Entity Types and Attributes

Table of contents

Entity, entity type, entity set 1
Kinds of attribute 2
Key attributes and candidate keys 4
Value sets and nulls 4
Why the attribute kind matters later 6

The two smallest building blocks of an ER diagram: the *kinds of thing* a database records, and the *facts* it records about them. Introduced in 2026-02-23-conceptual-database-design-and-the-er-model[1] and practised in week2-tutorial-applied-class-1-intro-to-dbms-and-basic-erd[2]; see er-diagram-notation[3] for the symbols and the naming style guide.

Entity, entity type, entity set

Three words that are easy to blur together, and are worth keeping apart:

Term	Meaning
Entity	A physical or conceptual object with data associated with it — one *particular* student, one *particular* chef
Entity type	The format for the data that needs to be recorded to represent a particular entity — described by its *name* and its *attributes*. Drawn as a rectangle with attribute ovals attached

[1] courses/infs1200/2026-02-23-conceptual-database-design-and-the-er-model.pdf
[2] courses/infs1200/week2-tutorial-applied-class-1-intro-to-dbms-and-basic-erd.pdf
[3] courses/infs1200/er-diagram-notation.pdf

Term	Meaning
Entity set	The collection of all entities of one entity type in the database at any point in time

The entity type is the schema-level thing (it doesn't change as data comes and goes); the entity set is the instance-level thing (it changes with every insert and delete). The entity set is what becomes a table once the diagram is mapped — see er-to-relational-mapping[4].

Which attributes an entity type carries is decided by the UoD, not by the real-world object: an R&D organisation records an employee's `Degree` and `Field of Study` where a retail organisation records `Sales Experience` instead. See conceptual-database-design[5].

Kinds of attribute

Attributes are classified along several independent axes — an attribute can be composite *and* multivalued, key *and* composite, and so on:

Kind	Meaning	Example
Simple	Can't be meaningfully split further	`Email`, `DoB`
Composite	Divides into smaller parts which are simple attributes *with independent meaning*	`Name` → `Fname`, `Mname`, `Lname`; `Address` → `Number`, `Sname`, `Suburb`, `Pcode`
Single-valued	Holds at most one value per entity	`DoB`
Multivalued	Can hold more than one value per entity (double-lined oval)	`Degree` — one person can hold several; `Specialisations`, `Ingredients`, `Locations`
Stored	Physically recorded	`BirthDate`
Derived	Computed from another stored attribute (dashed oval)	`Age` from `BirthDate`; `Number_of_employees` on a department
Key	Value is unique across the whole entity set (name underlined)	`StudentID`, `Ssn`

A single `EMPLOYEE` entity type can carry every kind at once:

[4] courses/infs1200/er-to-relational-mapping.pdf
[5] courses/infs1200/conceptual-database-design.pdf

