Database Design for a Vocational University (Yrkeshögskola) YrkesCo

A structured solution to manage education operations.

Presented by: *Katrin Rylander DE24*
Date: 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:
  - Students and Classes
  - Courses and Teachers
  - Enrollment and Grades

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 and relationships
- Create conceptual Entity- Relationship Diagram (ERD)
- Define attributes and create logical Entity Relationship Diagram
- Normalize to 3NF
- Add data types, logic, & constraints (triggers, roles)
- Implement in PostgreSQL
- Populate with fake data

Real-World Flow ➡ Database Logic

Real-World Action	**Corresponding Entity**
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 Entity**
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

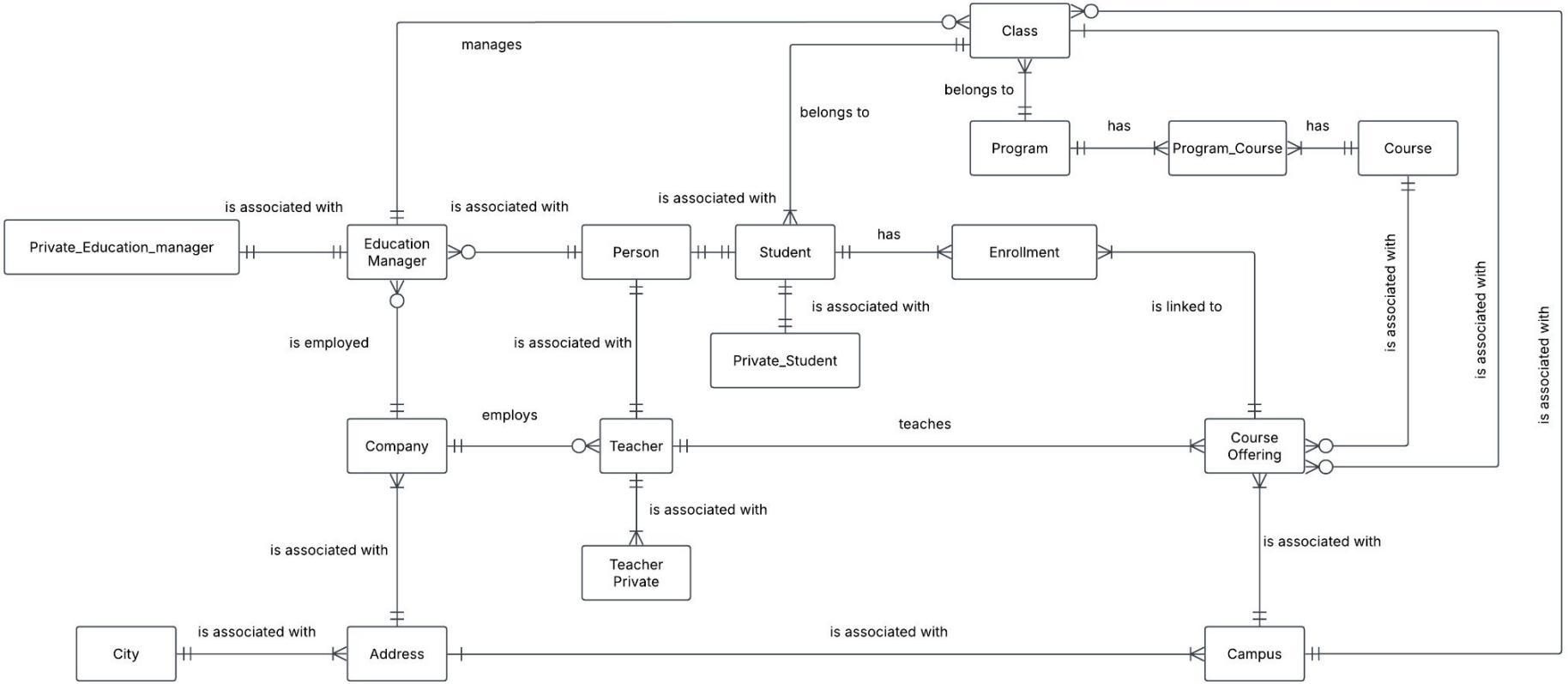
Real-World Flow ➡ Database Logic

Real-World Action	**Corresponding Entity**
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: first name, last name, school email
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	The real-world instance of an otherwise abstract course. Describes when (during a term), where (at a location), by whom (a specific teacher) the course is taught.
Enrollment	Links students to offerings, holds grades

