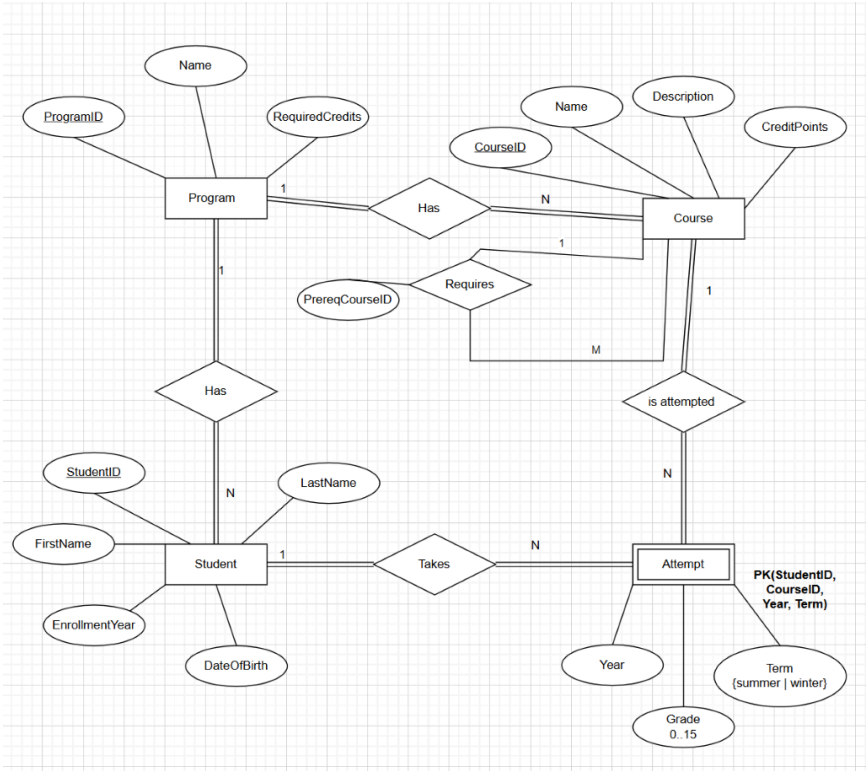
Lab1

Freitag, 31. Oktober 2025 16:57

Assignment 1: Student Information System

1) entity-relationship-model: ER model:



Entities & keys attributes :

- **Program**
  - **ProgramID** *(PK)*
  - **Name** *(unique)*
  - RequiredCredits
- **Course**
  - **CourseID** *(PK)*
  - Name
  - Description
  - CreditPoints
- **Student**
  - **StudentID** *(PK)*
  - FirstName
  - LastName
  - DateOfBirth
  - EnrollmentYear
- **Attempt** *(associative entity between Student and Course; stores extra data)* -----> **Attempt** as a **weak entity** (double rectangle) because its PK is made of FKs.
  - **StudentID** *(FK → Student)*
  - **CourseID** *(FK → Course)*
  - **Year**
  - **Term** *("summer" or "winter")*
  - Grade *(0…15)*

    - **Primary key suggestion:** *(StudentID, CourseID, Year, Term)* — allows re-taking a course in a new term.
- **Prerequisite** *(self-relationship on Course)*
    - **CourseID** *(FK → Course)* — the "target" course
    - **PrereqCourseID** *(FK → Course)* — the required earlier course
    - **Primary key:** *(CourseID, PrereqCourseID)*
      Rule: CourseID ≠ PrereqCourseID -----> **A course cannot be its own prerequisite.**

Relationships (cardinality, participation):

- **Program—Course**: *1 : N*
    - A Program **has** one or more Courses. *(Program 1..N, total on Course side)*
    - Each Course **belongs to** exactly one Program. *(Course participation = total)*
- **Program—Student**: *1 : N*
    - A Student **is enrolled in** exactly one Program. *(Student participation = total)*
    - A Program must have one or many Students. *(Program participation = total)*
- **Student—Course** via **Attempt**: *M : N* with attributes (Year, Term, Grade)
    - A Student can attempt many Courses (from **their own Program** only).
    - A Course can be attempted by many Students.
- **Course—Course** via **Prerequisite**: *M : N*
    - A Course can require zero, one, or many other Courses.
    - A Course can be a prerequisite for zero, one, or many Courses.

2) adding some additional composite, multivalued, or derived attributes in this example?

Composite attributes:

- **StudentAddress** = {Street, City, ZIP}

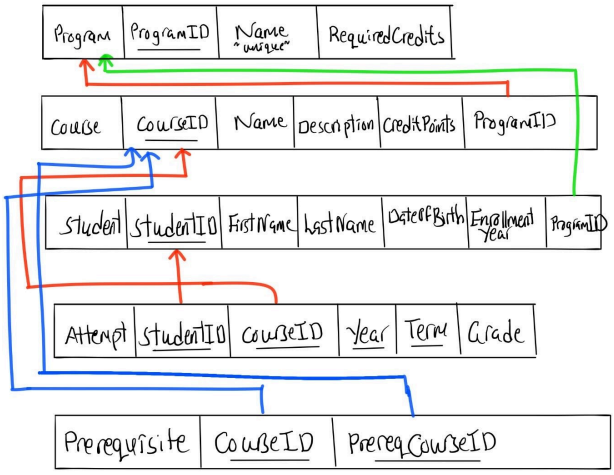
Multivalued attributes:

- **Student.PhoneNumbers** = {"+49…", "+1…"}
- **Student.Emails** = {school email, personal email}

Derived attributes (computed, not stored):

- **Student.Age** = today − DateOfBirth
- **Student.CreditsCompleted** = sum(CreditPoints of passed Attempts)

3) Relational Schema:



Note:

- Prerequisite forbids self-pairs: CourseID ≠ PrereqCourseID.