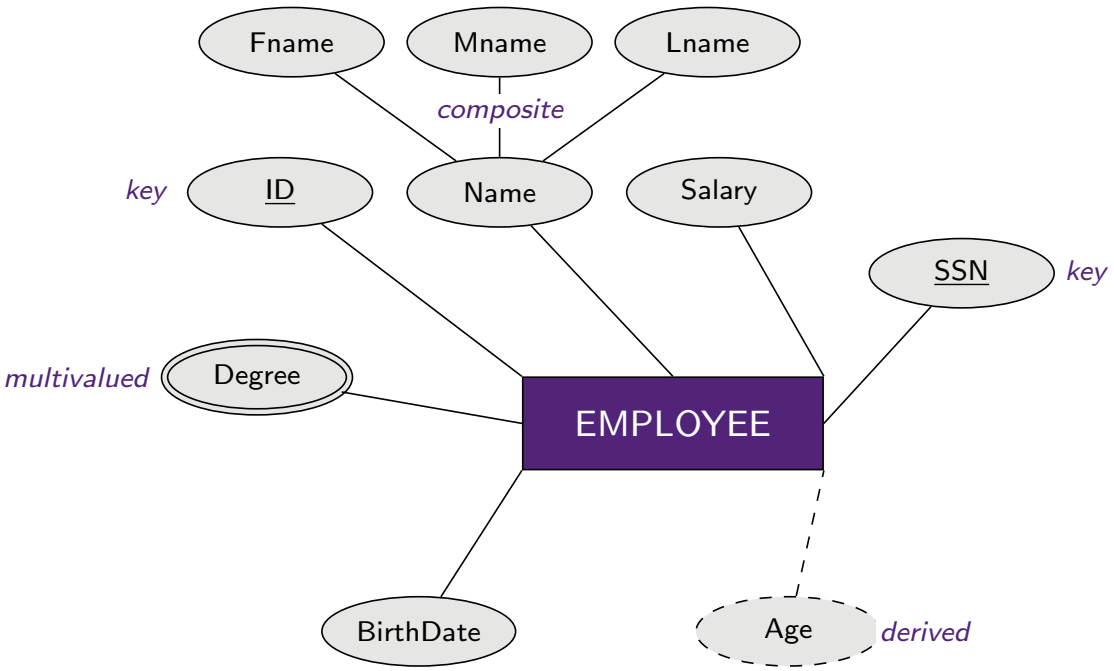


Figure 1: Every kind of attribute on one `EMPLOYEE` entity type. `ID` and `SSN` are two independent key attributes (solid underline). `Name` is composite — it fans out into `Fname`/`Mname`/`Lname`, and the parent oval itself is unmarked. `Degree` is multivalued (double outline): one person can hold several degrees. `Age` is derived from `BirthDate`, so it gets a dashed outline *and* a dashed connecting line, and it is never stored.

The single/multivalued split is stated over *simple* attributes — a composite attribute's multiplicity is really a property of its components. The distinction between composite and multivalued is the one that catches people out: `Name` splitting into first/middle/last is *composite* (one name made of parts), whereas `Phone` on a student is *multivalued* (several independent phone numbers). Both attach several ovals to the diagram, but they mean different things and map differently.

Key attributes and candidate keys

Every entity type has at least one **key constraint** (equivalently, a *uniqueness constraint*), carried by a key attribute whose value is distinct for each individual entity in the entity set. Two properties matter:

- **Multiple keys are possible.** An entity type can have several independent candidate keys, each underlined in its own oval — the `EMPLOYEE` above carries both `ID` and `SSN`. Elsewhere in the course: a `DISH` identified by either its `Name` or its `Number`, a `VENUE` by either its `ID` or the combination of `Name` and `Address`, a `VACCINE` by either its `ID` or `Brand` + `Name`. Any claim that an entity is "only identifiable by X" is false as soon as a second key oval is on the diagram.
- **A key must hold for every possible extension of the entity type.** It isn't enough that the current data happens to have no duplicates; the key must be a rule about every state the entity set could ever reach.

A **composite key** is a combination of simple attributes that together must be unique — a car's `Registration` = `State` + `Number`, alongside a separate simple key `VehicleID`. On the diagram this is drawn as one underlined parent oval that splits into its component ovals, which is how you can tell "these two attributes are jointly a key" apart from "these are two separate keys":

The single exception to "every entity type has a key" is the **weak entity**, which has no key attribute of its own and is identified by its owner's key plus its own partial key — see relationship-constraints[7]. See also relational-keys[8] for the same idea stated over relations rather than entity types.

Value sets and nulls

An attribute's **value set** specifies the set of values that may be assigned to that attribute of an entity — `employeeAge`: integers between 21 and 65; `vehicleRegistrationNum`: a string of three alphabetic characters followed by three integers. Value sets are never displayed on the

[6] courses/infs1200/er-to-relational-mapping.pdf
[7] courses/infs1200/relationship-constraints.pdf
[8] courses/infs1200/relational-keys.pdf