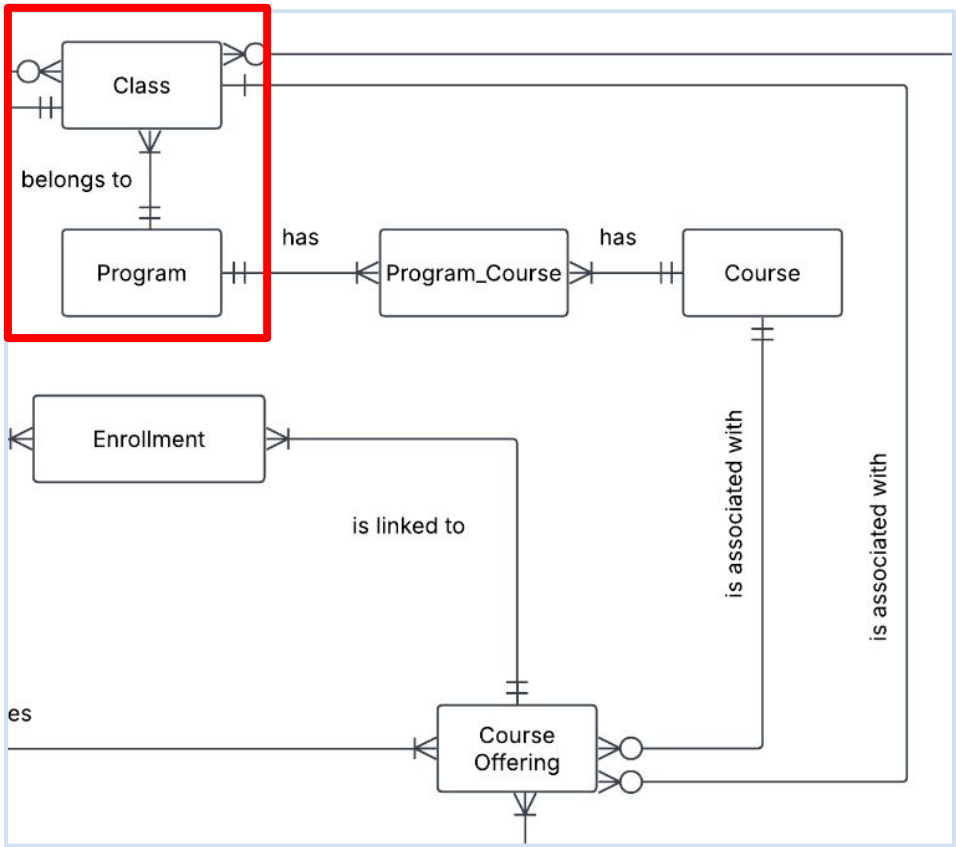
Conceptual ERD: High-level view of entity relationships



Selected Relationships

Program ↔ Class (One-to-Many)

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



Selected Relationships

Program ↔ Course (Many-to-Many)

A Program consists of multiple Courses, and each Course can be associated with multiple Programs.

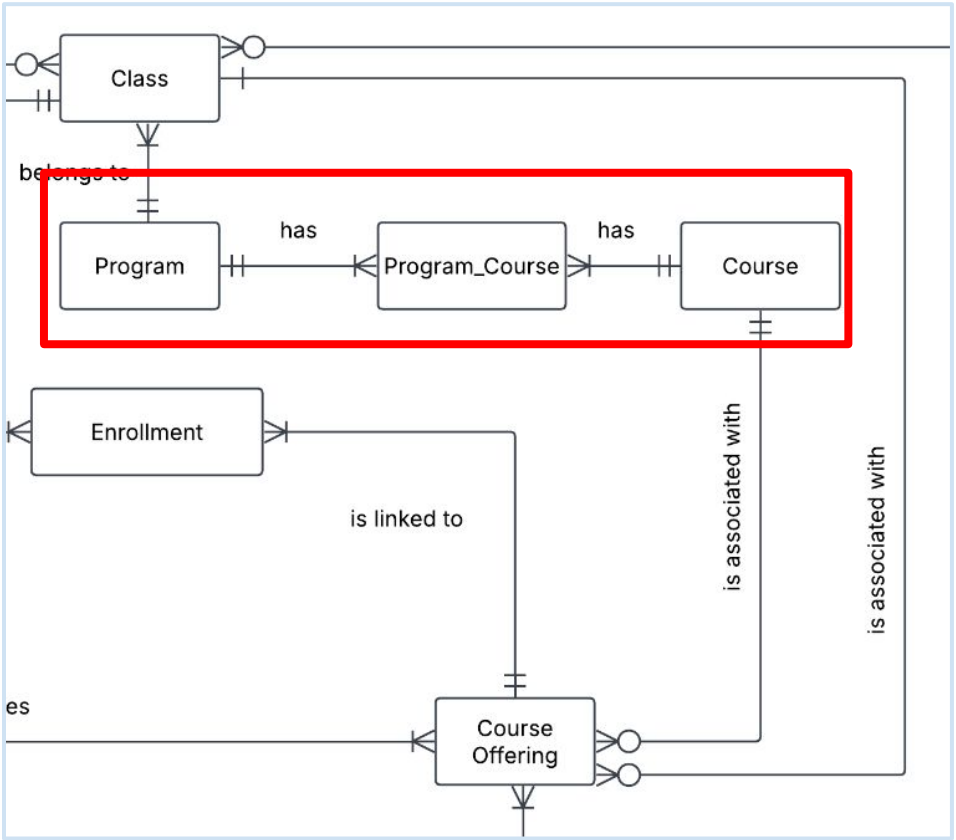
This relationship cannot be modeled directly!

