MVP Database for Yrkeshögskola YrkesCo

A structured solution to manage education operations, designed for real needs.

Presented by: *Katrin Rylander DE24*

11.04.2025

YrkesCo’s - Context & Goals

About YrkesCo

- A private higher vocational education provider
- Runs multiple programs
- Receives government funding for particular number of program iterations
- Needs to track:
  - Classes, Students, Teachers, Courses
  - Enrollment, Grades, Locations

Challenge

- Decentralized Excel-based systems
- Fragmented student/teacher/program data
- GDPR compliance risk
- No clear reporting or data access

Data Modeling Process

- Understand the business -> Identify requirements
- Define entities relationships
- Create conceptual Entity- Relationship Diagram (ERD)
- Define attributes and create logical Entity Relationship Diagram
- Normalize to 3NF
- Add logic, data types & constraints (triggers, roles)
- Implement in PostgreSQL

Real-World Flow ➡ Database Logic

Real-World Action	**Corresponding Table**
Hire education managers	**Education_manager**, **Company**, **Address**
Program approval	**Program**, **Course**
Class creation	**Class**, **Campus**
Add students	**Student**
Hire teachers	**Teacher**, Company, Address
Schedule courses	**Course_offering**
Students enroll	**Enrollment**
Teachers grade	Enrollment (grade)

Real-World Flow ➡ Database Logic

Real-World Action	**Corresponding Table**
Hire education managers	**Education_manager**, **Company**, **Address**
Program approval	**Program**, **Course**
Class creation	**Class**, **Campus**
Add students	**Student**
Hire teachers	**Teacher**, Company, Address
Schedule courses	**Course_offering**
Students enroll	**Enrollment**
Teachers grade	Enrollment (grade)

Don't forget the business rules!

Business Rules in Action

Examples:

- A person can **only have one role** at a specific time
- Managers manage **max 3 classes**
- Students **must belong to a class** in order to enroll for stand alone courses
- Teachers and Managers are **exclusive,** their roles do not overlap

Important to include it already at conceptual level!

They are enforced later with **constraints and triggers in the database**

GDPR Compliance

- Personal information must be stored separately (need for private_* tables)
- Access control by separation and permissions
- Referenced by IDs only
- Personal address will not be stored in the database:
  - retrieved from central register when needed