4-

Query | Query History

```sql
-- Enum for term
DO $$
BEGIN
  IF NOT EXISTS (SELECT 1 FROM pg_type WHERE typname = 'term_enum') THEN
    CREATE TYPE term_enum AS ENUM ('summer','winter');
  END IF;
END
$$;

-- ========== Program ==========
CREATE TABLE Program (
  ProgramID        SERIAL PRIMARY KEY,
  Name             VARCHAR(100) NOT NULL UNIQUE,
  RequiredCredits  SMALLINT     NOT NULL CHECK (RequiredCredits BETWEEN 0 AND 400)
);

-- ========== Course (each Course belongs to exactly one Program) ==========
CREATE TABLE Course (
  CourseID      SERIAL PRIMARY KEY,
  Name          VARCHAR(150) NOT NULL,
  Description   VARCHAR(1000),
  CreditPoints  SMALLINT     NOT NULL CHECK (CreditPoints BETWEEN 1 AND 30),
  ProgramID     INT          NOT NULL,
  CONSTRAINT fk_course_program
    FOREIGN KEY (ProgramID) REFERENCES Program(ProgramID)
);

-- ========== Student (each Student in exactly one Program) ==========
CREATE TABLE Student (
  StudentID      SERIAL PRIMARY KEY,
  FirstName      VARCHAR(60) NOT NULL,
  LastName       VARCHAR(60) NOT NULL,
  DateOfBirth    DATE        NOT NULL CHECK (DateOfBirth <= CURRENT_DATE),
  EnrollmentYear SMALLINT    NOT NULL CHECK (EnrollmentYear BETWEEN 1900 AND 2100),
  ProgramID      INT         NOT NULL,
  CONSTRAINT fk_student_program
    FOREIGN KEY (ProgramID) REFERENCES Program(ProgramID)
);

    FOREIGN KEY (ProgramID) REFERENCES Program(ProgramID)
);

-- ========== Attempt (Student-Course M:N with attributes; allows retakes) ==========
CREATE TABLE Attempt (
  StudentID  INT       NOT NULL,
  CourseID   INT       NOT NULL,
  Year       SMALLINT  NOT NULL CHECK (Year BETWEEN 1900 AND 2100),
  Term       term_enum NOT NULL,
  Grade      SMALLINT  CHECK (Grade BETWEEN 0 AND 15),
  PRIMARY KEY (StudentID, CourseID, Year, Term),
  CONSTRAINT fk_attempt_student
    FOREIGN KEY (StudentID) REFERENCES Student(StudentID),
  CONSTRAINT fk_attempt_course
    FOREIGN KEY (CourseID)  REFERENCES Course(CourseID)
);

-- ========== CoursePrerequisite (Course↔Course relationship table) ==========
CREATE TABLE CoursePrerequisite (
  CourseID        INT NOT NULL,   -- target course
  PrereqCourseID  INT NOT NULL,   -- required earlier course
  PRIMARY KEY (CourseID, PrereqCourseID),
  CONSTRAINT fk_cp_course
    FOREIGN KEY (CourseID)       REFERENCES Course(CourseID),
  CONSTRAINT fk_cp_required
    FOREIGN KEY (PrereqCourseID) REFERENCES Course(CourseID),
  CONSTRAINT prereq_no_self_reference CHECK (CourseID <> PrereqCourseID)
);
```

5)

```sql
-- Programs
INSERT INTO Program (ProgramID, Name, RequiredCredits) VALUES
  (1, 'Computer Science', 180),
  (2, 'Business IT',      180);

-- Courses (CS = ProgramID 1)
INSERT INTO Course (CourseID, Name, Description, CreditPoints, ProgramID) VALUES
  (1, 'Databases 1',     'Intro to relational databases',               5, 1),
  (2, 'Algorithms',      'Basic algorithms and complexity',             5, 1),
  (3, 'Web Development', 'HTML/CSS/JS and backend basics',              5, 1),
  (6, 'Programming 1',   'Intro to variables, control flow, functions', 5, 1);

-- Courses (Business IT = ProgramID 2)
INSERT INTO Course (CourseID, Name, Description, CreditPoints, ProgramID) VALUES
  (4, 'ER Modeling',     'Entity-Relationship modeling for databases',  5, 2),
  (5, 'Accounting',      'Introduction to financial accounting',        5, 2);

-- Students
INSERT INTO Student (StudentID, FirstName, LastName, DateOfBirth, EnrollmentYear, ProgramID) VALUES
  (1, 'Ebrahim', 'Al-Amoudi', DATE '2000-06-15', 2024, 2),
  (2, 'Anna',    'Schmidt',   DATE '2001-03-10', 2023, 1),
  (3, 'Max',     'Mueller',   DATE '2002-11-20', 2024, 1);

-- Course prerequisites (Course ↔ Course)
INSERT INTO CoursePrerequisite (CourseID, PrereqCourseID) VALUES
  (2, 6),  -- Algorithms requires Programming 1
  (3, 6),  -- Web Development requires Programming 1
  (1, 6);  -- Databases 1 requires Programming 1

-- Attempts (Student-Course with attributes)
-- Ebrahim (Program 2)
INSERT INTO Attempt (StudentID, CourseID, Year, Term, Grade) VALUES
  (1, 4, 2025, 'summer', 12),
  (1, 5, 2025, 'winter', 10);

-- Anna (Program 1)
INSERT INTO Attempt (StudentID, CourseID, Year, Term, Grade) VALUES
  (2, 6, 2023, 'winter', 14),
  (2, 1, 2024, 'summer', 13),
  (2, 2, 2025, 'winter', 11);

```