Relationship Types

Table of contents

Relationship, relationship type, relationship set 1
Degree 2
Roles 2
Recursive relationships 3
Constraints 4
What happens to a relationship on mapping 4

The other half of an ER diagram: the associations *between* entity types. Introduced in 2026-02-23-conceptual-database-design-and-the-er-model[1] and practised in week3-tutorial-applied-class-2-entity-relationship-diagrams[2]; see er-diagram-notation[3] for the symbols and entity-types-and-attributes[4] for what sits at each end.

Relationship, relationship type, relationship set

The same schema/instance split as entity types:

Term	Meaning
Relationship	An association among two or more entities — "Paris Lane works on the project FileZilla"
Relationship type	The definition of that association, drawn as a diamond connected to the participating entity types
Relationship set	The collection of all relationships of one relationship type in the database at any point in time

[1] courses/infs1200/2026-02-23-conceptual-database-design-and-the-er-model.pdf
[2] courses/infs1200/week3-tutorial-applied-class-2-entity-relationship-diagrams.pdf
[3] courses/infs1200/er-diagram-notation.pdf
[4] courses/infs1200/entity-types-and-attributes.pdf

A relationship set can be drawn as lines between two ovals of entities, or written out as a table of pairs — `WORKSON` as rows `(e1, p1)`, `(e2, p1)`, `(e2, p4)`, and so on. The tabular form is the one exam questions use when they ask whether a given relationship set is *possible* under a diagram's constraints; see interpreting-er-diagrams[5].

A relationship type may carry its own **descriptive attributes**, which belong to the *pairing* rather than to either entity on its own — `EMPLOYEE WORKSON PROJECT` hangs `StartDate` and `EndDate` off the diamond, because a start date belongs to one employee's stint on one project and to nothing else. A `Grade` belongs to the enrolment of one student in one course, not to the student and not to the course. Other examples across the notes: a `Placement` on `PARTICIPATES` (athlete/event), a `Fee` on `CONSULTS` (patient/doctor), a `PrepTime` on `BUYS` (customer/bubble tea).

A relationship type may also carry **key attributes** of its own — on `WORKSON` it is `StartDate` that is underlined, which says one employee cannot start on the same project twice on the same day. The `Timestamp` that distinguishes repeat orders on `BUYS` does the same job.

Degree

The **degree** of a relationship type is the number of participating entity types:

Degree	Name	Example
2	**Binary**	`DEPARTMENT WORKSFOR EMPLOYEE`
3	**Ternary**	`SUPPLIER` / `PROJECT` / `PART` in one `SUPPLIES` diamond
n	**n-ary**	The general case, 3+ entity types

Degree is decided by counting the lines into the diamond, and it is a real design decision rather than a description — the same UoD can often be modelled as one ternary relationship or as three binary ones, with different consequences. See er-design-choices[6].

Roles

Each entity type that participates in a relationship type plays a particular **role** in it. The **role name** signifies the role a participating entity from that entity type plays in each relationship instance — that is to say, *it explains what the relationship means.* `DEPARTMENT` is the *Employer* and `EMPLOYEE` is the *Worker* in `WORKSFOR`, the relationship between those two entity types.

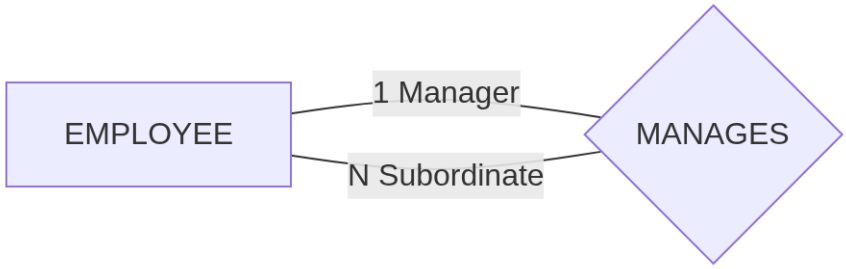
[5] courses/infs1200/interpreting-er-diagrams.pdf
[6] courses/infs1200/er-design-choices.pdf

`WORKSON` is a different relationship: the M:N `EMPLOYEE/PROJECT` one (relationship-constraints[7], er-to-relational-mapping[8]).

Roles are optional labelling on an ordinary binary relationship between two different entity types, where the meaning is usually obvious from the names — but they become mandatory in the recursive case below, where they're the only thing distinguishing the two ends.

Recursive relationships

The same entity type can participate more than once in one relationship type, under different roles — an `EMPLOYEE` who `MANAGES` another `EMPLOYEE`. Both lines run from the same rectangle back to the same diamond, so the role labels carry all of the meaning:



Two different relationships share the name `MANAGES`, and they are not the same thing: this recursive `EMPLOYEE/EMPLOYEE` one, and the 1:1 `EMPLOYEE/DEPARTMENT` one that records who runs each department (relationship-constraints[9], and the `mgrSSN` foreign key in er-to-relational-mapping[10]). The recursive relationship is also named `SUPERVISION` in the company diagram used for mapping, and that is the name used for it elsewhere in these notes.

Recursive relationships turn up constantly once you start looking, and they take every cardinality:

Relationship	Roles	Ratio
`EMPLOYEE SUPERVISION EMPLOYEE`	*supervisor* / *supervisee*	1:N
`COURSE PREREQUISITE COURSE`	*higher* / *requires*	M:N
`CUSTOMER FRIENDS CUSTOMER`	*Requestor* / *Requestee*	M:N
`VACCINE WITH VACCINE`	*First* / *Second*	partial on both sides
`NON-WIN SEQUEL NON-WIN`	*PrequelTo* / *SequelTo*	1:1

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

The `WITH` example from week3-tutorial-applied-class-2-entity-relationship-diagrams[11] is the instructive one: because participation is *partial* on both sides, the relationship records that a vaccine *may* be part of a two-dose program — it does not assert that every vaccine has a second dose. Reading a recursive relationship as if it were total is a standard trap.

Constraints

Cardinality ratio (1:1, 1:N, M:N) and participation (total/partial) are covered in relationship-constraints[12]; how to read a claim off them without tripping over the direction is covered in interpreting-er-diagrams[13].

What happens to a relationship on mapping

The relationship type's cardinality decides its fate in the relational schema. An M:N relationship becomes its own relation, combining both sides' foreign keys into one composite primary key (relational-keys[14]), plus any descriptive attributes it carried — `UserRatesDriver [userID, driverID, rating]` in week4-tutorial-case-study-3-dirt-road-driving[15]. A 1:N relationship doesn't need a relation of its own; it shows up as a foreign key on the N side, the way `EmergencyContact.userID` carries the `User`/`EmergencyContact` link in the same schema. The full set of rules is in er-to-relational-mapping[16].

[11] courses/infs1200/week3-tutorial-applied-class-2-entity-relationship-diagrams.pdf
[12] courses/infs1200/relationship-constraints.pdf
[13] courses/infs1200/interpreting-er-diagrams.pdf
[14] courses/infs1200/relational-keys.pdf
[15] courses/infs1200/week4-tutorial-case-study-3-dirt-road-driving.pdf
[16] courses/infs1200/er-to-relational-mapping.pdf