Overview of the Database Design Process

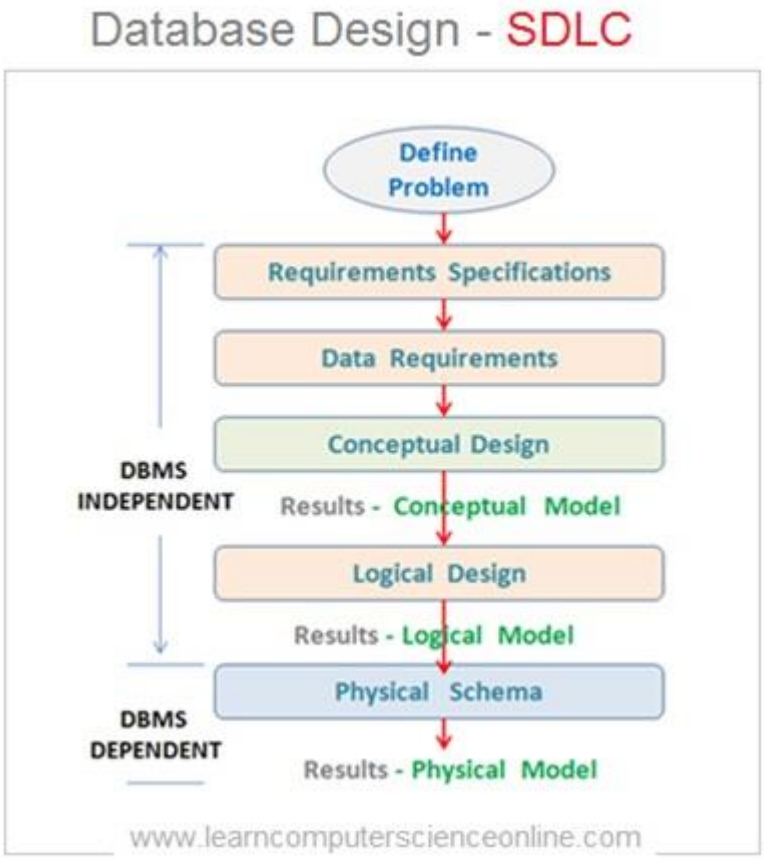
The **database design process** is a systematic approach used to **collect requirements**, **model data**, and **implement an efficient database system** that satisfies user needs while ensuring data integrity, consistency, and performance.

Objectives of Database Design

- To accurately represent real-world data
- To avoid data redundancy
- To ensure data integrity and consistency
- To support efficient data storage and retrieval

Phases of the Database Design Process

The database design process consists of **four major phases**:



---

1. Requirement Analysis

This is the **first and most important phase**.

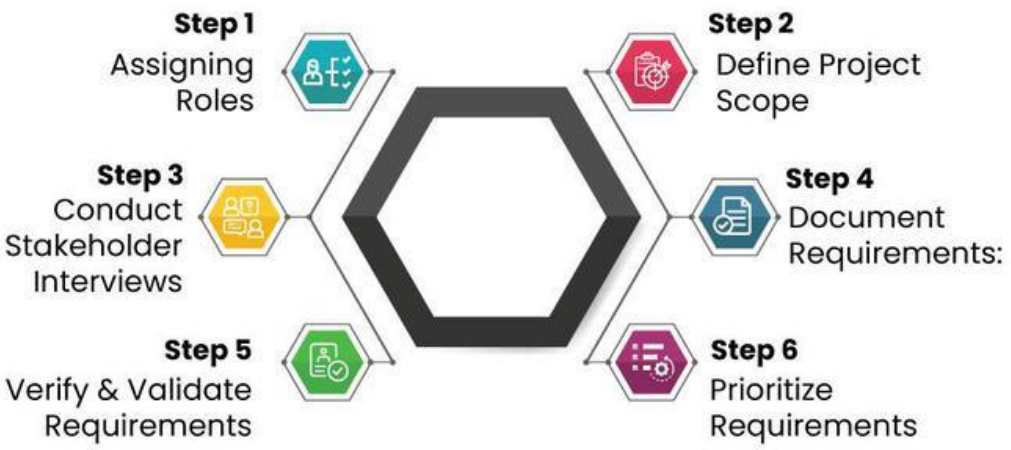
Activities:

- Identify user requirements
- Understand data to be stored
- Identify operations (queries, updates)
- Define constraints and rules

Outcome:

A clear **requirement specification document**

Example:
A university needs to store student details, course details, and enrollment information.



