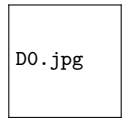
Lecture 06: Extended ER Model

ISA Hierarchies and Aggregation

Z2004: Database Management Systems



IIT Madras Zanzibar

Tuesday, 10 March 2026

Recap: ER Building Blocks

From L04 and L05:

- **Entity** — object with independent existence (rectangle)
- **Attribute** — property of an entity (ellipse)
- **Relationship** — association between entities (diamond)
- **Cardinality:** 1:1, 1:N, M:N
- **Participation:** total (double line), partial (single line)
- **Weak Entity** — no key of its own; depends on owner (double rectangle + double diamond)

Today we extend the ER model:

1. **ISA hierarchies** (specialization / generalization)
2. **Aggregation** — treating a relationship as an entity

Why Extend?

Real-world domains often have subtype relationships *"every employee is a person"* and complex associations (*"a contract links a project to a sponsor via a manager"*). Basic ER cannot express these cleanly.

ISA Hierarchies: Superclasses & Subclasses

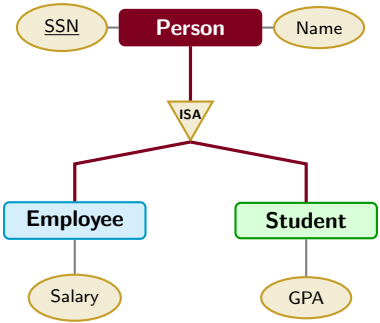
1. Specialisation (Top-Down)

A process where we identify subgroups of an entity that have distinct attributes or participate in distinct relationships.

- **Subclass:** The specialized subgroup e.g., *Employee*.
- **Superclass:** The parent entity e.g., *Person*.
- **Notation:** A triangle labelled `ISA`.

2. Generalisation (Bottom-Up)

The reverse process. We recognize that several entity sets share common attributes, so we abstract them up into a shared **Superclass**.



Every `Employee` IS A `Person`.

Every `Student` IS A `Person`.

ISA: Constraints You Must Specify

Constraints to specify:

1. Membership constraint (how assigned)
Condition-defined: Follows a predicate
e.g. $salary > 0 \Rightarrow$ *Employee*

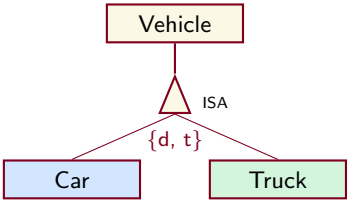
User-defined: DBA/app explicitly assigns entity

2. Disjointness constraint
Disjoint (d): Entity belongs to *at most one* subclass

Overlapping (o): May belong to *multiple* subclasses simultaneously

3. Completeness constraint
Total: Every superclass entity must be in some subclass

Partial: Some superclass entities may belong to no subclass



disjoint + total:

every vehicle is exactly one of Car or Truck

Common Mistake

Forgetting to state disjointness and completeness. Both must be explicitly specified in your ER diagram or description.

Multi-Level ISA Hierarchies

Inheritance in ISA

A subclass **inherits** all attributes and relationships of its superclass.

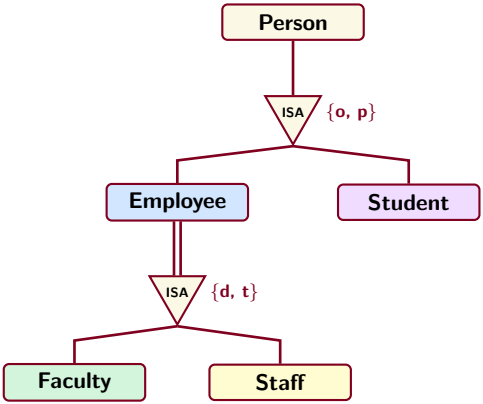
Subclasses add their own **specialised attributes** and relationships.

Practical example: University staff hierarchy

- `Person` → `Employee` → `Faculty` or `Staff`
- `Faculty` has attributes: `rank`, `department`
- `Staff` has attributes: `grade`, `supervisor`
- Both inherit: `SSN`, `name`, `salary` from `Person`/`Employee`

Use ISA when:

- Subgroups have **distinct attributes**, OR
- Subgroups participate in **distinct relationships**

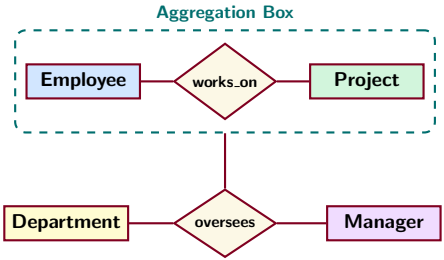


Aggregation: Relationships on Relationships

The Problem Aggregation Solves

Scenario: An employee *works on* a project. The company hires a *manager* to *oversee* that work arrangement.

Problem: In basic ER, a relationship (*works_on*) cannot participate in another relationship (*oversees*).



