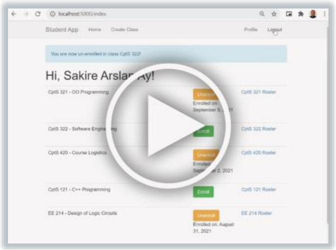
Building Data-Driven Web Apps with Flask and SQLAlchemy

Sakire Arslan Ay

March 6, 2023



You are now un-enrolled in class CptS 322!

Hi, Sakire Arslan Ay!

CptS 321 - OO Programming — Enrolled on: September 5, 2021 — CptS 321 Roster

CptS 322 - Software Engineering — CptS 322 Roster

CptS 420 - Course Logistics — Enrolled on: September 2, 2021 — CptS 420 Roster

CptS 121 - C++ Programming — CptS 121 Roster

EE 214 - Design of Logic Circuits — Enrolled on: August 31, 2021 — EE 214 Roster

SQLAlchemy

Flask

https://github.com/arslanay/GMU-IT-Lecture

Sakire Arslan Ay

You can call me Shakira.

Scholarly Associate Professor,

Coordinator for Software Engineering M.S. and B.S. programs

Cookie

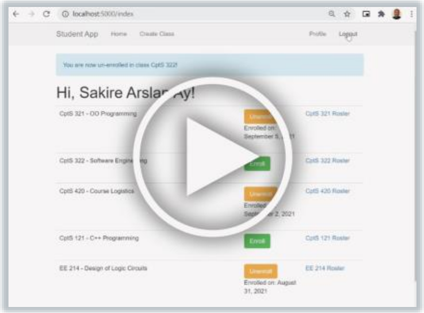
Snow

- Education:
  - Ph.D. in Computer Science – University of Southern California (2010)
  - M.S. in Computer Science – University of Southern California (2004)
  - B.S. in Computer Engineering – Bogazici University – Turkey (1999)
- Research Interests:
  - Large-Scale Geospatial Data Management and Indexing
  - Sensor-Rich Video Annotation and Search

- Courses at Washington State University
  - Software Engineering I
  - Programming Language Design
  - Introduction to Database Systems
  - Software Design Project I (Capstone)
  - Software Design Project II (Capstone)
  - Software Design

Review

- Web application development using Flask + SQLAlchemy
- Example application:
  - Student App



You are now un-enrolled in class CptS 322!

Hi, Sakire Arslan Ay!

CptS 321 - OO Programming — Enrolled on: September 5, 2021 — CptS 321 Roster

CptS 322 - Software Engineering — CptS 322 Roster

CptS 420 - Course Logistics — Enrolled on: September 2, 2021 — CptS 420 Roster

CptS 121 - C++ Programming — CptS 121 Roster

EE 214 - Design of Logic Circuits — Enrolled on: August 31, 2021 — EE 214 Roster

- Designing the Database for Student App
  - Entities; relations
  - Constraints on relations
    - One to one; one to many ; many to many
  - Creating the database model using SQLAlchemy + Python

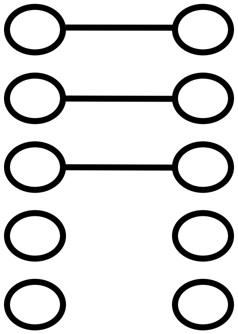
Outline for today's lecture

- Revise the current schema and customize it according to the application requirements.
- Add additional relations and constrains as needed
- Update the SQLAlchemy model and implement changes

Multiplicity (Key) Constraints for Relations

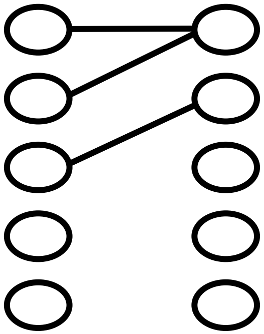
Consider binary relationship set R between entity sets **A** and **B**

- ***One to one:*** an entity in **A** is associated with at most one entity in **B**, and an entity in **B** is associated with at most one entity in **A**.
- ***Many to One***: An entity in **A** is associated with at most one entity in **B**, an entity in **B** is associated with many entities in **A**.
- ***Many to Many:*** An entity in **A** is associated with many entities in **B**, and an entity in **B** is associated with many entities in **A**.