Processes of Requirements Gathering in Software Development

---

2. Conceptual Design

This phase creates a **high-level data model** independent of DBMS.

Key Tool:
✓ Entity–Relationship (ER) Model

Activities:

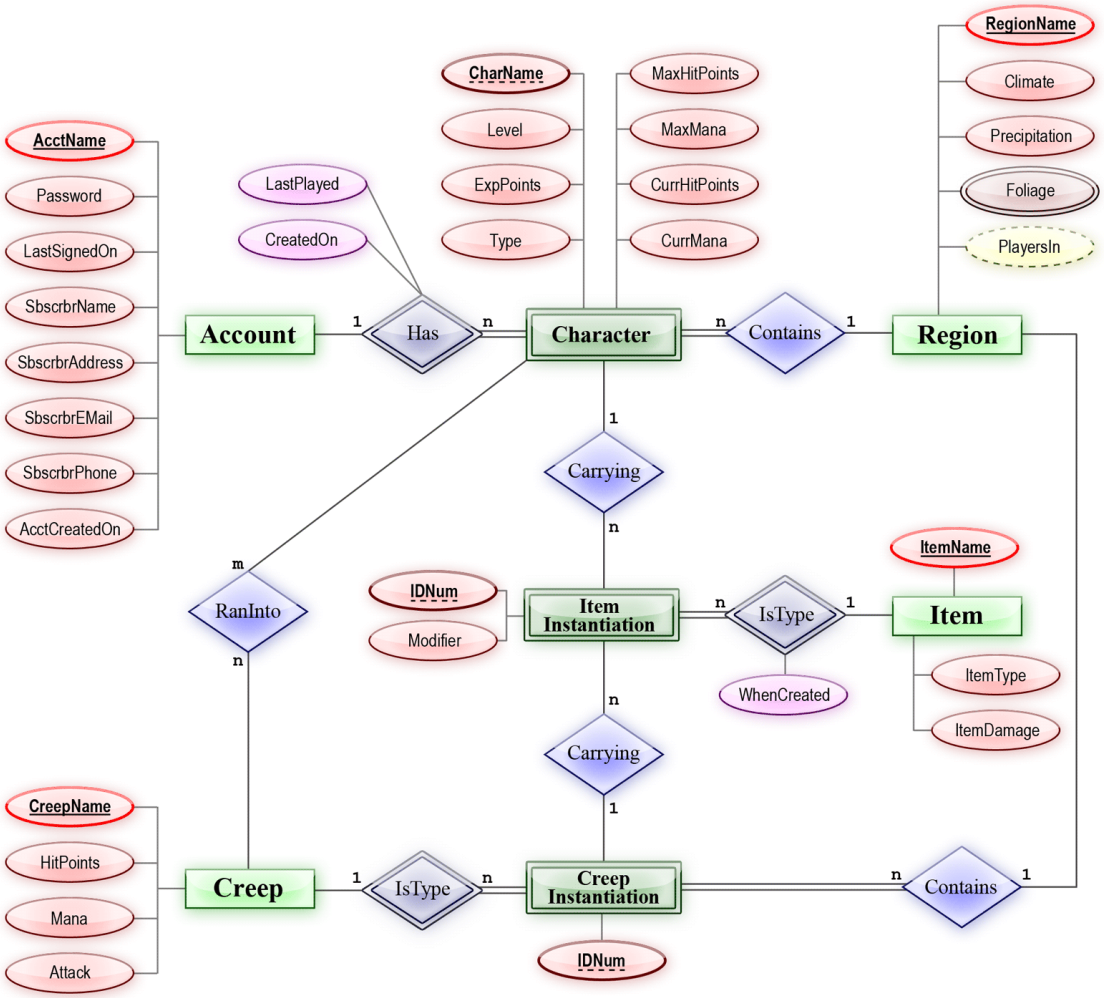
- Identify entities
- Identify relationships
- Assign attributes