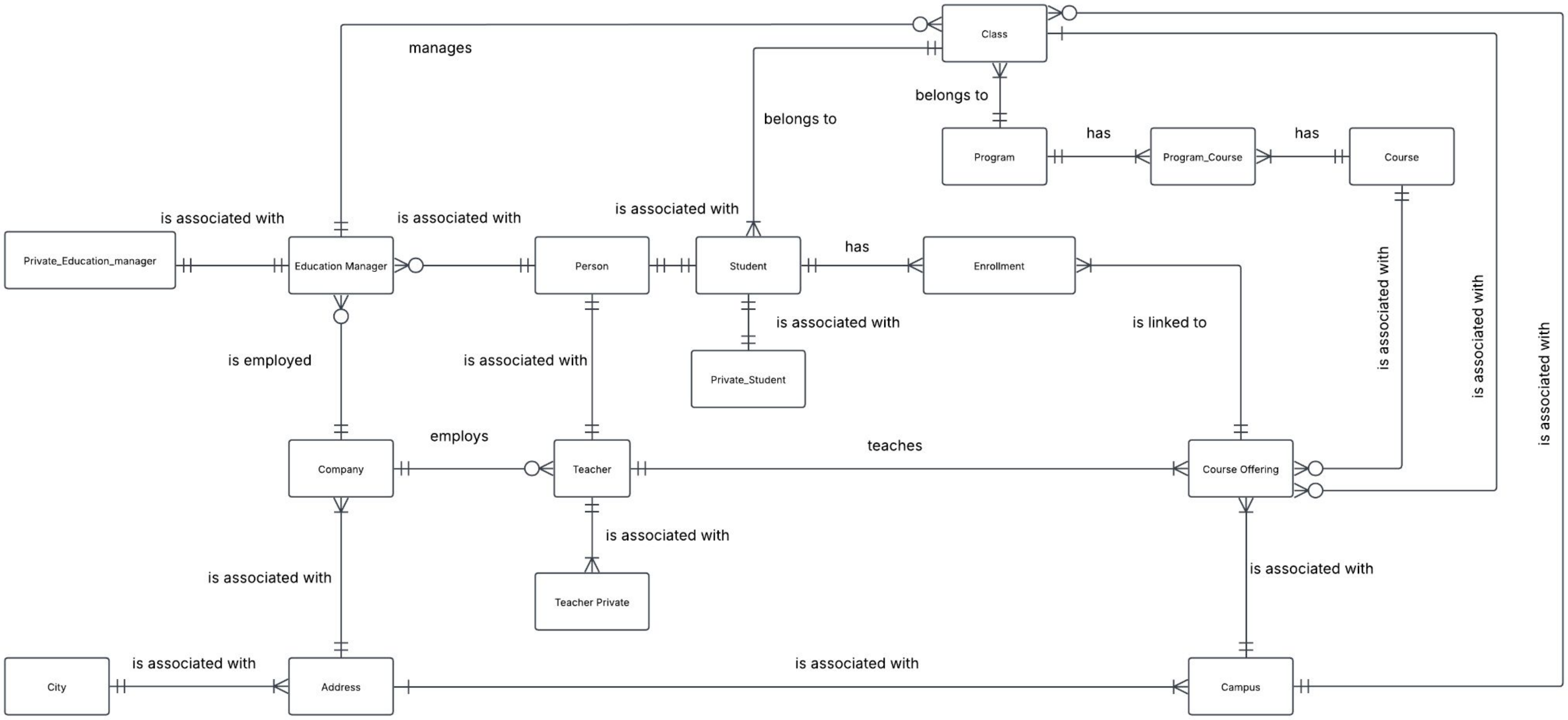
Real-World Flow ➡ Database Logic

Real-World Action	**Corresponding Table**
Hire education managers	**Person, Education_manager**, **Private_Education_manager**, **Company**, **Address**
Program approval	**Program**, **Course, ProgramCourse**
Class creation	**Class**, **Campus**
Add students	**Person, Student, Private_Student**
Hire teachers	**Person, Teacher**, **Private_Student,** Company, Address
Schedule courses	**Course_offering**
Students enroll	**Enrollment**
Teachers grade	Enrollment (grade)

Key Entities Overview

Entity	**Description**
Person	Shared base info for all roles
Student	Belongs to a Class
Education Manager	Non-teaching role managing classes, always hired directly by the school
Teacher	Can be consultant or hired by the school
Program	Abstract container for Courses
Class	Tangible instance (Live iteration) of a Program
Course	Abstract, module of learning used in programs or standalone
Course_offering	Scheduling layer (who, where, when)
Enrollment	Links students to offerings, holds grades

Conceptual ERD: High-level view of entity relationships



Selected Relationships

Program ↔ Class (One-to-Many)

A Program can have multiple Classes, but each Class belongs to one Program.

Program ↔ ProgramCourse (One-to-Many)