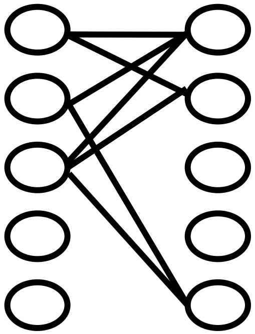
One-to-one

an *employee* has only one *spouse* in a *married-to* relationship



Many-to-one

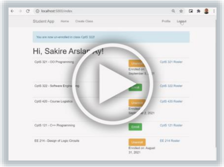
an *employee* works in a single *department* but a *department* consists of many *employees.*



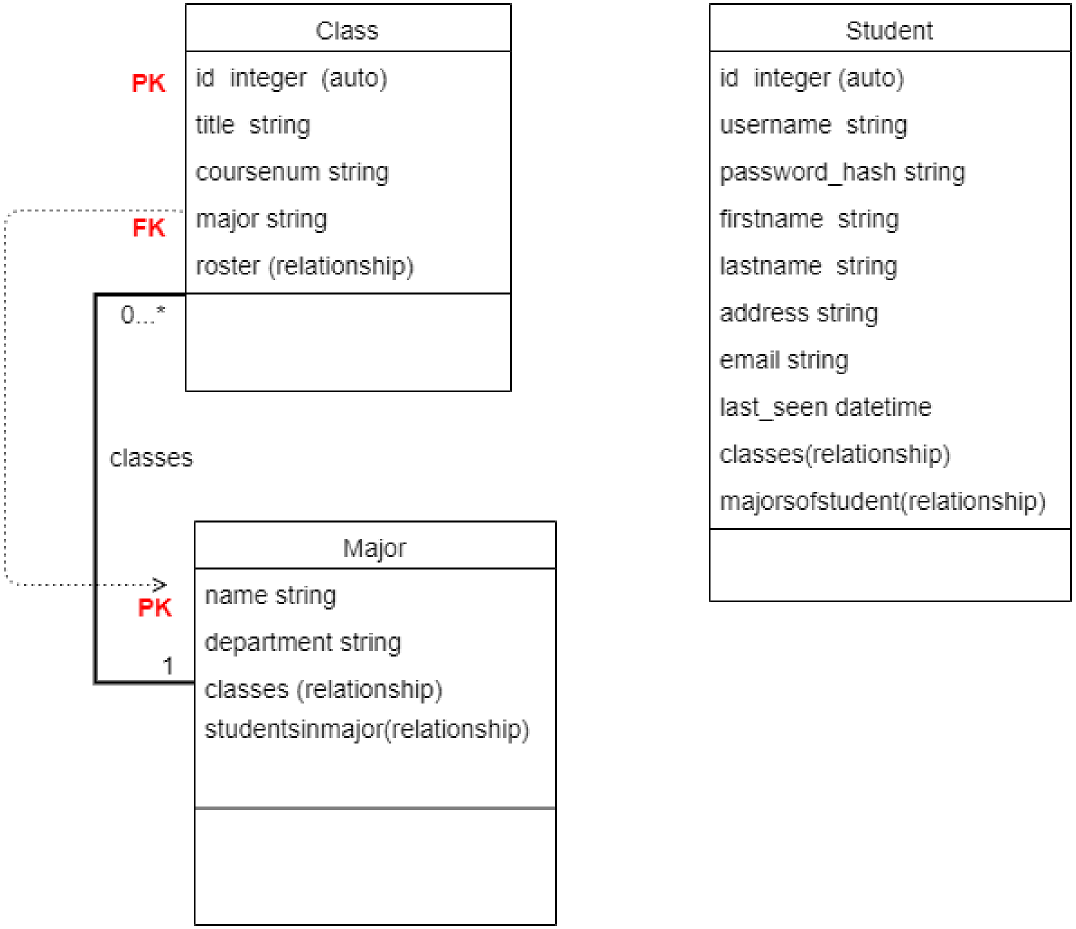
Many-to-many

A *customer* may have many *bank accounts*. *Accounts* may be joint between multiple *customers*