- Define constraints

Outcome:
An **ER Diagram**

Advantages:

- Easy to understand
- Visual representation
- No implementation details



---

3. Logical Design

In this phase, the conceptual model is converted into a **logical schema**.

Activities:

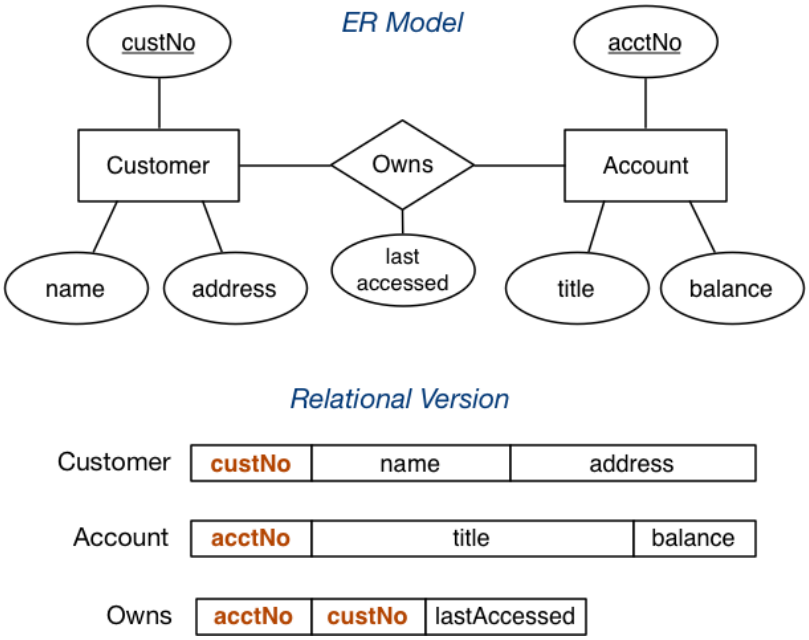
- Convert ER diagram into tables
- Define primary keys and foreign keys

- Apply normalization rules
- Ensure integrity constraints

Outcome:
Relational schema (tables)

Example:

- STUDENT(StudentID, Name, Age)
- COURSE(CourseID, Title)



---

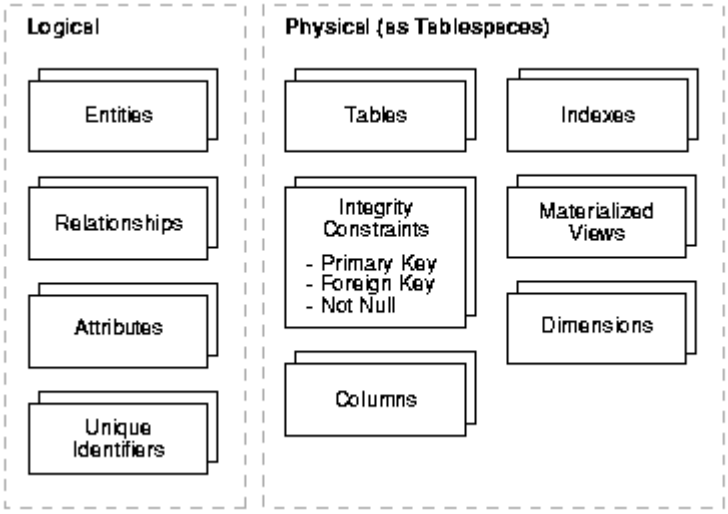
4. Physical Design

This phase deals with **how data is stored** in the system.

