Lab 4 **CSI2532** **Le 7 Janvier 2021**

TA: Laith Grira

Par Giorgio Sawaya **300126961**

Question 1

Les professeurs peuvent enseigner le même cours sur plusieurs semestres et seule la plus récente doit être enregistrée.

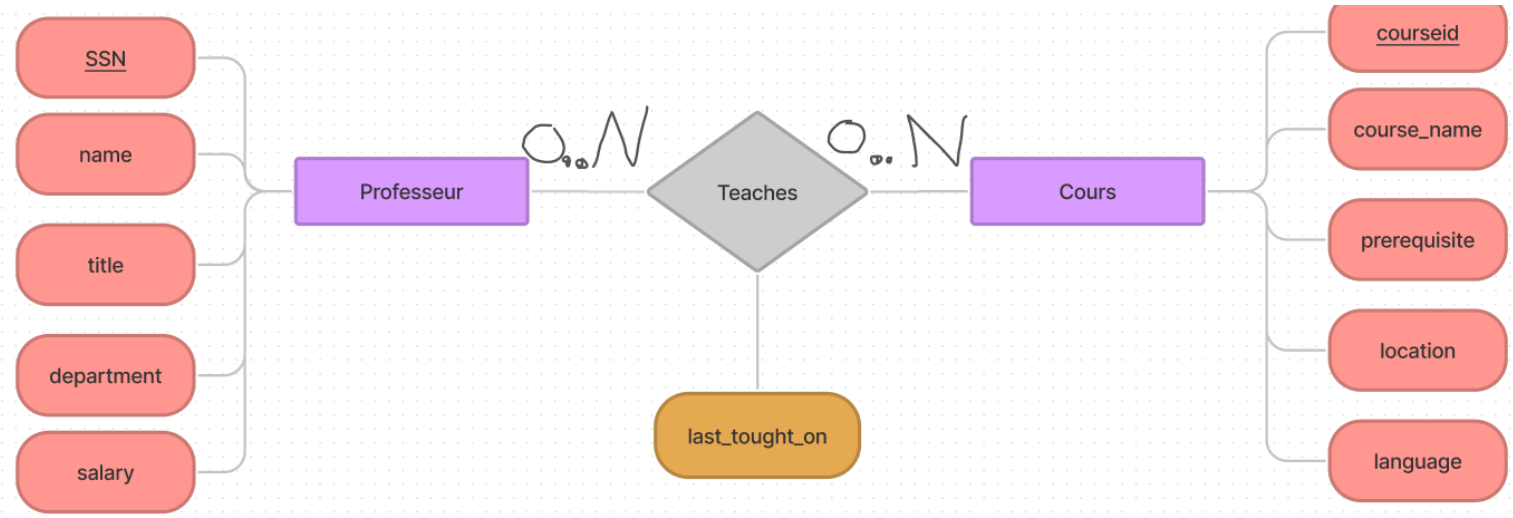
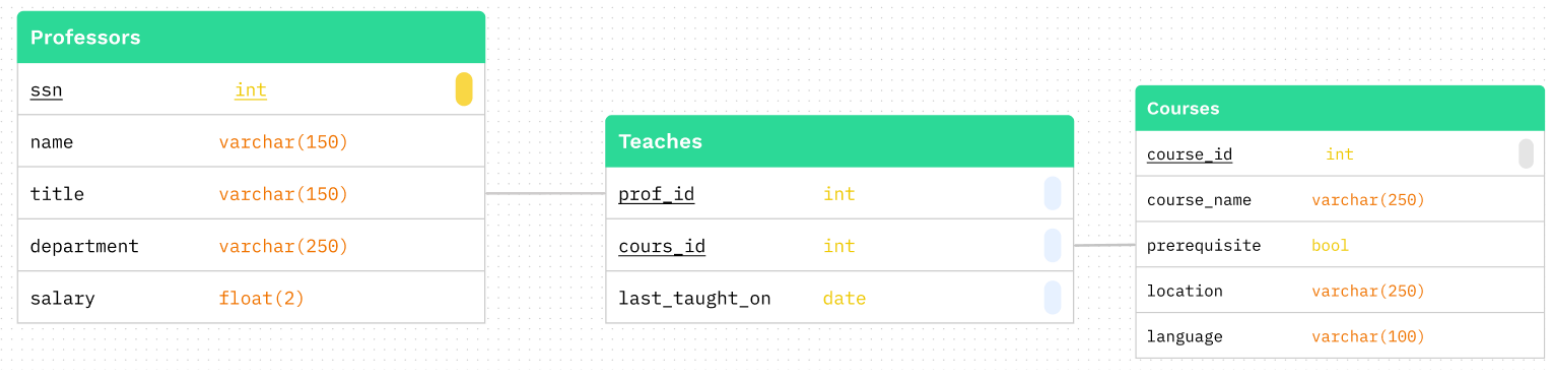


