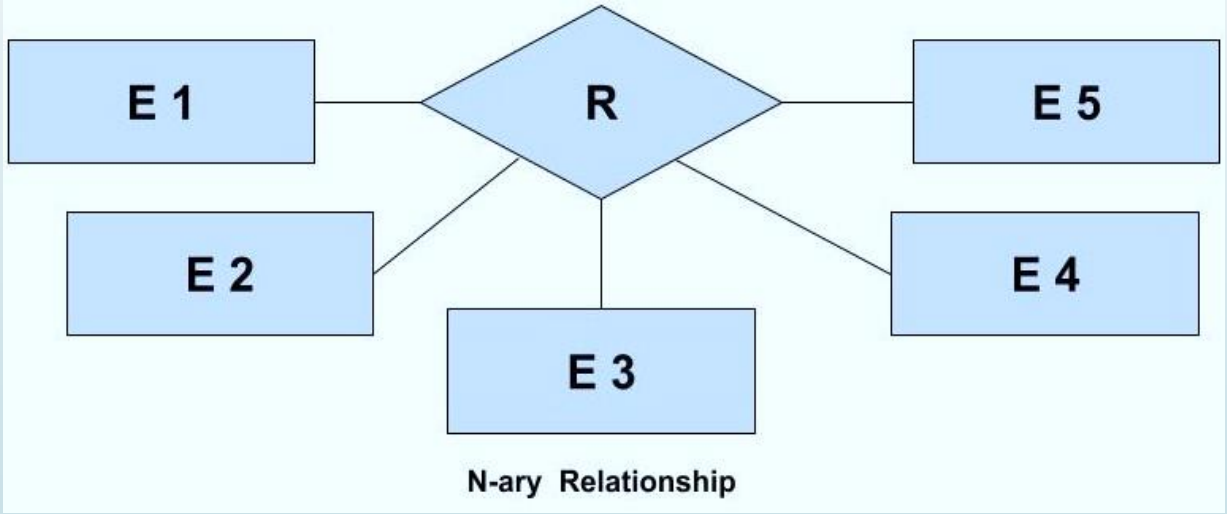
4. **Placing Relationship Attributes**
   - The cardinality ratios can become an affective measure in the placement of the relationship attributes.
   - So, it is better to associate the attributes of one-to-one or one-to-many relationship sets with any participating entity sets, instead of any relationship set.

Higher Degree Relationships

Degree of a Relationship Set

- Means number of entity set associated (Participated) in the relationship set
- Most of relationship sets in the ER diagram are binary.
- Occasionally however relationship sets involve more than two entity sets

