Conceptual Database Design

Table of contents

The Universe of Discourse 1
A model is never perfect 2
What the ER model is for 3
Where conceptual design sits 3

Conceptual design is everything that happens *before* a single table exists: deciding which slice of the world the database is about, and expressing that slice as an ER/EER diagram. It is a very important phase in designing a successful database application, and it follows a two-step procedure. Introduced in 2026-02-23-conceptual-database-design-and-the-er-model[1].

The Universe of Discourse

- **Step 1** — identify the **Universe of Discourse** (UoD): the "mini-world" the database models. A database never models everything in the world, only the part an organisation actually needs to record.
- **Step 2** — convert that UoD into a data model a database can capture.

Those two steps are the whole of conceptual design in this course: *the world* → *the UoD* → *the data model*. Everything in entity-types-and-attributes[2], relationship-types[3] and specialisation-and-generalisation[4] is machinery for step 2.

The UoD is not a formality to get past. Every constraint drawn later — cardinality ratios, participation, disjointness — is *read off* the UoD rather than chosen arbitrarily (see relationship-constraints[5]), so a vague or ambiguous UoD produces a diagram whose constraints can't be

[1] courses/infs1200/2026-02-23-conceptual-database-design-and-the-er-model.pdf
[2] courses/infs1200/entity-types-and-attributes.pdf
[3] courses/infs1200/relationship-types.pdf
[4] courses/infs1200/specialisation-and-generalisation.pdf
[5] courses/infs1200/relationship-constraints.pdf

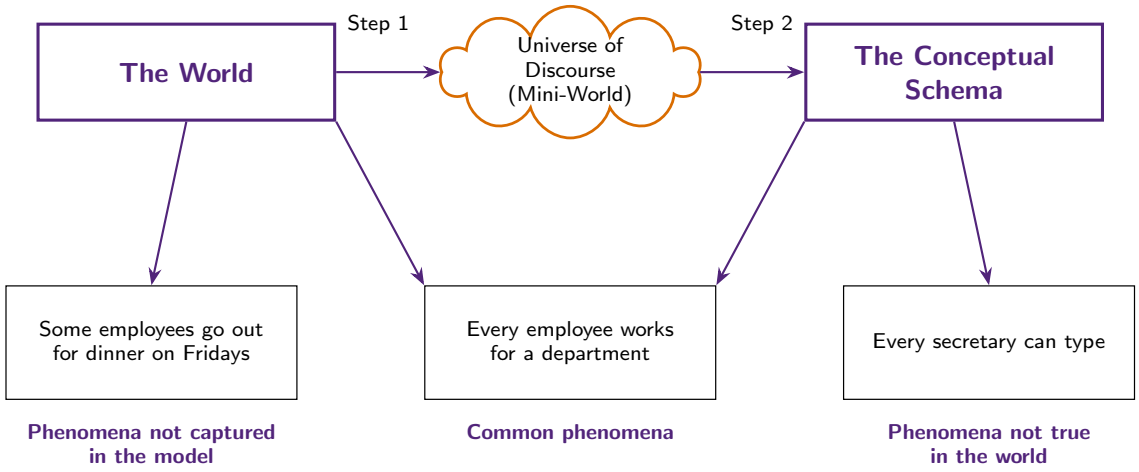


Figure 1: The two-step pipeline and the three categories of phenomena around the finished schema. The pipeline is the World → UoD → Data Model chain, and the three phenomenon boxes fan off the World and the Conceptual Schema. Only the middle box has an arrow from *both*, which is exactly what makes it *common*: true in the world and captured in the model.

justified. Where the UoD genuinely is ambiguous, that ambiguity becomes a design decision to record explicitly — see er-design-choices[6].

A model is never perfect

Around any conceptual schema there are three categories of phenomena (Figure 1):

Category	Meaning	Example
Common phenomena	True in the world *and* captured in the model	"Every employee works for a department"
Phenomena not captured	True in the world, deliberately left out as irrelevant	"Some employees go out for dinner on Fridays"
Phenomena not true in the world	Captured by the model, but an over-general assumption	"Every secretary can type"

The first category is the target. The second is fine and unavoidable — a model that captured everything would be the world itself. The third is the dangerous one: the schema will happily enforce a rule that reality violates, and the mismatch only surfaces once real data arrives.

[6] courses/infs1200/er-design-choices.pdf

What the ER model is for

The Entity-Relationship (ER) model provides a **graphical representation of data entities**, which is what makes it useful for defining a project's scope and requirements *with* clients and businesses — it's a communication artefact agreed on before implementation, not an implementation itself. Three questions run through the module: *why* conceptual modelling matters for designing a successful database application, *how* the ER model is used to do it, and how to *evaluate the trade-offs* between design choices by comparing alternative ER diagrams for the same system — the third of which is the whole content of er-design-choices[7].

An **entity** is a physical or conceptual object with data (attributes) associated with it. Crucially, the *same* real-world entity is recorded differently depending on the system's requirements: an R&D organisation records an employee's `Degree` and `Field of Study`, while a retail organisation records `Sales Experience` instead. There is no "correct" attribute list for a real-world thing in the abstract — only one that's correct for a given UoD. See entity-types-and-attributes[8] and relationship-types[9] for the building blocks, specialisation-and-generalisation[10] for the EER extension on top of them, and er-diagram-notation[11] for the symbol legend. Reading a finished diagram back out is its own skill — interpreting-er-diagrams[12].

Where conceptual design sits

"Conceptual" is also the middle layer of a DBMS's three-schema architecture (dbms-fundamentals[13]) — the level that describes the whole database for a community of users, as opposed to any one user's view of it. Conceptual design is where that schema gets written.

This course names no formal design lifecycle. The subject splits into three concerns, worked through in dependency order:

Concern	Covered by
Database design — conceptual database design, then design theory and normalisation	This note, then functional-dependencies[14] and normal-forms[15]

[7] courses/infs1200/er-design-choices.pdf
[8] courses/infs1200/entity-types-and-attributes.pdf
[9] courses/infs1200/relationship-types.pdf
[10] courses/infs1200/specialisation-and-generalisation.pdf
[11] courses/infs1200/er-diagram-notation.pdf
[12] courses/infs1200/interpreting-er-diagrams.pdf
[13] courses/infs1200/dbms-fundamentals.pdf
[14] courses/infs1200/functional-dependencies.pdf
[15] courses/infs1200/normal-forms.pdf

Concern	Covered by
Data models — the relational model, its integrity constraints, and ER-to-relational mapping	relational-model[16], integrity-constraints[17], er-to-relational-mapping[18]
Physical storage — file organisation, hashing, indexing	Explicitly deferred to later INFS courses

So the working pipeline is: conceptual design produces the ER/EER diagram; mapping turns it into a relational schema; SQL (sql-ddl[19]) declares that schema; and normalisation refines it afterwards. Physical storage — the internal level of the three-schema architecture — never enters the picture here, which is exactly what physical data independence is supposed to buy.

That ordering is why conceptual mistakes are expensive. A decision that looks cosmetic on the diagram propagates into the schema and into the constraints the DBMS can actually enforce — the mapping errors dissected in week4-tutorial-case-study-3-dirt-road-driving[20] are all downstream consequences of getting an entity's identity or a relationship's endpoint wrong at the conceptual stage.

[16] courses/infs1200/relational-model.pdf
[17] courses/infs1200/integrity-constraints.pdf
[18] courses/infs1200/er-to-relational-mapping.pdf
[19] courses/infs1200/sql-ddl.pdf
[20] courses/infs1200/week4-tutorial-case-study-3-dirt-road-driving.pdf