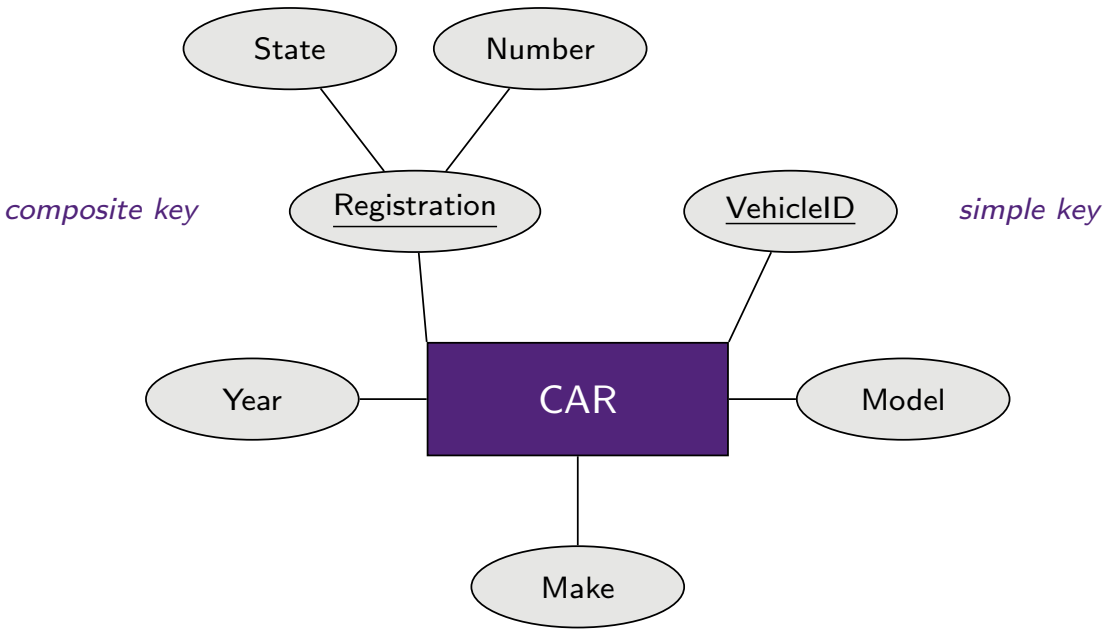


Figure 2: A `CAR` entity type with two candidate keys side by side: the composite `Registration` (underlined parent oval, splitting into `State` and `Number` — neither of which is unique alone) and the simple `VehicleID`. `Year`, `Model` and `Make` are ordinary attributes. Which of the two candidate keys is chosen as the primary key is deferred to the mapping stage, and changes the resulting schema — see er-to-relational-mapping[6].

ER diagram itself — they resurface later as the **domains** of the relational-model[9], enforced as the domain constraint among the integrity-constraints[10].

A particular entity may not have an applicable value for an attribute, and a **null** value is used to represent that. Three reasons a null appears are worth keeping apart, because they are different facts about the world:

Reason	Example
Not applicable	`tertiaryDegree` for a person with no university education
Not known whether it exists	`homePhone`
Missing — exists, but not recorded	`height`

Nulls are the reason an attribute being "optional" is a real modelling decision rather than an afterthought; see sql-nulls[11] for how they behave once queried.

Why the attribute kind matters later

The classification isn't decorative — each kind maps differently:

- A **multivalued** attribute becomes its own relation, keyed on the owner's key *plus the value itself*: `Staff`'s multivalued `Phone` becomes `StaffPhone [id, phone]`. It has to — a relational column holds one atomic value (relational-model[12]). Leaving `phone` out of that key would cap each staff member at one phone number, defeating the point of making it multivalued in the first place.
- A **derived** attribute is computed on demand, so it gets no column of its own: a trip's `Fare` is derived from its start and end times and simply doesn't appear in the mapped `Trip` relation.
- A **composite** attribute is flattened into its simple components (`Name` → `firstName`, `middleName`, `lastName`).

All three are worked through in week4-tutorial-case-study-3-dirt-road-driving[13]; the mapping rules themselves live in er-to-relational-mapping[14].

[9] courses/infs1200/relational-model.pdf
[10] courses/infs1200/integrity-constraints.pdf
[11] courses/infs1200/sql-nulls.pdf
[12] courses/infs1200/relational-model.pdf
[13] courses/infs1200/week4-tutorial-case-study-3-dirt-road-driving.pdf
[14] courses/infs1200/er-to-relational-mapping.pdf