Activities:

- Decide file organization
- Indexing strategies
- Storage structures
- Access paths

Outcome:
Optimized database storage



Database Design Process Flow Diagram

```
User Requirements
        ↓
Requirement Analysis
        ↓
Conceptual Design (ER Model)
        ↓
Logical Design (Relational Schema)
        ↓
Physical Design (Storage & Indexing)
```

Roles Involved in Database Design

- Database Designer
- Database Administrator (DBA)
- End Users
- Application Developers

Advantages of Proper Database Design

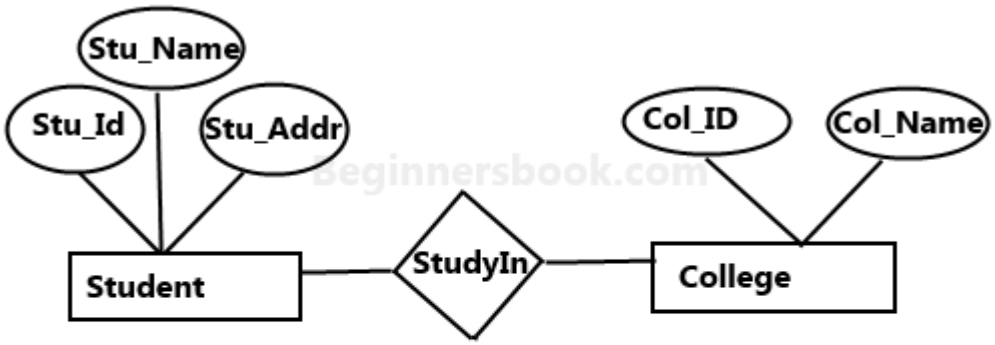
- Reduces redundancy
- Improves performance
- Enhances data integrity
- Simplifies maintenance
- Supports scalability

2. Entity–Relationship (ER) Model

The **ER model** is a **high-level conceptual data model** used to represent real-world data.

Components of ER Model

- Entities
- Attributes
- Relationships
- Constraints



Sample E-R Diagram

---

3. Entity and Entity Set

Entity

An entity is a **real-world object** that is distinguishable from other objects.

Examples:

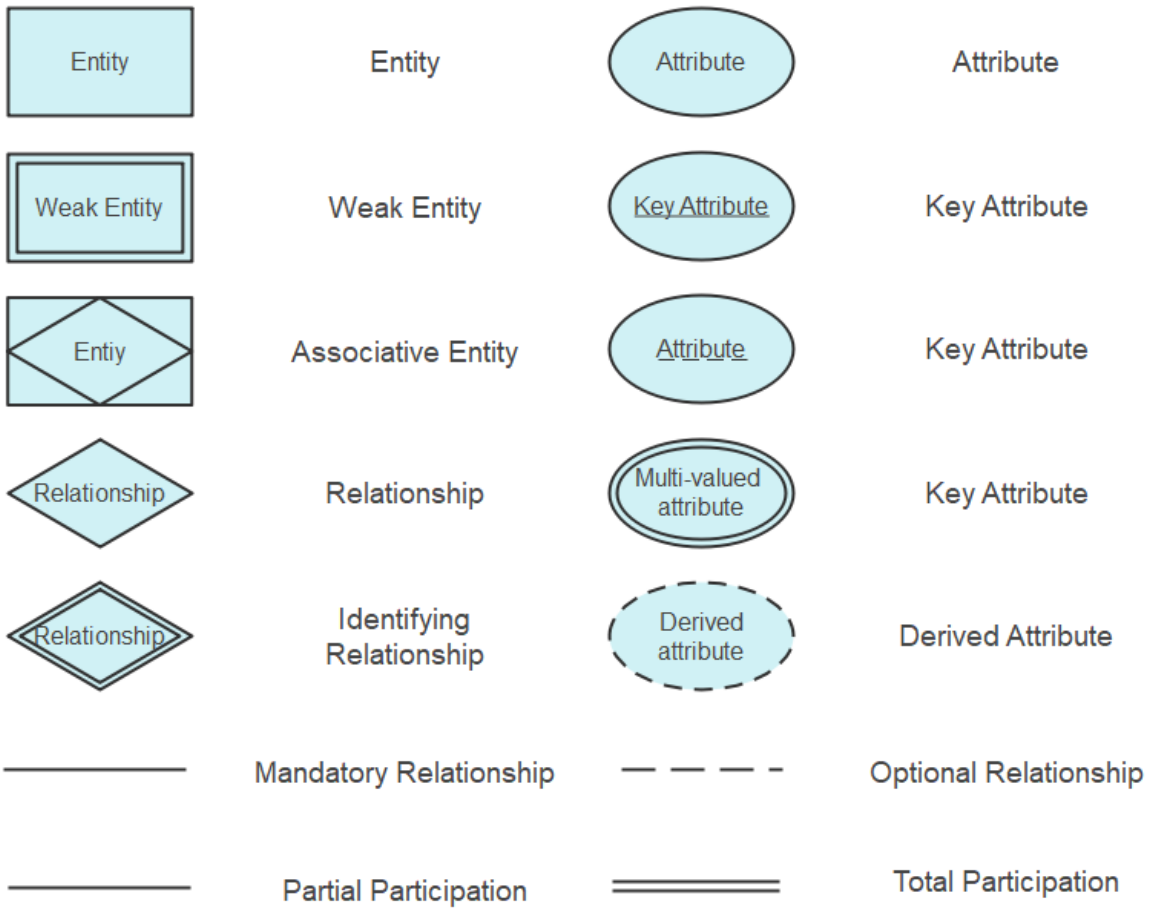
- Student
- Employee
- Book

Entity Set

A collection of similar entities.

Example:

- All students in a college



---

4. Types of Entities

Strong Entity

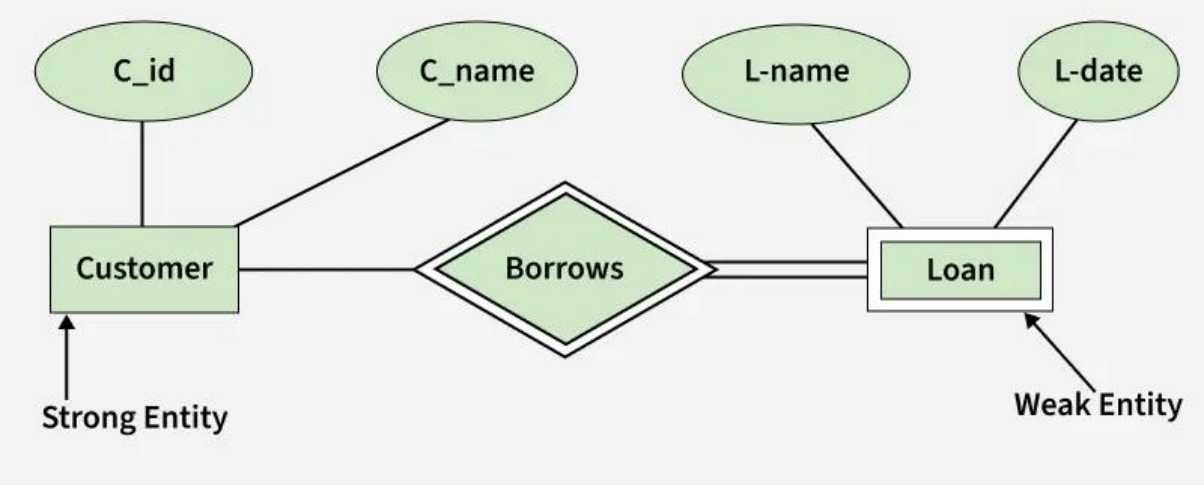
- Has its own primary key
- Exists independently

Example: Student (Student_ID)

Weak Entity

- Does not have a primary key
- Depends on a strong entity
- Uses **partial key**