Diagramme relationnelle:



Commande SQL:

```
CREATE SCHEMA IF NOT EXISTS "University_lab4_Q1"
    AUTHORIZATION gsawa066;

COMMENT ON SCHEMA "University_lab4_Q1"
    IS 'This is the university schema for lab 4';


CREATE TABLE Professors (
  ssn int, name varchar(150),
  title varchar(100) NOT NULL,
  department varchar(250) NOT NULL,
  salary float(2),
  PRIMARY KEY (ssn)
);

CREATE TABLE Courses (
  cours_id int NOT NULL,
  course_name varchar(250),
  prerequisite bool,
  location varchar(250),
  language varchar(100),
  PRIMARY KEY (cours_id)
);

CREATE TABLE Teaches (
  prof_id int NOT NULL,
  cours_id int NOT NULL,
  last_taught_on date,
  PRIMARY KEY (prof_id, cours_id),
  FOREIGN KEY (prof_id) REFERENCES Professors(ssn),
  FOREIGN KEY (cours_id) REFERENCES Courses(cours_id)
);
```

Question 3

Chaque professeur enseigne exactement un cours (ni plus, ni moins).

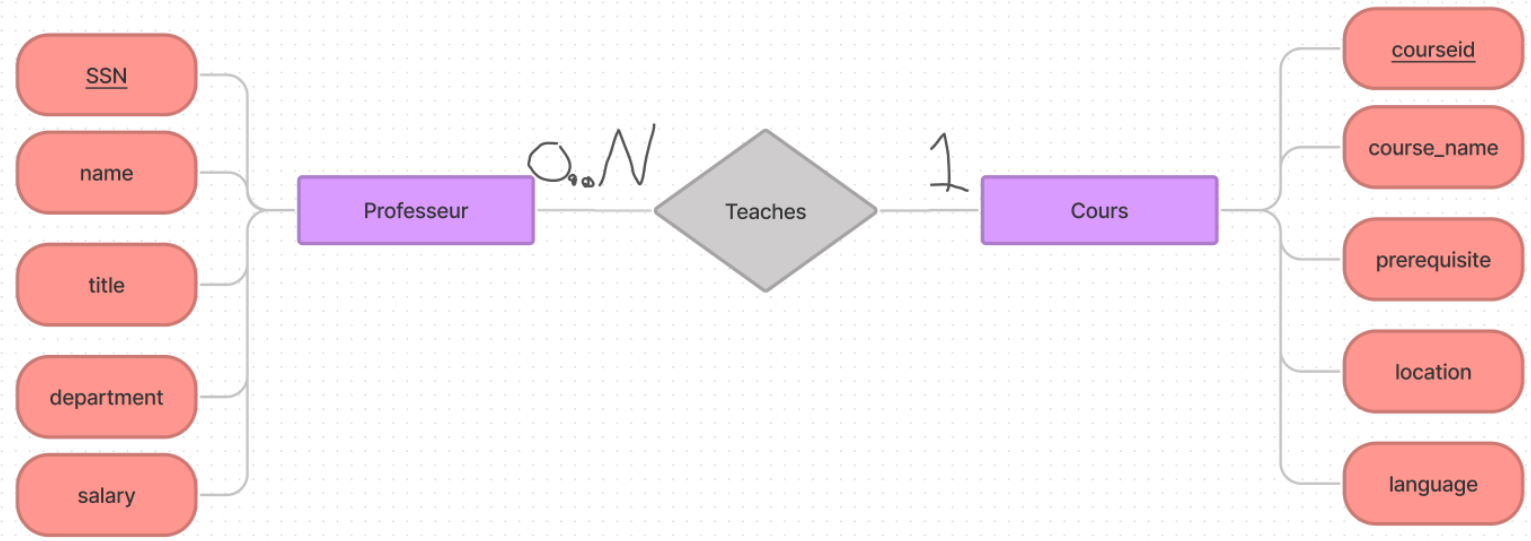
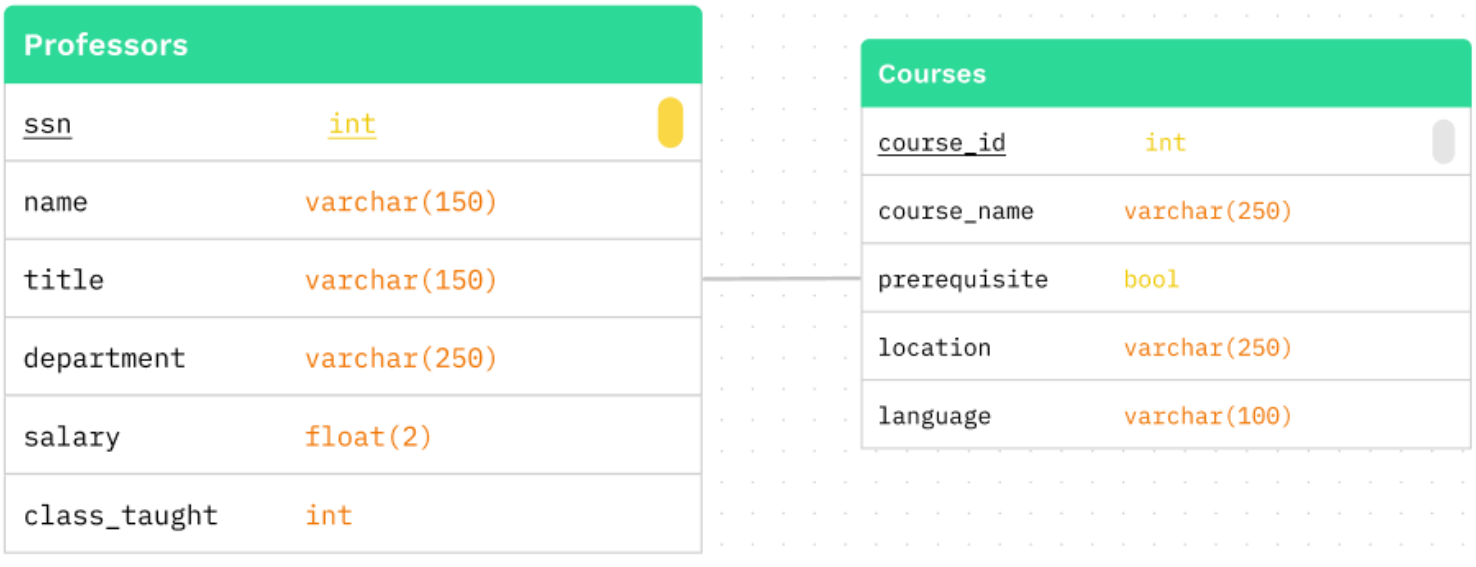