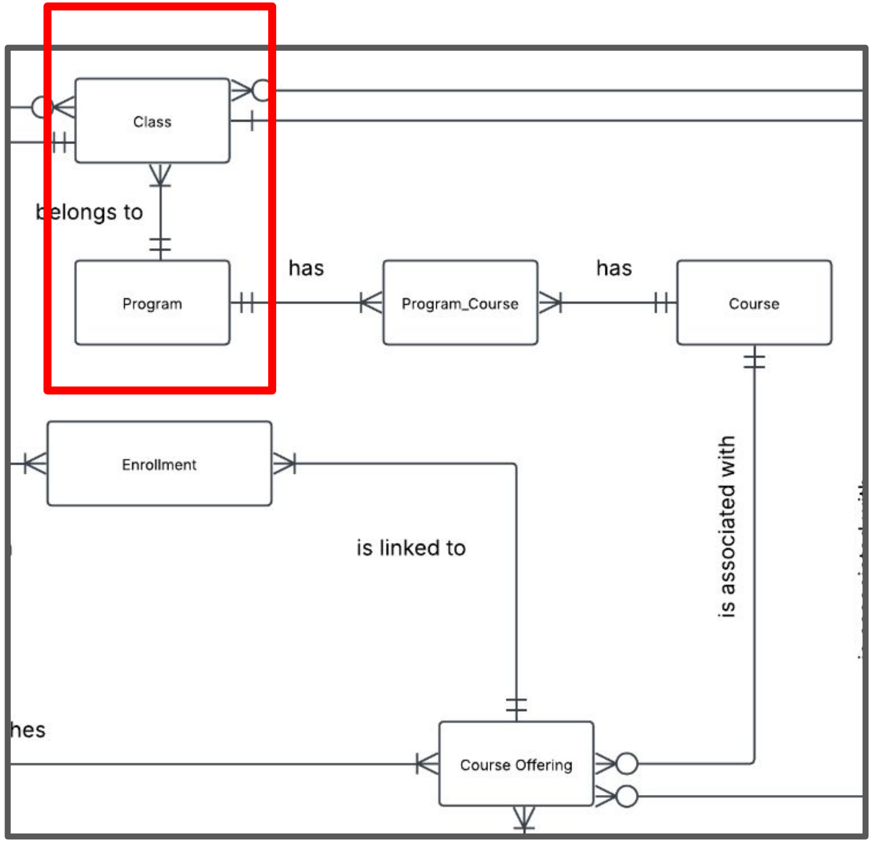
A Program can offer multiple ProgramCourses, and each ProgramCourse links one Program to one Course.

Course ↔ ProgramCourse (One-to-Many)

A Course can be included in multiple ProgramCourses, but each ProgramCourse links one Course to one Program.

Course ↔ Course_Offering (One-to-Many)

A Course can have multiple CourseOfferings, but each CourseOffering is for one Course.



Selected Relationships

Program ↔ Class (One-to-Many)

A Program can have multiple Classes, but each Class belongs to one Program.

Program ↔ ProgramCourse (One-to-Many)

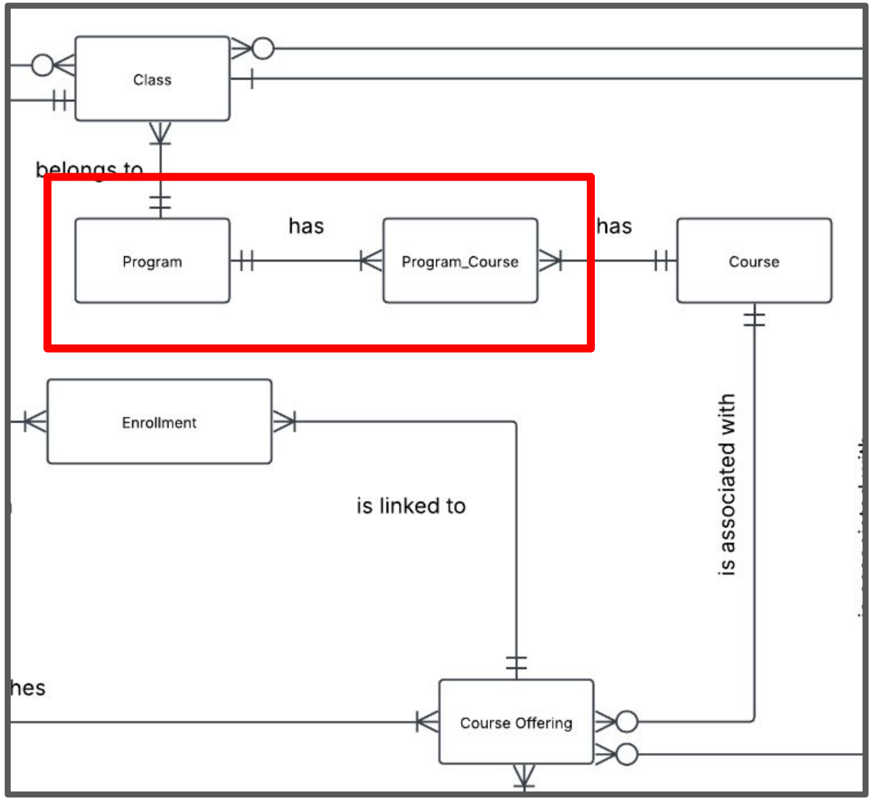
A Program can offer multiple ProgramCourses, and each ProgramCourse links one Program to one Course.