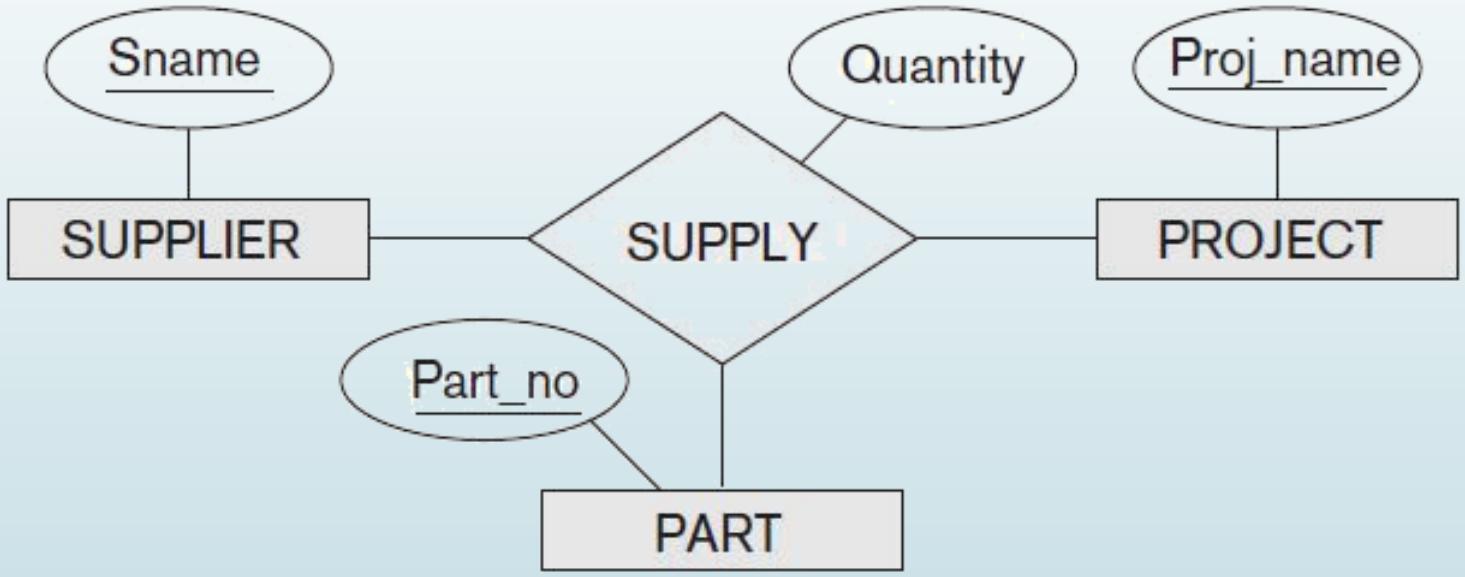




Higher Degree Relationships

Choosing between Binary and Ternary (or Higher-Degree) Relationships

- In general, a relationship type R of degree n will have n edges in an ER diagram, one connecting R to each participating entity type

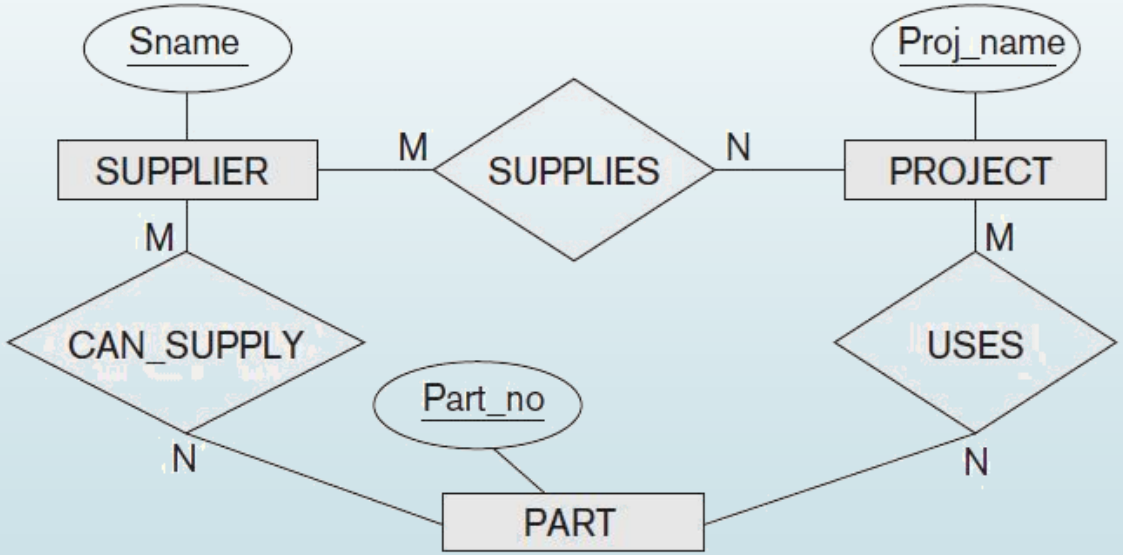


- The relationship set of SUPPLY is a set of relationship instances (s, j, p)
- **Meaning**: **s** is a SUPPLIER who is currently supplying a PART **p** to a PROJECT **j**

Higher Degree Relationships

Choosing between Binary and Ternary (or Higher-Degree) Relationships

- ER diagram for three binary relationship types **CAN_SUPPLY**, **USES**, and **SUPPLIES**
- In general, a ternary relationship type represents different information than do three binary relationship types



- The existence of three relationship instances (s, p), (j, p), and (s, j) in **CAN_SUPPLY**, **USES**, and **SUPPLIES**, respectively, does not necessarily imply that an instance (s, j, p) exists in the ternary relationship **SUPPLY**, because the meaning is different.