Diagramme relationnelle

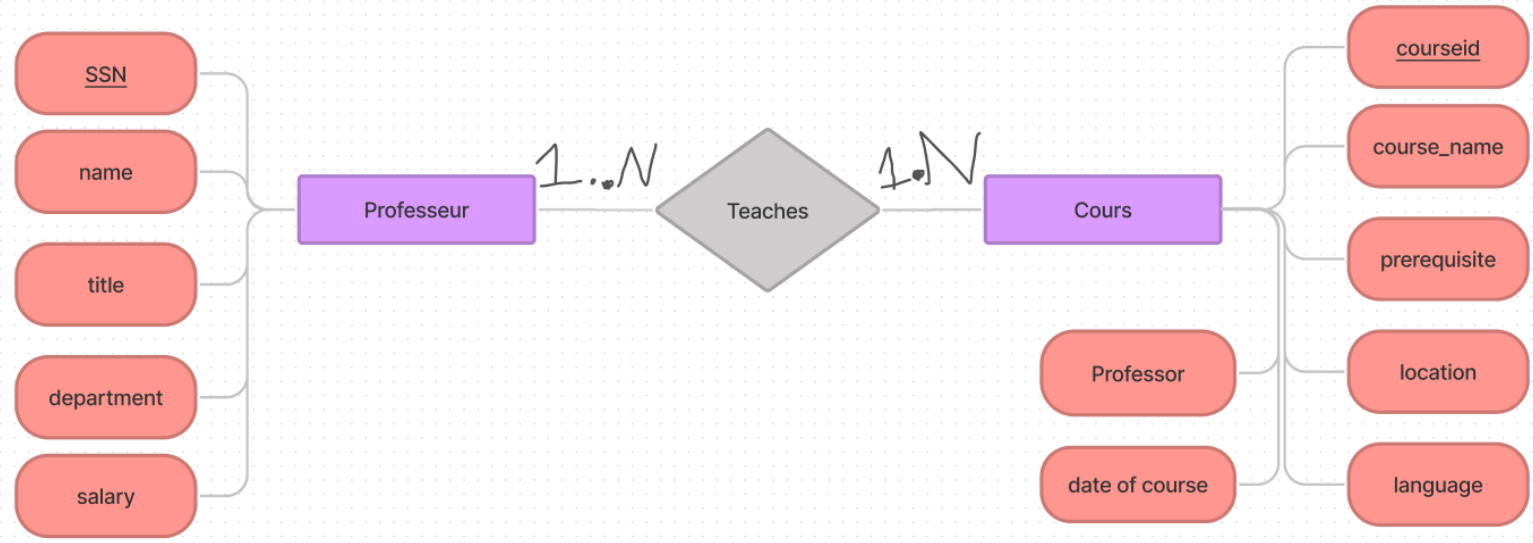


Commande SQL:

```
CREATE SCHEMA IF NOT EXISTS "University_lab4"
    AUTHORIZATION gsawa066;

COMMENT ON SCHEMA "University_lab4"
    IS 'This is the university schema for lab 4';

CREATE TABLE Courses  (
  cours_id int NOT NULL,
  course_name varchar(250),
  prerequisite bool,
  location varchar(250),
  language varchar(100),
  PRIMARY KEY (cours_id)
);

CREATE TABLE Professors  (
  ssn int,
   name varchar(150),
  title varchar(100) NOT NULL,
  department varchar(250) NOT NULL,
  salary float(2),
  class_taught int NOT NULL,
  PRIMARY KEY (ssn),
  FOREIGN KEY (class_taught) REFERENCES Courses(cours_id)
);
```

