

Exercise walk-through

8.1

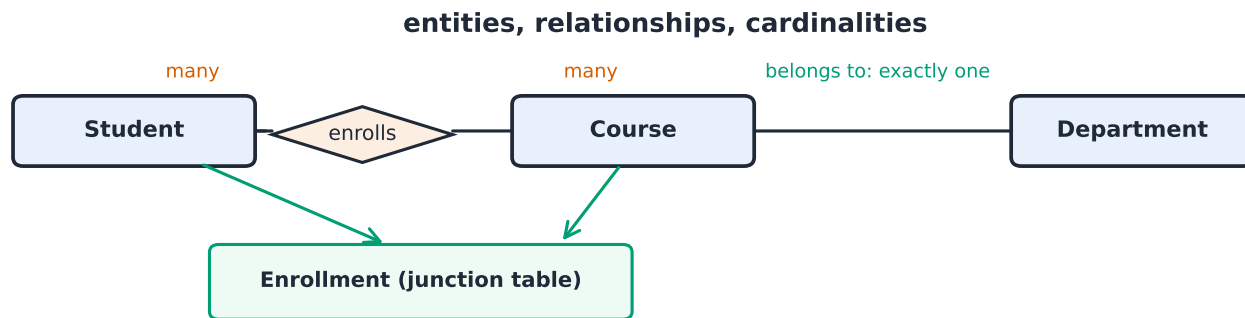
Relational modelling

University Computer Science

Relational modelling: entities to keys

- 1 **Model** Student, Course, Department are entities; enrolls is many-to-many and becomes the Enrollment junction table of foreign-key pairs.
- 2 **Keys PRIMARY KEY** (student, course, term) rejects a duplicate triple at the engine - rules live in the schema.
- 3 **Join** Student to Enrollment to Course to Department follows the foreign keys: the sample data (8-6-14) yields 14 rows.

ER diagram



many-to-many becomes its own table of foreign-key pairs; PRIMARY KEY (student, course, term) enrolls once per term

referential integrity: every Enrollment.student and .course names a real row

sample data: 8 students - 6 courses - 14 enrollments

Students enroll in Courses through the Enrollment junction table.

Exercises 2.1-2.3

2.1 A department has one building. Name the table that should carry the column, and why. [3]

2.2 Model enrolls-in as a many-to-many: give the junction table's two foreign keys. [3]

2.3 State what referential integrity forbids, in one sentence. [3]

Exercises 2.1-2.3

2.1 A department has one building. Name the table that should carry the column, and why. [3]

2.1

- Department – one value per department, so it lives on the one side ✓.
-

2.2 Model enrolls-in as a many-to-many: give the junction table's two foreign keys. [3]

2.2

- `student_id` referencing Student and `course_id` referencing Course ✓.
-

2.3 State what referential integrity forbids, in one sentence. [3]

2.3

- No referencing row may name a target row that does not exist ✓.

Exercise 3.1

3.1 A cinema domain: Customer, Film, Screening (a film shown in a room at a time), Ticket.

Identify the entities and give a one-to-many and a many-to-many relationship with their table treatments. Choose a sensible primary key for Ticket and justify it against the alternative of a surrogate ID. State the rule a composite key on Ticket would need to enforce one seat per screening. [8]

Exercise 3.1

3.1 A cinema domain: Customer, Film, Screening (a film shown in a room at a time), Ticket.

Identify the entities and give a one-to-many and a many-to-many relationship with their table treatments. Choose a sensible primary key for Ticket and justify it against the alternative of a surrogate ID. State the rule a composite key on Ticket would need to enforce one seat per screening. [8]

3.1

- Entities: Customer, Film, Screening, Ticket ✓.
- One-to-many: Screening to Ticket – `ticket.screening_id` foreign key ✓.
- Many-to-many: Customer to Film through Ticket (a customer watches many films; a film gathers many customers) ✓.
- Key for Ticket: (`screening_id`, `seat_no`) identifies one seat at one showing – natural and duplicate-proof ✓.
- Surrogate alternative: a `ticket_id` is stable if seats get renumbered and carries no meaning to maintain ✓.
- Composite rule: PRIMARY KEY (`screening_id`, `seat_no`) rejects a second sale of the same seat ✓.

Exercise 3.2

3.2 The enrolment schema stores PRIMARY KEY (student_id, course_id, term) on Enrollment.

Explain what this key rejects, with the exact duplicate it stops. State what is lost if the key is dropped and only the application checks duplicates. Give the two foreign keys the table also needs and what each protects. [7]

Exercise 3.2

3.2 The enrolment schema stores PRIMARY KEY (`student_id`, `course_id`, `term`) on Enrollment.

Explain what this key rejects, with the exact duplicate it stops. State what is lost if the key is dropped and only the application checks duplicates. Give the two foreign keys the table also needs and what each protects. [7]

3.2

- It rejects a second row with the same (`student`, `course`, `term`) – the same student enrolling twice in one term ✓.
- Dropped: two racing applications both check-then-insert and both succeed – the engine no longer guarantees it ✓.
- Foreign keys: `student_id` (every enrollment names a real student) and `course_id` (a real course) ✓.