Higher Degree Relationships

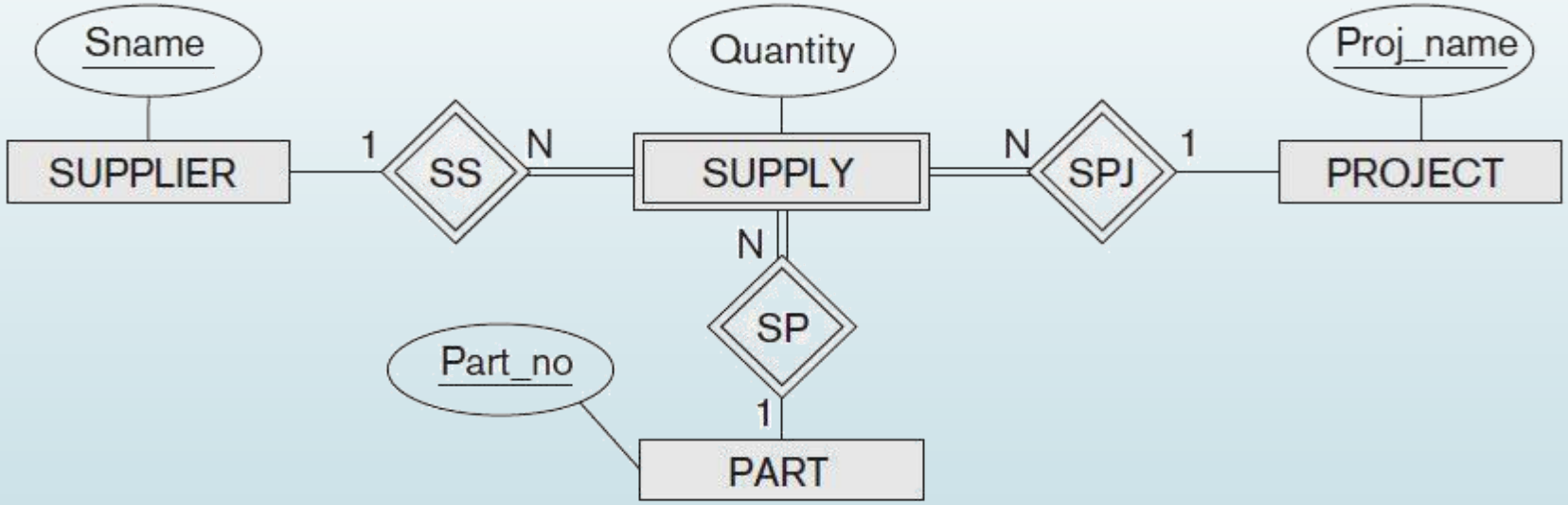
Choosing between Binary and Ternary (or Higher-Degree) Relationships

- It is often tricky to decide whether:
  - A particular relationship should be represented as a relationship type of degree **n**
  - Should be broken down into several relationship types of smaller degrees.
- The designer must base this decision on the semantics or meaning of the particular situation being represented
- The typical solution is to include the ternary relationship plus one or more of the binary relationships, if they represent different meanings and if all are needed by the application.

Higher Degree Relationships

Choosing between Binary and Ternary (or Higher-Degree) Relationships

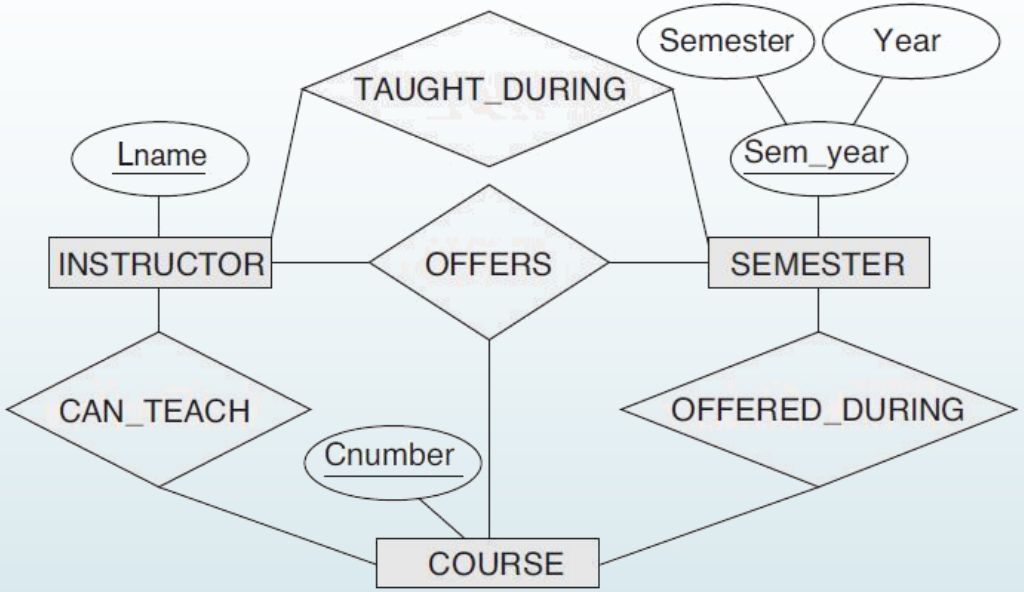
- Some database design tools permit only binary relationships.
- In this case, a ternary relationship such as **SUPPLY** must be represented as a weak entity type, with no partial key and with three identifying relationships