Question 5

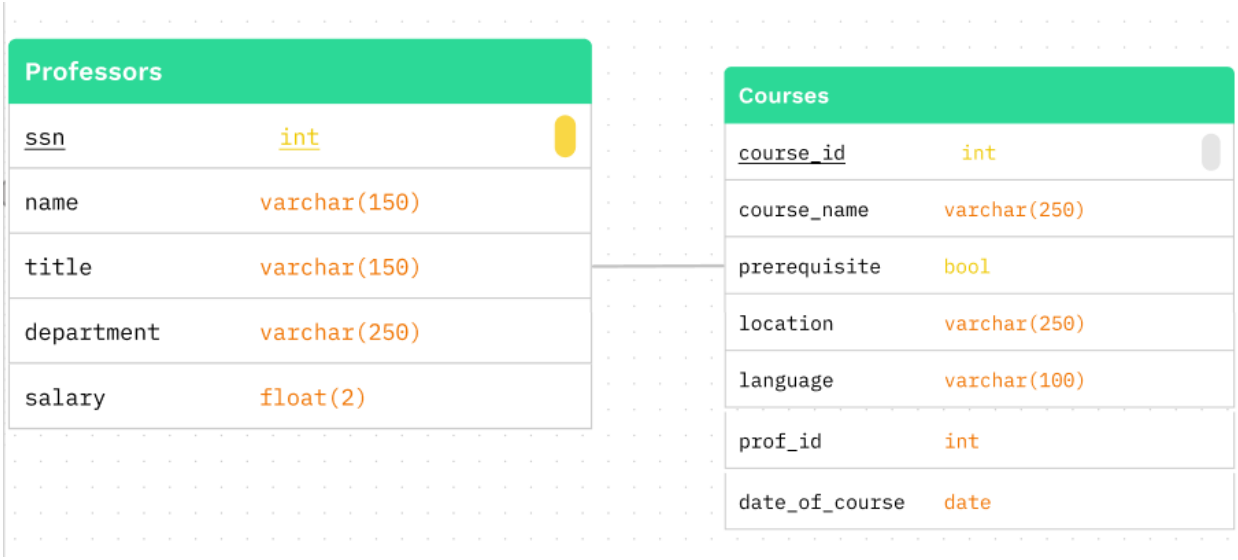
Les professeurs peuvent enseigner le même cours sur plusieurs semestres et chaque doit être enregistrée



*ERREUR DE MULTIPLICITÉ Le prof n'a pas besoin d'être associé à un cours et vice verca donc les vrai multiplicité sont 0-N A 0-n

Diagramme relationnelle:

Avec ce schéma , on peut créer un tuple course pour chaque cours qui a été donné par le même enseignant mais avec les dates differente:



Commande SQL:

```
CREATE TABLE Professors  (
  ssn int,
  name varchar(150),
  title varchar(100) NOT NULL,
  department varchar(250) NOT NULL,
  salary float(2),
  PRIMARY KEY (ssn),

);

CREATE TABLE Courses  (
  cours_id int NOT NULL,
  course_name varchar(250),
  prerequisite bool,
  location varchar(250),
  language varchar(100),
  prof_id int,
  date_of_course date,
  PRIMARY KEY (cours_id),
  FOREIGN KEY (prof_id) REFERENCES Professors(ssn)
);
```

Question 6

Supposons maintenant que certains cours puissent être enseignés conjointement par une équipe de professeurs, mais il est possible qu'aucun professeur dans une équipe ne puisse enseigner le cours. Modélisez cette situation en introduisant des ensembles d'entités et des ensembles de relations supplémentaires si nécessaire.