Program ↔ ProgramCourse (One-to-Many)

A Program can offer multiple ProgramCourses, with each ProgramCourse linking a single Program to a single Course.

Course ↔ ProgramCourse (One-to-Many)

A Course can be part of multiple ProgramCourses, but each ProgramCourse links a single Course to a single Program.



Selected Relationships

Program ↔ Course (Many-to-Many)

A Program consists of multiple Courses, and each Course can be associated with multiple Programs.

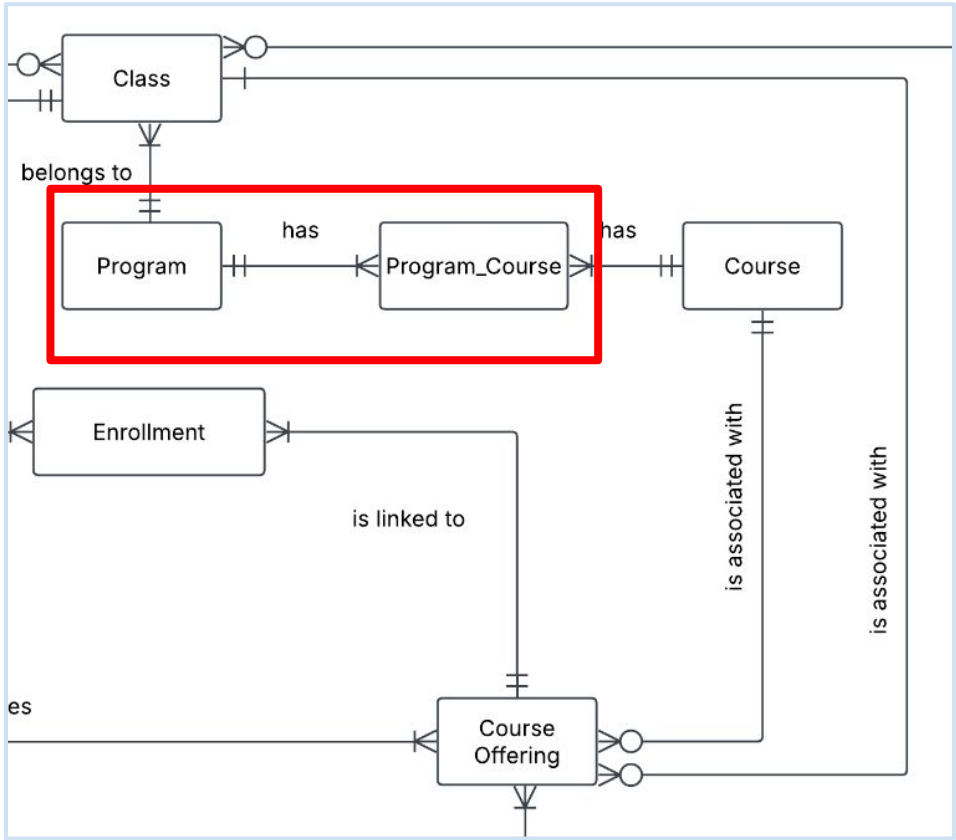
This relationship cannot be modeled directly!