- It is also possible to represent the ternary relationship as a regular entity type
- In this example, a key attribute **Supply_id** could be used for the supply entity type, converting it into a regular entity type.

Higher Degree Relationships

Choosing between Binary and Ternary (or Higher-Degree) Relationships



- The ternary relationship type OFFERS represents information on instructors offering courses during particular semesters
- It includes a relationship instance (**i**, **s**, **c**) whenever INSTRUCTOR **i** offers COURSE **c** during SEMESTER **s**

Concept of Enhanced ER (EER) Model

- Today, with the increase in the production of data, the complexity associated with data has also increased multifold
- It becomes more and more difficult to use the **traditional ERDs** for database modeling.
- To reduce this complexity of modeling there arose a need to make improvements and enhancements to the existing **ERM**
- So that it is able to handle the complexity of modeling & application in a better way
- The requirements and complexity of complicated databases are represented using **enhanced entity-relationship diagrams (EERDs)**
- This is where the concept of **Enhanced ER Models** comes into the picture.

Concept of Enhanced ER (EER) Model

- EERDs are sophisticated database diagrams very similar to standard ER diagrams
- Hybrid of **ER models** with some additional complexities
- In addition to ER model concepts EE-R includes:
  1. **Subclasses** and **Super classes**
  2. **Specialization** and **Generalization**
  3. **Category** or **union type**
  4. **Aggregation**
- These concepts are used to create EE-R diagrams

Concept of Enhanced ER (EER) Model

ADDITIONAL FEATURES OF EER:

- EER creates a design more accurate to database schemas.
- It reflects the data properties and constraints more precisely.
- It includes all modeling concepts of the ER model.
- Diagrammatic technique helps for displaying the EER schema.
- It includes the concept of specialization and generalization.
- Used to represent a collection of objects that is union of objects of different of different entity types

Concept of Enhanced ER (EER) Model

SUBCLASS AND SUPERCLASS:

- Sub class and Super class relationship leads the concept of Inheritance.
- The relationship between sub class and super class is denoted with (d) symbol.

SuperClass (Supertype)

- Superclass is an entity type that has a relationship with one or more subtypes.
- High-level entity that can be further segmented into subclasses or subsets
- Also referred to as a Parent class
- For example: **Shape** superclass is having sub groups as **Square**, **Circle**, **Triangle**.

SubClass (Subtype)

- Subclass is a group of entities with unique attributes.
- A subclass can be referred to as a child or derived class
- Subclass inherits properties and attributes from its super class.
- For example: **Square**, **Circle**, **Triangle** are the subclass of **Shape** superclass.

Concept of Enhanced ER (EER) Model

SUBCLASS AND SUPERCLASS:

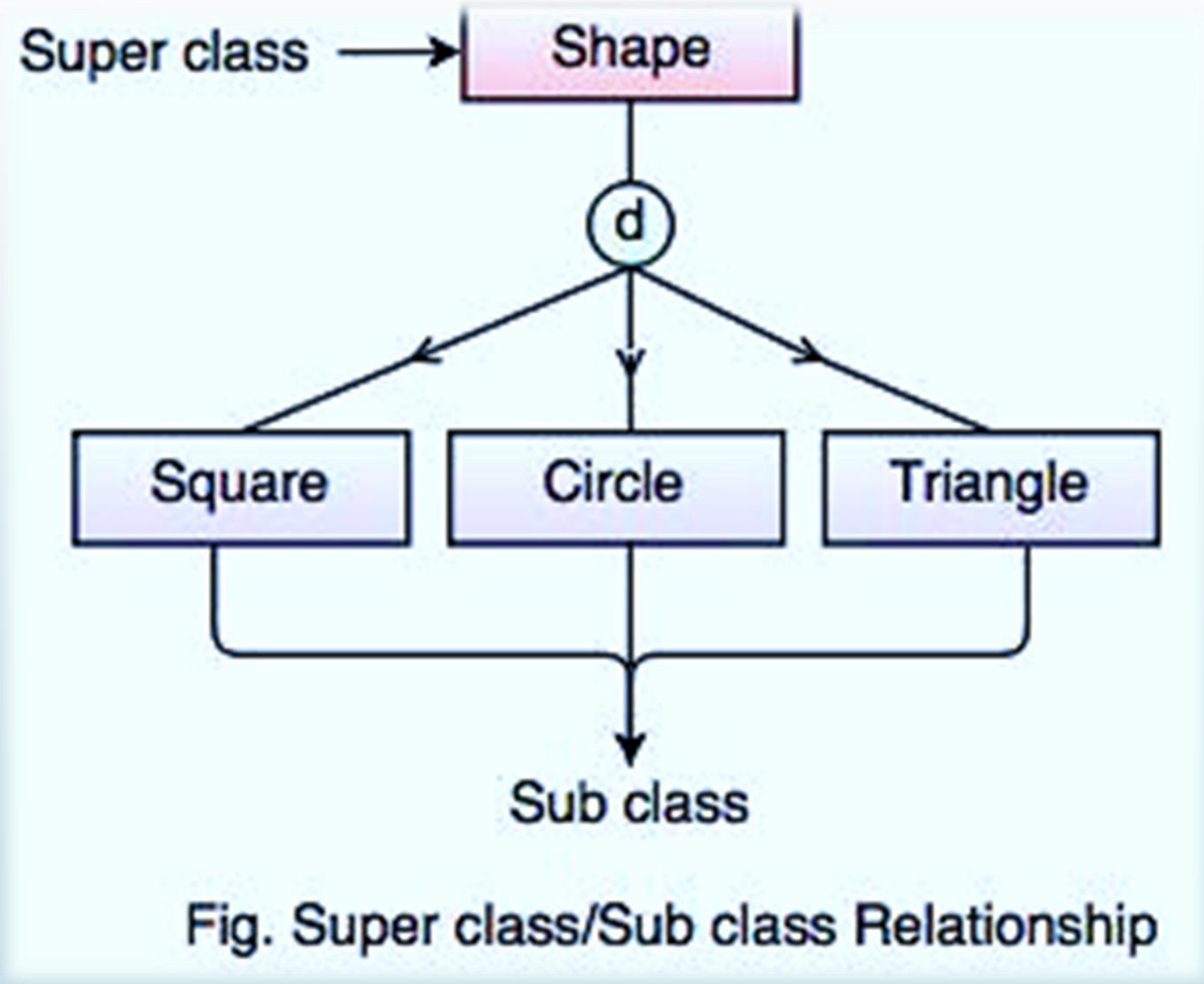


Fig. Super class/Sub class Relationship

SPECIALIZATION AND GENERALIZATION:

- Two normal kinds of relationships we added to the normal ER model for enhancement
- Inspired by the OOP, where we divide the code into classes and objects
- And in the same way, we have divided entities into subclass and superclasses
- Specialized classes are called subclasses
- Generalized classes are called superclasses or base classes

SPECIALIZATION AND GENERALIZATION:

- We can learn the concept of subclass by 'IS-A' analysis.
- **For example**, 'Laptop IS-A computer.' Or 'Clerk IS-A employee.'
- In this relationship, one entity is a subclass or superclass of another entity
- **For example**, in a university, a faculty member or clerk is a specialized class of employees
- So an employee is a generalized class, and all others are its subclass.