Student App – Database Schema



UML Class Diagram for Student App Database -- Version 1



Class

PK id integer (auto)

title string

coursenum string

FK major string

roster (relationship)

Student

id integer (auto)

username string

password_hash string

firstname string

lastname string

address string

email string

last_seen datetime

classes(relationship)

majorsofstudent(relationship)

Major

PK name string

department string

classes (relationship)

studentsinmajor(relationship)

0...*

classes

1

Entities

- Major
- Class
- Student

Relationships:

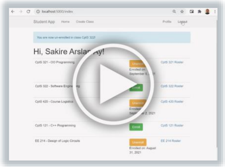
- Classes : one-to-many

  Example:

  CS major -> many classes

  CS322 -> one major

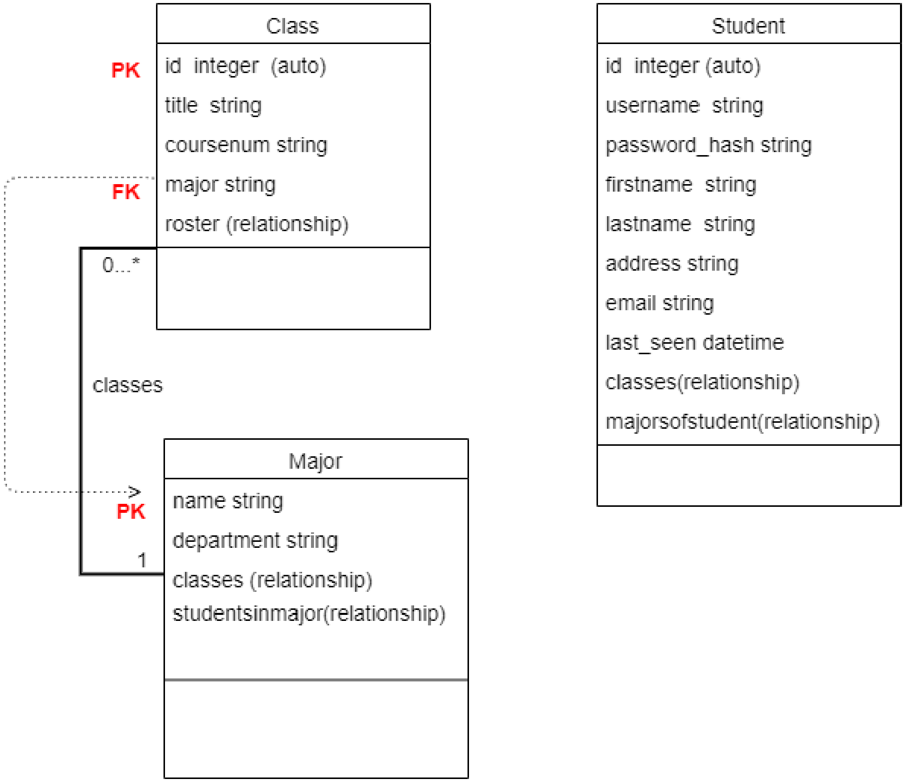
Student App – Database Schema



Observations:

- Classes and Students are related
  - Students enroll in classes
  - Class rosters include students
    - Many to many
    - A class may include many enrolled students
    - A student may enroll in many classes

UML Class Diagram for Student App Database -- Version 1

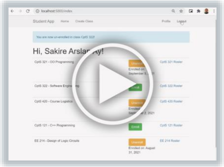


Class
- id integer (auto)
- title string
- coursenum string
- major string
- roster (relationship)