Course ↔ ProgramCourse (One-to-Many)

A Course can be included in multiple ProgramCourses, but each ProgramCourse links one Course to one Program.

Course ↔ Course_Offering (One-to-Many)

A Course can have multiple CourseOfferings, but each CourseOffering is for one Course.



Selected Relationships

Program ↔ Class (One-to-Many)

A Program can have multiple Classes, but each Class belongs to one Program.

Program ↔ ProgramCourse (One-to-Many)

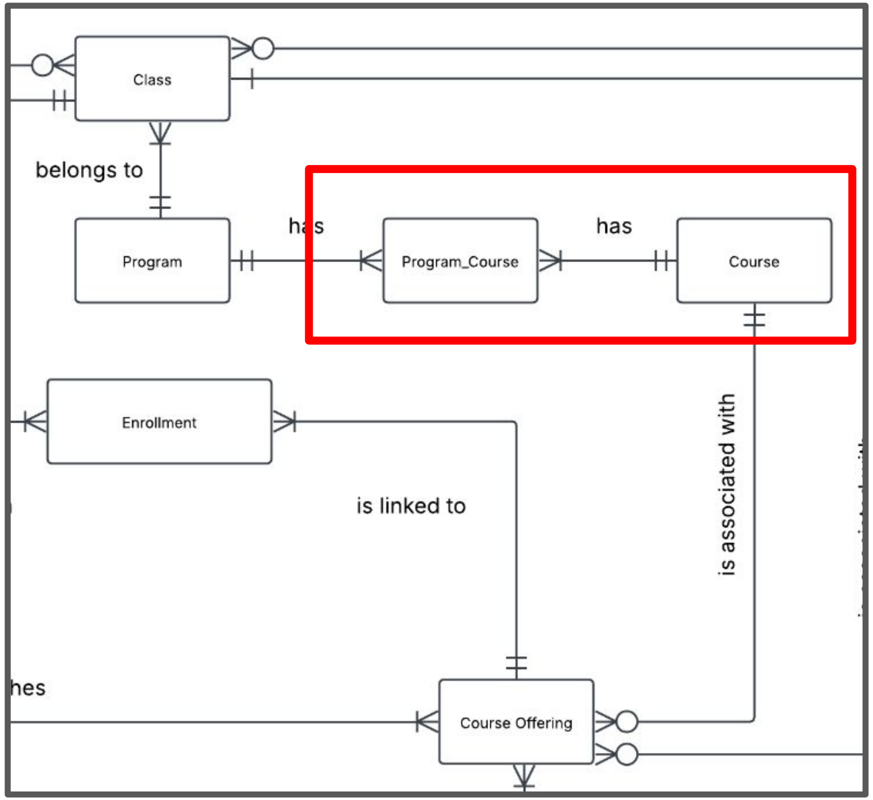
A Program can offer multiple ProgramCourses, and each ProgramCourse links one Program to one Course.