Example:
Dependent depends on Employee



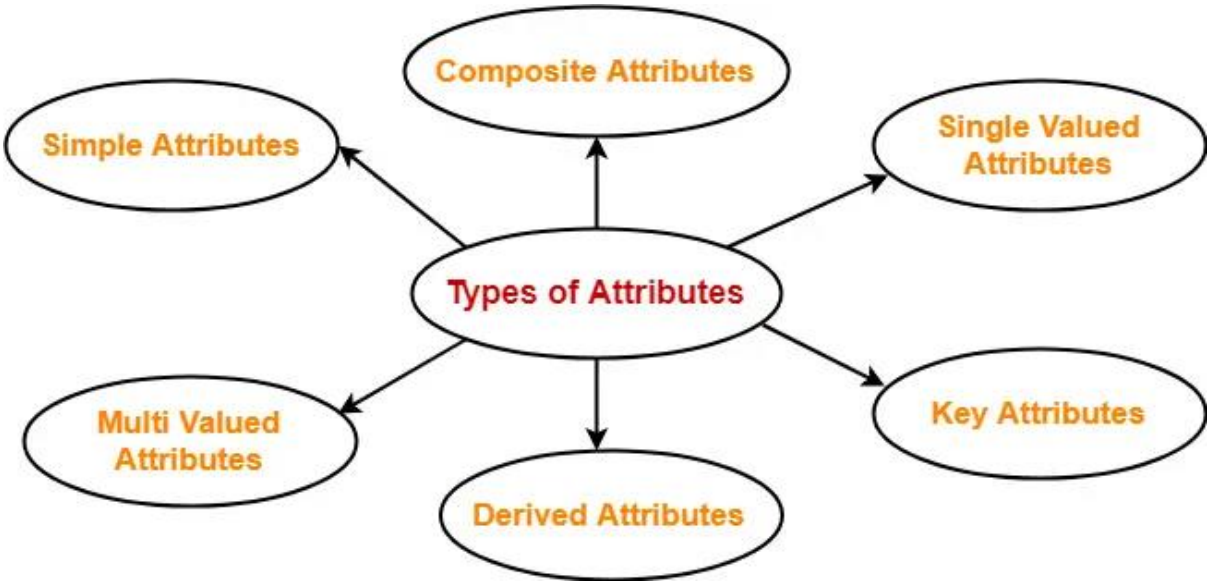
---

5. Attributes

Attributes describe the properties of entities.

Types of Attributes

1. **Simple Attribute** – Cannot be divided
   Example: Age
2. **Composite Attribute** – Can be divided
   Example: Address (Street, City, PIN)
3. **Single-Valued Attribute**
   Example: Roll number
4. **Multi-Valued Attribute**
   Example: Phone numbers
5. **Derived Attribute**
   Example: Age (derived from Date of Birth)
6. **Key Attribute**
   Example: Student_ID



---

6. Relationship

A relationship represents an **association among entities**.

Relationship Set

Collection of similar relationships.

Example:
STUDENT — ENROLLS — COURSE

---

7. Degree of Relationship

Unary Relationship

Relationship within the same entity
Example: Employee supervises Employee

Binary Relationship

Between two entities
Example: Student enrolls Course

Ternary Relationship

Between three entities
Example: Supplier supplies Part to Project

---

8. Constraints in ER Model

1. Key Constraint

- Specifies uniqueness of entity

2. Cardinality Constraint

Defines number of entities participating.

Type	Example
One-to-One	Person–Passport
One-to-Many	Department–Employee
Many-to-One	Employee–Department
Many-to-Many	Student–Course

3. Participation Constraint

- **Total Participation** – Mandatory
- **Partial Participation** – Optional

---

9. Entity–Relationship Diagrams (ER Diagrams)

Symbols Used

- Rectangle → Entity
- Ellipse → Attribute
- Diamond → Relationship
- Double Rectangle → Weak Entity
- Double Diamond → Identifying Relationship

Purpose

- Visual representation of database
- Helps in easy understanding
- Used before implementation

---

10. Weak Entity Sets

Characteristics

- Cannot exist independently
- Identified using:
    - Strong entity key
    - Partial key