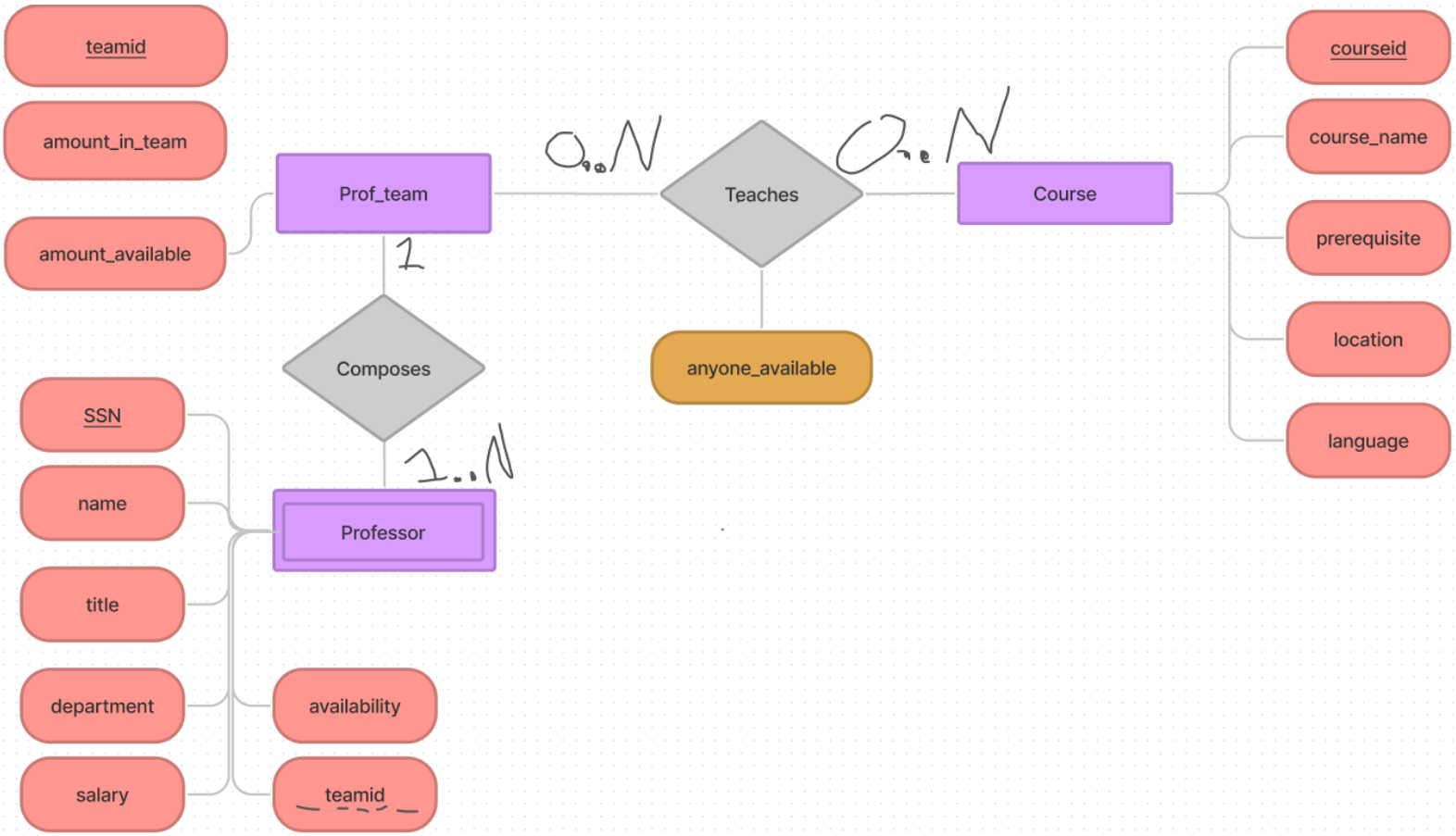
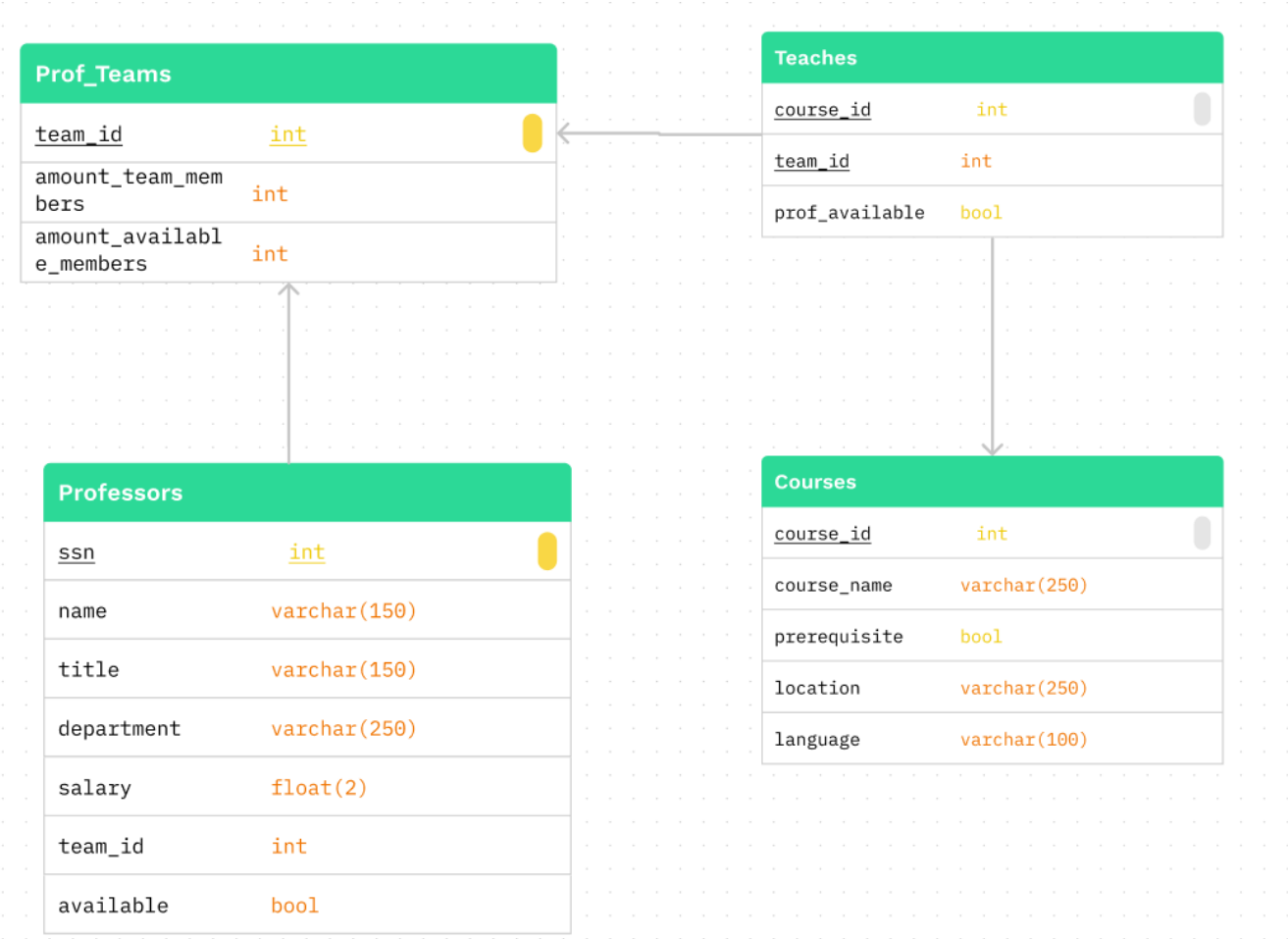


Diagramme relationnelle:



SQL QUERY:

```
CREATE SCHEMA IF NOT EXISTS "University_lab4"
    AUTHORIZATION gsawa066;

CREATE TABLE Professors  (
  ssn int,
  name varchar(150),
  title varchar(100) NOT NULL,
  department varchar(250) NOT NULL,
  salary float(2),
  team_id int,
  available bool,
  PRIMARY KEY (ssn),
  FOREIGN KEY (team_id) REFERENCES Prof_Teams(team_id)
);


CREATE TABLE Prof_Teams  (
  team_id int,
  amount_team_members, int
  amount_available_members int,
  PRIMARY KEY (team_id)
);

CREATE TABLE Courses  (
  cours_id int NOT NULL,
  course_name varchar(250),
  prerequisite bool,
  location varchar(250),
  language varchar(100),
  PRIMARY KEY (cours_id),
);

CREATE TABLE Teaches (
  team_id int NOT NULL,
  cours_id int NOT NULL,
  prof_available bool,
  PRIMARY KEY (team_id, cours_id),
  FOREIGN KEY (team_id) REFERENCES Prof_Teams(team_id),
  FOREIGN KEY (cours_id) REFERENCES Courses(cours_id)
);
```