Program ↔ ProgramCourse (One-to-Many)

A Program can offer multiple ProgramCourses, with each ProgramCourse linking a single Program to a single Course.

Course ↔ ProgramCourse (One-to-Many)

A Course can be part of multiple ProgramCourses, but each ProgramCourse links a single Course to a single Program.



Selected Relationships

Program ↔ Course (Many-to-Many)

A Program consists of multiple Courses, and each Course can be associated with multiple Programs.

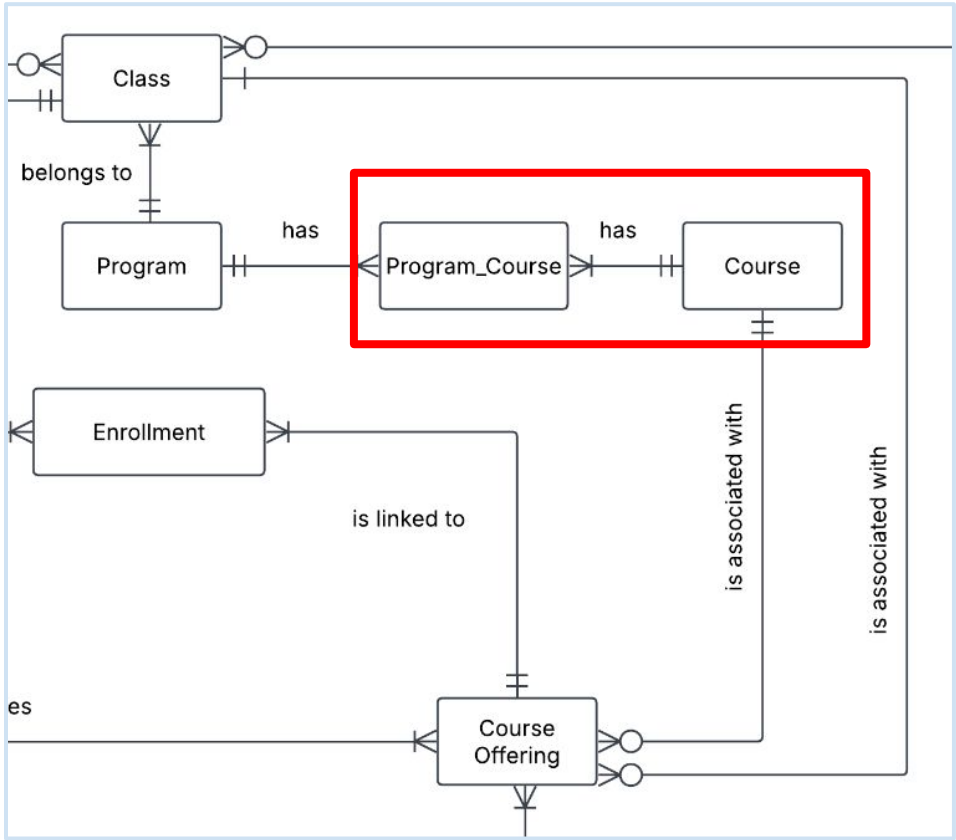
This relationship cannot be modeled directly!

Program ↔ ProgramCourse (One-to-Many)