Example

Employee (Emp_ID)
Dependent (Dep_Name)

---

11. Special ER Concepts

Generalization

- Bottom-up approach
- Combine lower-level entities into higher-level

Specialization

- Top-down approach
- Divide entity into subclasses

Inheritance

- Sub-entities inherit attributes from super-entity

Aggregation

- Treat relationship as an entity

---

12. ER Design Issues

Introduction

ER design issues arise while converting real-world requirements into an **Entity–Relationship (ER) model**.
The main objective is to represent data **accurately**, **without redundancy**, and **with correct semantics**.

---

1. Entity vs Attribute

Problem

Deciding whether a real-world concept should be modeled as an **entity** or an **attribute**.

Explanation

- An **attribute** describes a property of an entity and has no independent existence.
- An **entity** represents an object that can exist independently and may have its own attributes.

Guidelines

- Choose **attribute** if:
  - It has a single value
  - It does not participate in relationships
- Choose **entity** if:
  - It has multiple attributes
  - It participates in relationships
  - It needs to be referenced separately

Example

- Age → Attribute
- Address → Entity (Street, City, Pincode)

---

2. Entity vs Relationship

Problem

Whether a concept should be represented as an **entity** or a **relationship**.

Explanation

- A **relationship** represents an interaction between entities.
- An **entity** represents an object with independent meaning.

Guidelines

- Use **relationship** if the concept describes interaction.
- Convert relationship into **entity** if it has its own attributes.

Example

- Enrollment between Student and Course
    - Without attributes → Relationship
    - With attributes (Grade, Date) → Entity

---

3. Binary vs Ternary Relationship

Problem

Whether to represent relationships using **binary** or **ternary** relationships.

Explanation

- Binary relationships involve two entities.
- Ternary relationships involve three entities simultaneously.
- Converting ternary into binary may **change the meaning**.

Guidelines

- Use **binary** when semantics remain unchanged.
- Use **ternary** when interaction depends on all three entities.

Example

Supplier supplies Part to Project
(Binary conversion loses meaning)

---

4. Placement of Attributes

Problem

Deciding where to place an attribute: **entity or relationship**.

Explanation

- Attributes should be placed where they **logically belong**.
- Incorrect placement leads to redundancy or incorrect data modeling.

Guidelines

- Place attribute in **entity** if it describes the entity itself.
- Place attribute in **relationship** if it describes the interaction.

Example

- Salary → Employee entity
- Hours Worked → Employee–Project relationship

---

5. Use of Weak Entity

Problem

Identifying whether an entity should be modeled as a **weak entity**.

Explanation

- A weak entity cannot be uniquely identified without a strong entity.
- It depends on a strong entity for existence.

Guidelines

- Use weak entity if:
  - It has no primary key
  - It uses a partial key
  - Its existence depends on another entity

Example

- Dependent depends on Employee

---

6. Generalization vs Specialization

Problem

How to model class hierarchies correctly.

Explanation

- **Generalization** is a bottom-up approach.
- **Specialization** is a top-down approach.
- Inheritance allows subclasses to share attributes.

Guidelines

- Use specialization when subclasses have unique attributes.
- Use generalization when multiple entities share common features.

Example

Employee → Teacher, Clerk

---

7. Redundancy Avoidance

Problem

Duplicate data stored in multiple places.

Explanation

- Redundancy causes inconsistency and increased storage.
- ER design must minimize redundancy.

Guidelines

- Store data once.
- Use relationships instead of repeating attributes.

Example

Department name should not be repeated for every employee.

---

8. Constraint Representation

Problem

Representing constraints correctly in ER model.

Explanation

- Constraints ensure data integrity.
- Incorrect constraints lead to invalid database states.

Types

- Cardinality constraints
- Participation constraints
- Key constraints

Conclusion

ER design issues focus on **correct modeling decisions**.
Proper handling of these issues results in:

- Clear ER diagrams
- Reduced redundancy
- Accurate database design
- Efficient implementation