Specialisation and Generalisation

Table of contents

Classes, superclasses and subclasses 1
Two directions, one relationship 2
Constraints on a specialisation 2
One superclass, several independent specialisations 3
Subclasses carry their own relationships 4
What happens on mapping 4

The Enhanced ER (EER) model's one big addition to plain ER: the ability to say that one entity type is a *kind of* another. Introduced in 2026-03-02-weak-entities-the-eer-model-and-design-choices[1]; see er-diagram-notation[2] for the notation.

Classes, superclasses and subclasses

In the EER model an entity type is called a **class**, and entities in the same class share the same attributes. A class can be a **superclass** or a **subclass** of another:

- Attributes of a superclass are **inherited** by its subclasses, and a subclass can add its own specific attributes *and its own specific relationships* on top.
- Every entity in a subclass is also a member of its superclass(es). The subclass never holds entities the superclass doesn't.

The motivating example is a supermarket `ITEM` (`ProductName`, `Price`) versus a `FOOD` item, which is just an `ITEM` that additionally records an `ExpiryDate`. Modelling `FOOD` as a subclass says exactly that — it is an extension of a regular item, not a separate thing that happens to look similar.

[1] courses/infs1200/2026-03-02-weak-entities-the-eer-model-and-design-choices.pdf
[2] courses/infs1200/er-diagram-notation.pdf

Two directions, one relationship

Specialisation and **generalisation** describe the same superclass/ subclass structure discovered from opposite ends:

	Direction	Starting point	Move
Specialisation	Top-down	One existing entity type	Define a number of subclasses of it, each containing a subset of the superclass's entities, on the basis of a more specific *distinguishing characteristic*
Generalisation	Bottom-up	Several existing entity types	**Abstraction** — ignore the differences amongst the subclasses and generalise them into a common superclass

The two are duals, not alternatives: the subclasses are a specialisation of the superclass, and the superclass is a generalisation of the subclasses. Which word you use just records how you arrived at the diagram.

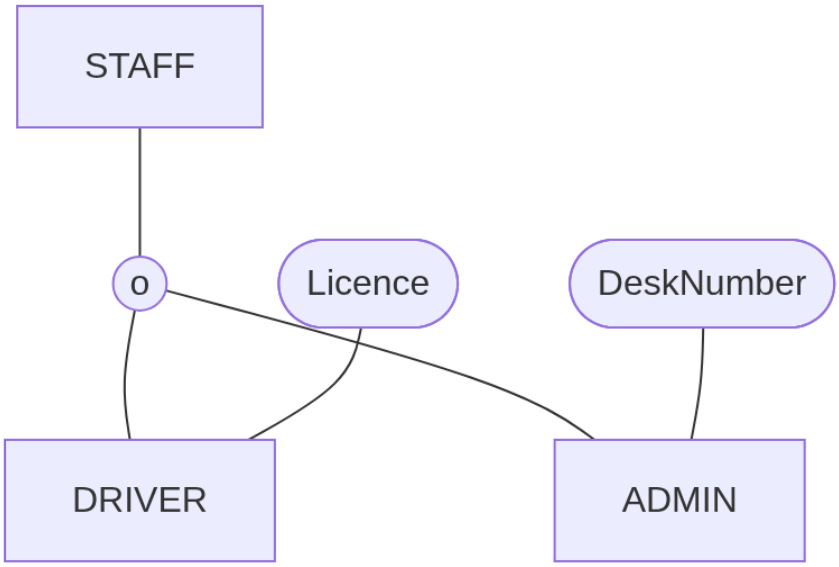
Constraints on a specialisation

Two independent constraints apply to every specialisation, and both come from the UoD rather than from preference. They are stated over different things: *a specialisation* may be total or partial, while *the subclass sets* may be overlapping or disjoint.

Constraint	Applies to	Options	Question it answers
Completeness	The specialisation	**Total** (double line) vs **partial** (single line)	Must every superclass entity belong to *some* subclass?
Disjointness	The subclass sets	**Disjoint** (`d`) vs **overlapping** (`o`)	Can one superclass entity belong to more than one subclass at once?

Worked readings from the tutorials:

- `DOCTOR` specialises **disjointly** into `GP` / `SPECIALIST` — "each doctor is either a GP or a specialist, never both".
- `PARTICIPANT` specialises into **overlapping** `HEALTH` / `AGE` subclasses — a participant can legitimately be in both at once.
- `STAFF` specialises into **overlapping** `DRIVER` / `ADMIN` in week2-tutorial-case-study-1-dirt-road-driving[3], because a staff member at a small rideshare company can be both at once. Getting this one wrong as disjoint would silently forbid a perfectly ordinary real employee.