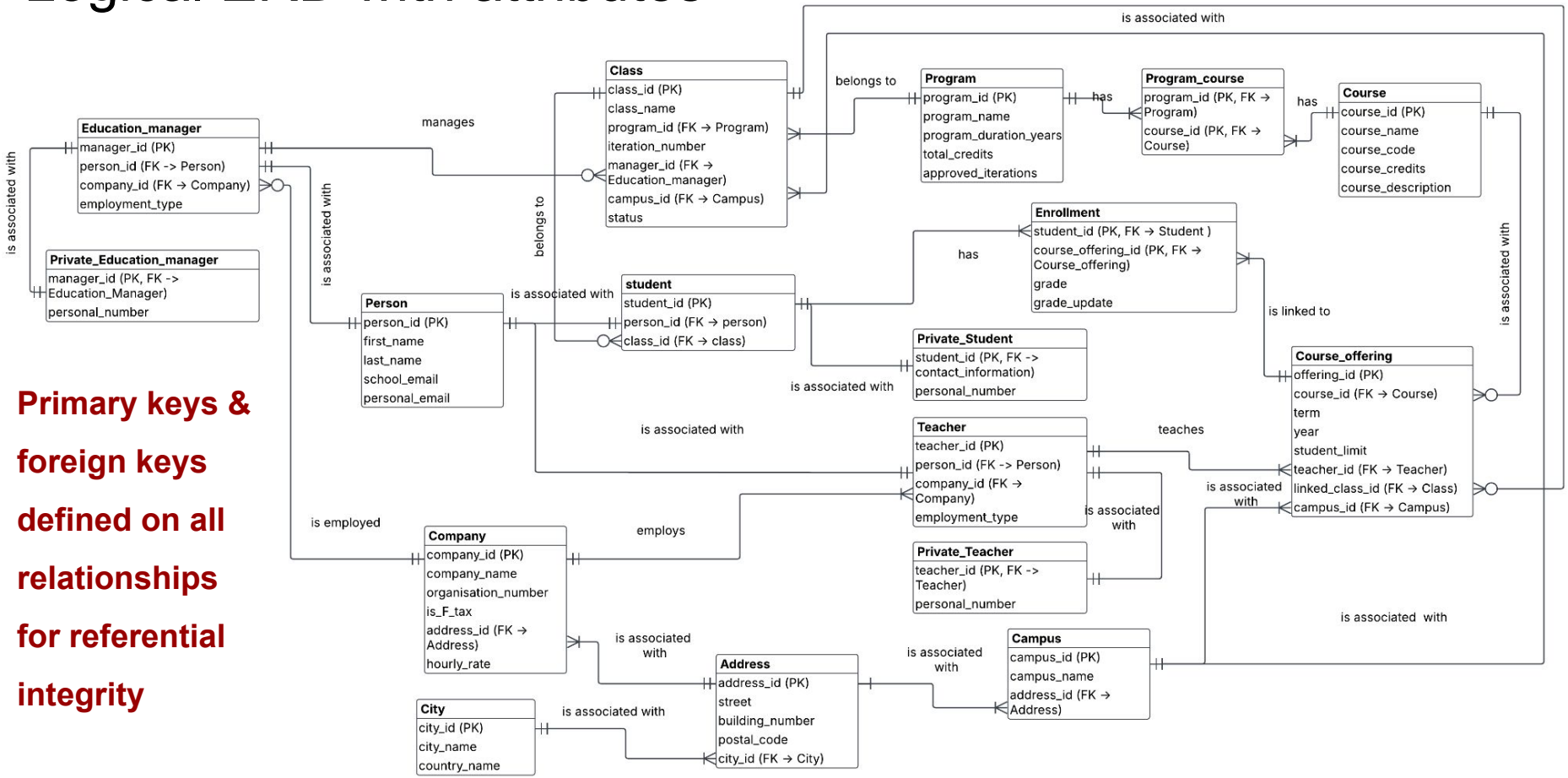
A Program can offer multiple ProgramCourses, with each ProgramCourse linking a single Program to a single Course.

Course ↔ ProgramCourse (One-to-Many)

A Course can be part of multiple ProgramCourses, but each ProgramCourse links a single Course to a single Program.



Logical ERD with attributes



Primary keys & foreign keys defined on all relationships for referential integrity

Normalization: 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
- ON DELETE logic to avoid orphaned records
- Use of ENUM data type for role, term, class status
- Trigger logic for business rules

Who & How uses the database

Role	Cares about	Useful queries
Education Manager: Oversees classes and manages communication between stakeholders.	● Teachers in each class ● Students contact information ● Students at risk (failing grades) ● Final report generation	● Get all teachers for a specific class ● Get emails of all students in a class ● List students with fail grades in current term ● List final grades for all students in a class
Teacher: Delivers content for specific course offering, grades students.	● Which courses they teach this term ● Who is enrolled in their courses ● Grading status ● Statistics on grades	● List all courses they are teaching in the current term ● Get a list of students per course ● Notify students missing grades ● See grading breakdown: how many students got VG/G/IG
VP / Executive / Management: Needs overviews to make strategic decisions.	● Course pass/fail rates ● Program performance ● Utilization of staff	● Pass rate per course ● Pass rate per class ● Breakdown of employment types of teachers ● How many courses were run each year per campus

Live System Preview (Video Pitch Only)

From Spreadsheets to Smart Solutions

When organizations have the right tools, they can focus on what they do best.

At YrkesCo, that means teaching — let the tools handle the rest.

Focus on what matters.
Teachers and administrators should spend their 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 handle grading and student interactions with ease

Less time managing. More time teaching. Leads to empowering students.