SPECIALIZATION AND GENERALIZATION:

GENERALIZATION :

- Process of generalizing the entities which contain the properties of all the generalized entities.
- Bottom up approach, in which two lower level entities combine to form a higher level entity.
- Reverse process of **Specialization**.
- It defines a general entity type from a set of specialized entity type.
- It minimizes the difference between the entities by identifying the common features

SPECIALIZATION AND GENERALIZATION:

GENERALIZATION :

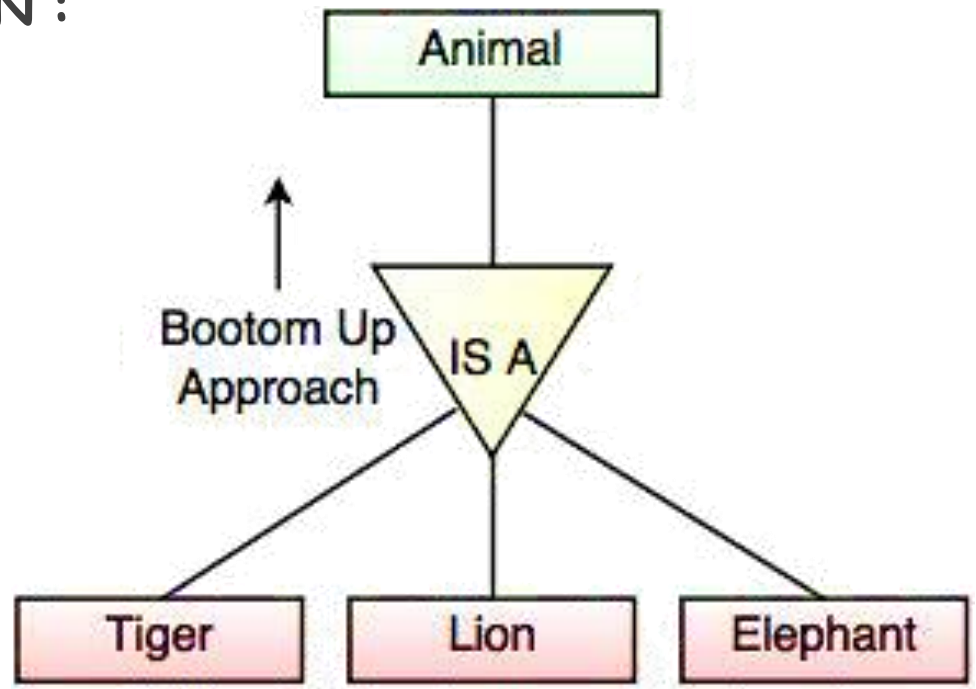


Fig. Generalization

In the above example, Tiger, Lion, Elephant can all be generalized as Animals

SPECIALIZATION AND GENERALIZATION:

SPECIALIZATION :

- Process that defines a group entities which is divided into sub groups based on their characteristic.
- Top down approach, in which one higher entity can be broken down into two lower level entity.
- It maximizes the difference between the members of an entity by identifying the unique characteristic or attributes of each member.
- It defines one or more sub class for the super class and also forms the superclass/subclass relationship.

SPECIALIZATION AND GENERALIZATION:

SPECIALIZATION :