One superclass, several independent specialisations

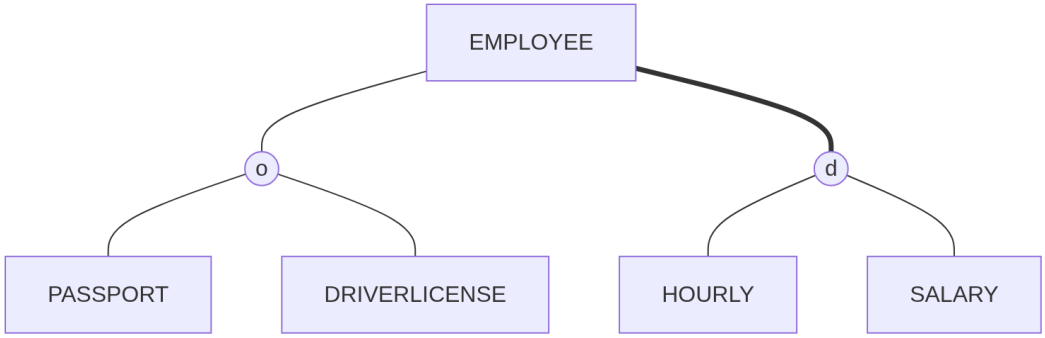
A superclass can be specialised more than once, along different characteristics, each with its own circle and its own **independent** pair of constraints. The canonical case is `EMPLOYEE` carrying two specialisations side by side:

Specialisation	Subclasses	Disjointness	Completeness
Documents held	`PASSPORT` / `DRIVERLICENSE`	**Overlapping** (`o`)	**Partial** (single line)
How they are paid	`HOURLY` / `SALARY`	**Disjoint** (`d`)	**Total** (double line)

All four boxes are ordinary subclasses of `EMPLOYEE` — none of them is a category or a union type, and each is drawn as a plain rectangle. The `o` on the left circle is what permits an

[3] courses/infs1200/week2-tutorial-case-study-1-dirt-road-driving.pdf

employee to hold a passport *and* a driver licence; the single line into it permits one to hold neither. The `d` on the right circle forbids an employee from being both hourly and salaried, and the double line into it requires every employee to be one or the other. The two circles say nothing about each other: an hourly employee is neither more nor less likely to hold a passport.



The same structure turns up elsewhere: `GAME` in week1-lab-practical-1-course-tools[4] splits disjointly into `SINGLEPLAYER` / `MULTIPLAYER` (*how it's played*) **and**, separately, disjointly into `WINNABLE` / `NON-WIN` (*whether it can be won*) — four subclasses, but two unrelated specialisations. `EMPLOYEE` is also specialised disjointly into `SECRETARY` / `ENGINEER` (with `TypingSpeed` and `EngineerDetails` respectively) in the running company diagram, which is a third, separate characteristic again.

Reading two circles as one is a common misreading: subclasses under *different* circles say nothing about each other, so disjointness in one specialisation places no restriction on membership in the other. See interpreting-er-diagrams[5] for the statement-evaluation version of this trap.

Subclasses carry their own relationships

Because a subclass is a full entity type in its own right, relationships can attach to it directly, and this is often the whole point of creating it. `SPECIALIST` (not `DOCTOR`) is what `RUNS` a `SPECIALIST CLINIC`, and `HEALTHWORKER` oversees `HEALTH` participants specifically, not every `PARTICIPANT`. Attaching a relationship at the subclass level rather than the superclass level is itself a design decision — see er-design-choices[6].

What happens on mapping

Each subclass becomes its own relation whose primary key *is* the superclass's key, plus a foreign key back to the superclass — `Driver [id, licence]` with `Driver.id references`

[4] courses/infs1200/week1-lab-practical-1-course-tools.pdf
[5] courses/infs1200/interpreting-er-diagrams.pdf
[6] courses/infs1200/er-design-choices.pdf

`Staff.id`. The subclass relation holds only its own specific attributes: repeating the superclass's attributes there (e.g. copying `make`/`model` from `Vehicles` onto `4WD`) is a mapping error, since it wastes space and lets the two copies drift apart. See step +1 of er-to-relational-mapping[7], er-mapping-pitfalls[8] for the rest of the subclass mistakes, and week4-tutorial-case-study-3-dirt-road-driving[9].

[7] courses/infs1200/er-to-relational-mapping.pdf
[8] courses/infs1200/er-mapping-pitfalls.pdf
[9] courses/infs1200/week4-tutorial-case-study-3-dirt-road-driving.pdf