Aggregation

Aggregation treats a **relationship set as an entity set**, so it can participate in further relationships.

Notation: a **dashed rectangle** (box) around the entire relationship set + its participating entities.

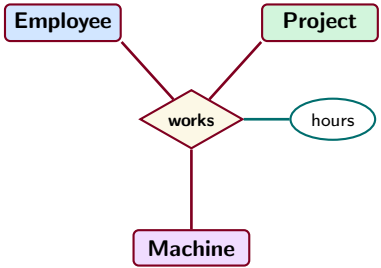
When to Use Aggregation vs. Ternary Relationships

Ternary Relationship

Three entities participate in one relationship set simultaneously.

Example: Employee works-on a Project using a Machine.

- One diamond, three connecting lines
- Attributes of the relationship apply to the **combination** of all three



Ternary Relationship

Aggregation

Use when you need to **attach attributes or relationships to an existing relationship**.

Example: oversee the (Employee, Project) work arrangement with a Manager and a budget.

Rule of Thumb

If the relationship itself needs to participate in another relationship → **use aggregation**.

If three entities participate simultaneously → **use ternary**.

Design Choices: ISA and Aggregation

ISA Design Guidelines

- Use ISA only when subclasses have **distinct attributes or relationships**
- A *"type"* attribute is NOT sufficient reason for ISA
- Prefer fewer levels (avoid deep hierarchies)
- State both constraints explicitly: {d/o, t/p}
- Consider whether partial vs. total coverage makes semantic sense

Aggregation Guidelines

- Use when a relationship needs to be *an entity* for further modelling
- Check: could a ternary relationship work instead? (often simpler)
- Aggregation adds complexity — use only when necessary

Common Design Errors

1. Creating ISA for `MaleEmployee` / `FemaleEmployee` (gender is just an attribute!)
2. Using a ternary when one entity is not always present
3. Nesting aggregations (rarely justified)
4. Forgetting ISA constraints in the ER diagram
5. Redundant attributes in subclasses (already inherited from superclass)

Design Choices: ISA (Specialization)

ISA Design Guidelines

- Use **ISA** only when subclasses have **distinct attributes** or **distinct relationships**.
- A *"type"* attribute is NOT a sufficient reason for ISA.
- **Keep it shallow:** Prefer fewer levels to avoid deep hierarchies.
- **Be Explicit:** State both constraints: $\{d/o, t/p\}$.
- Consider whether partial vs. total coverage makes semantic sense.

Key Principle Subclasses should represent real-world entities with unique behavior, not just data categories.

Design Choices: Aggregation & Pitfalls

Aggregation Guidelines

- Use when a relationship needs to be *an entity* for further modelling.
- **Ternary Check:** Could a ternary relationship work instead?
- Aggregation adds complexity - use only when necessary.

Common Design Errors

1. **Over-engineering:** ISA for `Male`/`Female` (Gender is an attribute).
2. **Ternary misuse:** Using ternary when one entity is not always present.
3. **Nesting:** Nesting aggregations (rarely justified).
4. **Missing Labels:** Forgetting ISA constraints $\{d, t\}$ in diagrams.
5. **Redundancy:** Redefining superclass attributes in subclasses.

Activity (4 min)

A university database has `Person` as superclass.

Subclasses: `Student` (with `GPA`), `Instructor` (with `salary`), `AdminStaff` (with `grade`).

Questions:

1. Is this hierarchy disjoint or overlapping? (Can someone be both Student and Instructor?)
2. Is it total or partial? (Can a Person belong to no subclass?)
3. An `Instructor` *teaches* a `Course`. A `Department` *evaluates* that teaching assignment. Which ER construct do you need: ternary relationship or aggregation?

Activity Solution

1. **Overlapping (o)**
   Why? A graduate student e.g., a Teaching Assistant can be both a `Student` and an `Instructor` at the same time.

2. **Partial (p)**
   Why? A `Person` in the database could be an alumnus, a guest speaker, or an external contractor; meaning they might not belong to any of those three specific subclasses.

3. **Aggregation**
   Why? The `Department` is not evaluating the `Instructor` in isolation, nor the `Course` in isolation. It is evaluating the *relationship* itself (the teaching assignment). Aggregation treats the *teaches* relationship as an entity that can participate in the *evaluates* relationship.

Lecture Summary

Key Takeaways

1. **ISA (specialisation/generalisation)** models entity subtypes using an ISA triangle.
2. Subclasses inherit all attributes and relationships from superclass.
3. Specify **disjointness (d/o)** and **completeness (t/p)** explicitly.
4. **Aggregation** treats a relationship as an entity, enabling relationships on relationships.
5. Use aggregation when a relationship needs to participate in further relationships.

Next Lecture (L07)

Relational Model Fundamentals

- Formal definition of a relation
- Tuples, domains, schemas, instances
- Superkey, candidate key, primary key
- Relational vs. ER model

Reading

Silberschatz, **Ch. 7**: Databse Design and the E-R Model.