Course ↔ ProgramCourse (One-to-Many)

A Course can be included in multiple ProgramCourses, but each ProgramCourse links one Course to one Program.

Course ↔ Course_Offering (One-to-Many)

A Course can have multiple CourseOfferings, but each CourseOffering is for one Course.



Selected Relationships

Program ↔ Class (One-to-Many)

A Program can have multiple Classes, but each Class belongs to one Program.

Program ↔ ProgramCourse (One-to-Many)

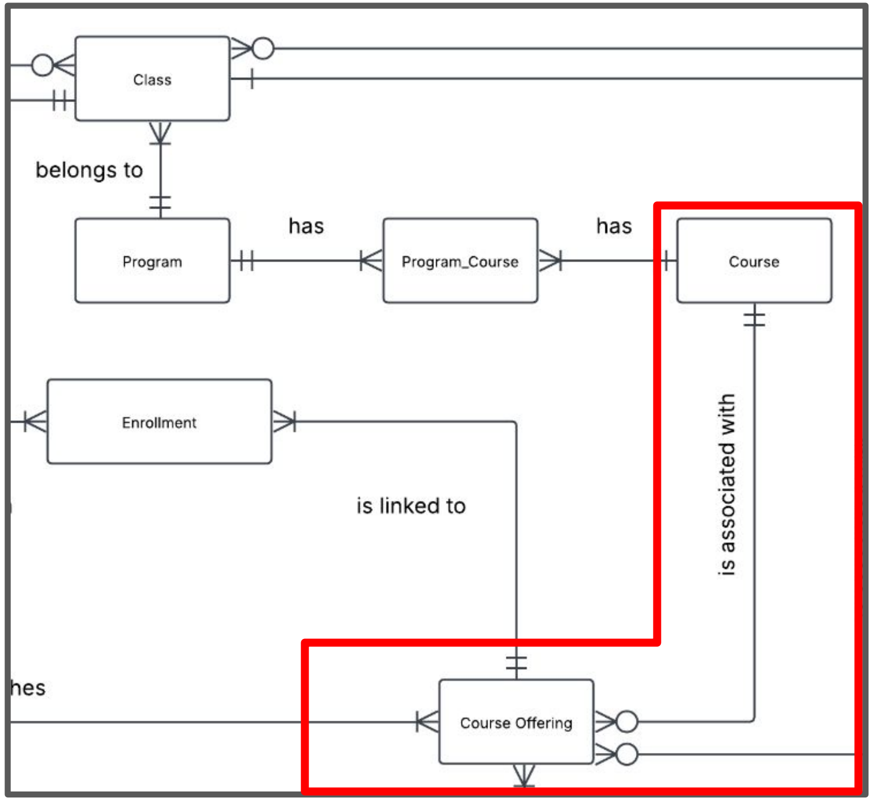
A Program can offer multiple ProgramCourses, and each ProgramCourse links one Program to one Course.

Course ↔ ProgramCourse (One-to-Many)