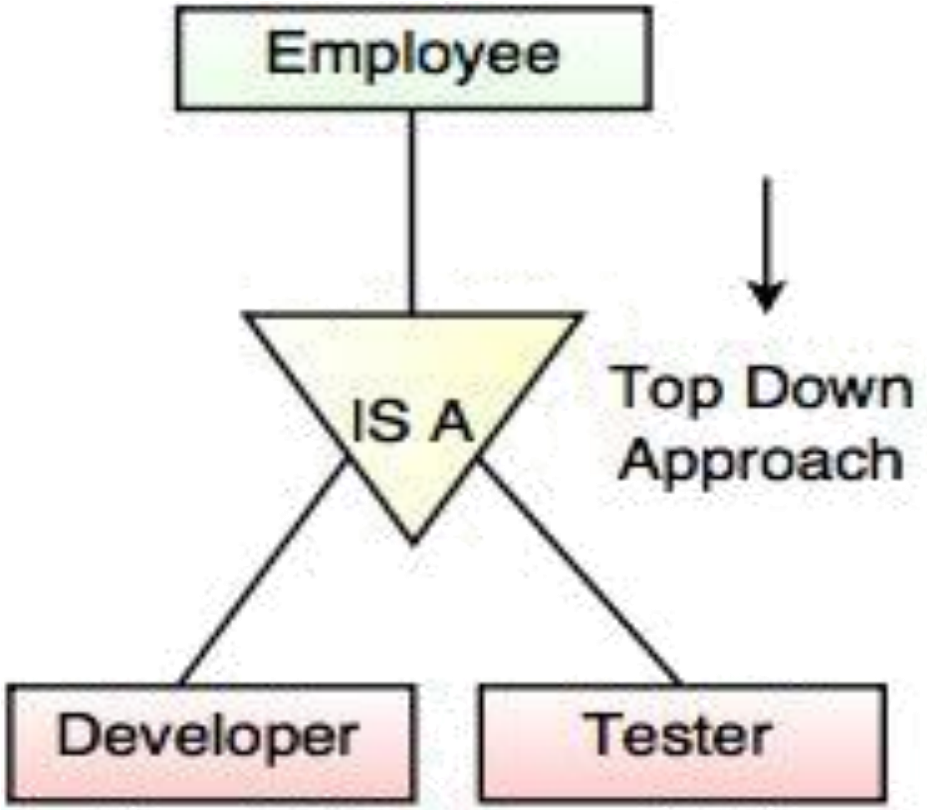


Fig. Specialization

Employee can be specialized as Developer or Tester, based on what role they play in an Organization

SPECIALIZATION AND GENERALIZATION:

CATEGORY OR UNION :

- Category represents a single super class or sub class relationship with more than one super class.
- It can be a total or partial participation.

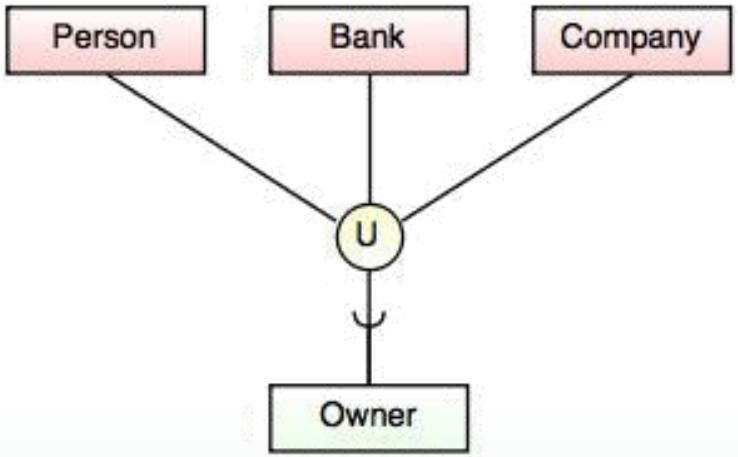


Fig. Categories (Union Type)

- For example **Car booking**
- Car owner can be a person, a bank (holds a possession on a Car) or a company.
- Category (sub class) → Owner is a subset of the union of the three super classes → Company, Bank, and Person.
- A Category member must exist in at least one of its super classes.

SPECIALIZATION AND GENERALIZATION:

AGGREGATION:

- Process that represent a relationship between a whole object and its component parts.
- It abstracts a relationship between objects and viewing the relationship as an object.
- It is a process when two entity is treated as a single entity

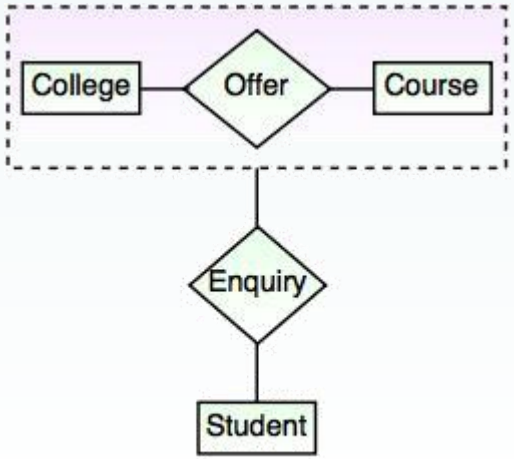


Fig. Aggregation

- In the above example, the relation between College and Course is acting as an Entity in Relation with Student

Creating an Effective EERD

- A well-designed EERD will help you build storage systems that are long-lasting and useful
- Consider the following when evaluating your ERD:
  1. **Stability**: Will the diagram support changing business needs?
  2. **Breadth**: Can all of the data that we need to store be organized in the model?
  3. **Flexibility**: Can data in this model be re-organized to support new information requirements?
  4. **Efficiency**: Is this model the simplest solution possible? Is the data modelled with the appropriate symbols?
  5. **Accessibility**: Can both creators and end users easily understand your EERD?
  6. **Conformity**: Will the model integrate easily with your existing database structure?