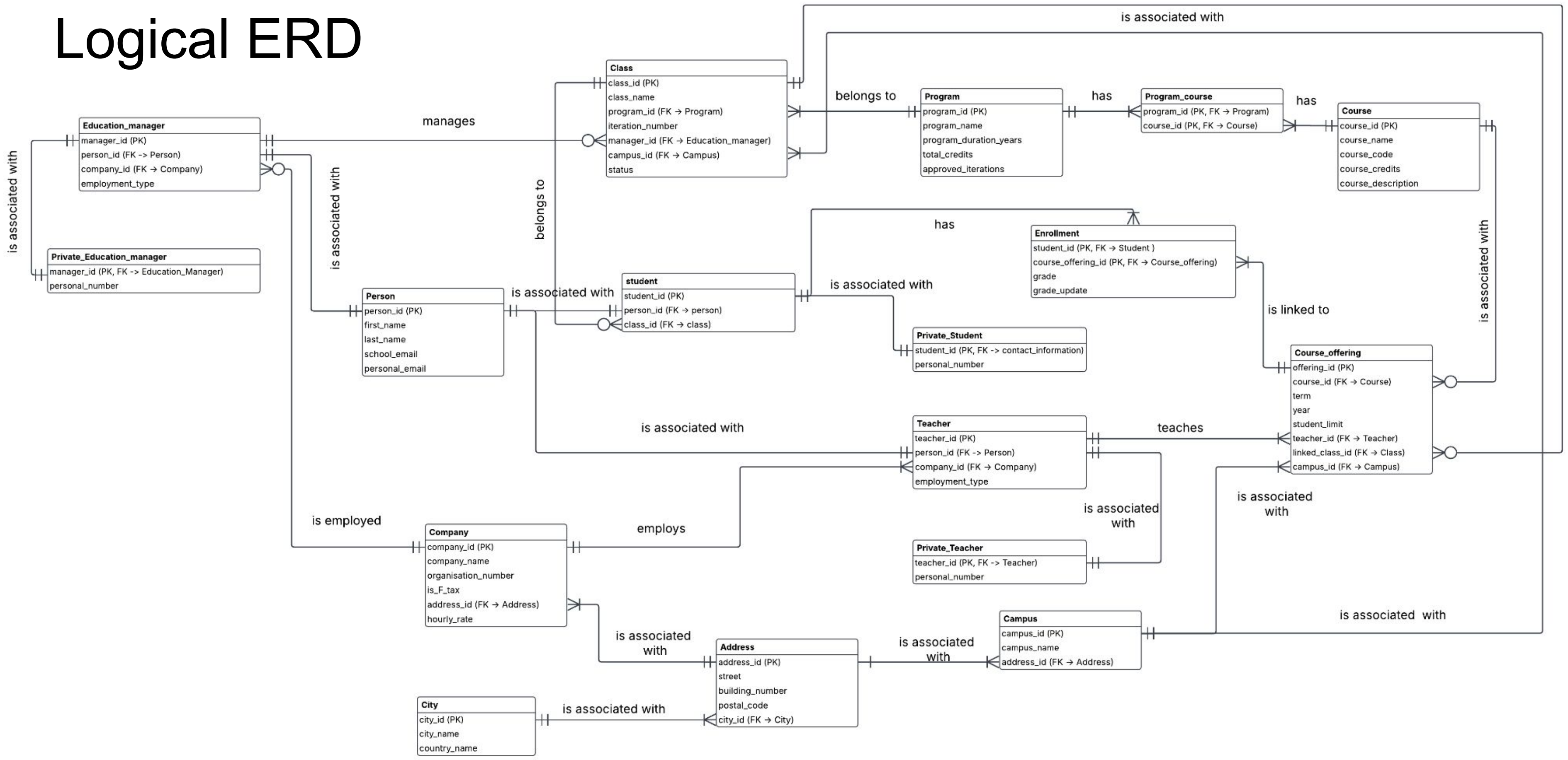
A Course can be included in multiple ProgramCourses, but each ProgramCourse links one Course to one Program.

Course ↔ Course_Offering (One-to-Many)

A Course can have multiple CourseOfferings, but each CourseOffering is for one Course.



Logical ERD


Class
class_id (PK)
class_name
program_id (FK → Program)
iteration_number
manager_id (FK → Education_manager)
campus_id (FK → Campus)
status
Program
program_id (PK)
program_name
program_duration_years
total_credits
approved_iterations
Program_course
program_id (PK, FK → Program)
course_id (PK, FK → Course)
Course
course_id (PK)
course_name
course_code
course_credits
course_description
Education_manager
manager_id (PK)
person_id (FK -> Person)
company_id (FK → Company)
employment_type
Private_Education_manager
manager_id (PK, FK -> Education_Manager)
personal_number
Person
person_id (PK)
first_name
last_name
school_email
personal_email
student
student_id (PK)
person_id (FK → person)
class_id (FK → class)
Enrollment
student_id (PK, FK → Student)
course_offering_id (PK, FK → Course_offering)
grade
grade_update
Private_Student
student_id (PK, FK -> contact_information)
personal_number
Course_offering
offering_id (PK)
course_id (FK → Course)
term
year
student_limit
teacher_id (FK → Teacher)
linked_class_id (FK → Class)
campus_id (FK → Campus)
Teacher
teacher_id (PK)
person_id (FK -> Person)
company_id (FK → Company)
employment_type
Private_Teacher
teacher_id (PK, FK -> Teacher)
personal_number
Company
company_id (PK)
company_name
organisation_number
is_F_tax
address_id (FK → Address)
hourly_rate
Address
address_id (PK)
street
building_number
postal_code
city_id (FK → City)
Campus
campus_id (PK)
campus_name
address_id (FK → Address)
City
city_id (PK)
city_name
country_name
is associated with
belongs to
has
has
manages
is associated with
is associated with
belongs to
has
is associated with
is associated with
is linked to
is associated with
is associated with
teaches
is employed
employs
is associated with
is associated with
is associated with
is associated with
is associated with
is associated with

Normalization (1NF–3NF)

3 NF check example

class_id	class_name	program_id	iteration_number	manager_id	campus_id	status
1	DE23	1	1	1	1	ongoing
2	BIM23	2	1	1	1	ongoing
3	DE24	1	2	1	1	ongoing
4	BIM24	2	2	2	2	ongoing
5	UX24	3	1	2	2	ongoing

Class

- `class_id` **(PK)**
- `class_name`
- `program_id` **(FK → Program)**
- `iteration_number`
- `manager_id` **(FK → Education_Manager)**
- `campus_id` **(FK → Campus)**
- `status: 'to open', 'ongoing', 'graduated', 'cancelled'`

✅Tables uses atomic values (1NF)

✅No partial dependency on composite PKs (2NF)

✅No transitive dependencies (3NF)

Data Integrity & Constraints

PostgreSQL Implementation

- 3NF schema
- Primary + foreign keys defined on all relationships for referential integrity
- Use of ENUMs for role, term, class status
- Trigger logic for business rules
- ON DELETE logic to avoid orphaned records

Who & How uses the database

Role	**Cares about**	**Useful queries**
Education Manager **Business role**: Oversees classes and manages communication between stakeholders.	● Teachers in each class ● Contact info of teachers and students ● Students at risk (failing grades) ● Progress of their classes ● Final report generation	● Get all teachers for a specific class ● Get emails of all students in a class ● List students with ‘IG’ grades in current term ● List final grades for all students in a class
Teacher **Business role**: Delivers content for specific course offering, grades students.	● Which courses they teach this term ● Who is enrolled in their courses ● Grading status ● Contact info of their students Access to previous course performance	● List all courses they are teaching in the current term ● Get a list of students per course + grade status ● See grading breakdown: how many students got VG/G/IG ● Notify students missing grades

Who & How uses the database

Role	Cares about	Useful queries
VP / Executive / Management **Business role**: Needs overviews to make strategic decisions.	● Total students per program/year ● Course pass/fail rates ● Program performance ● Utilization of staff ● How many consultants vs permanent teachers	● Summary of student count per program per year ● Pass rate per course ● Breakdown of employment types of teachers ● How many courses were run each year per campus

Live System Preview (Video Pitch Only)

Summary for the customer

When organizations have the right tools, they can focus on what they do best.
At YrkesCo, your best is teaching — let the tools take care of the rest.

Focus on what matters.

- Teachers and administrators should spend more time on education — not managing spreadsheets or logistics.

This solution does the heavy lifting:

- Manages student records, courses, and program details
- Tracks enrollment and academic progress
- Helps educators manage grades and student interactions with ease

Less time managing. More time teaching.
This system handles the complexity, so YrkesCo can stay focused on empowering students.