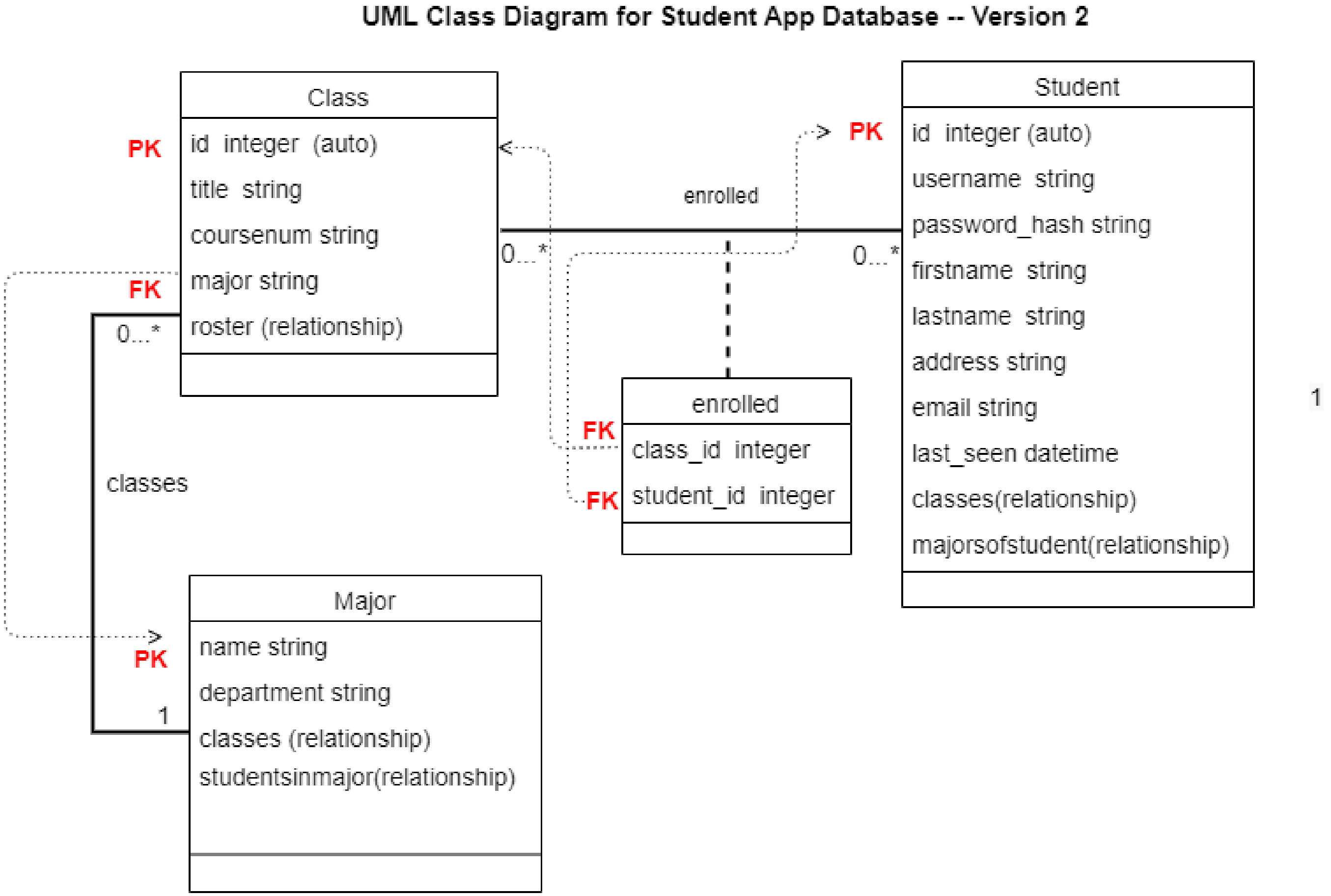
Major
- name string
- department string
- classes (relationship)
- studentsinmajor(relationship)

Student
- id integer (auto)
- username string
- password_hash string
- firstname string
- lastname string
- address string
- email string
- last_seen datetime
- classes(relationship)
- majorsofstudent(relationship)

Student App – Database Schema



- Add the many-to-many relationship “enrolled“ between Classes and Students (association table solution)



UML Class Diagram for Student App Database -- Version 2

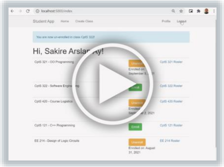
Many-to-many relationships in SQLAlchemy

- Option 1: Use a SQLAlchemy “association table”

```
enrolled = db.Table('enrolled',
    db.Column('studentid', db.Integer, db.ForeignKey('student.id')),
    db.Column('classid', db.Integer, db.ForeignKey('class.id'))
)
```

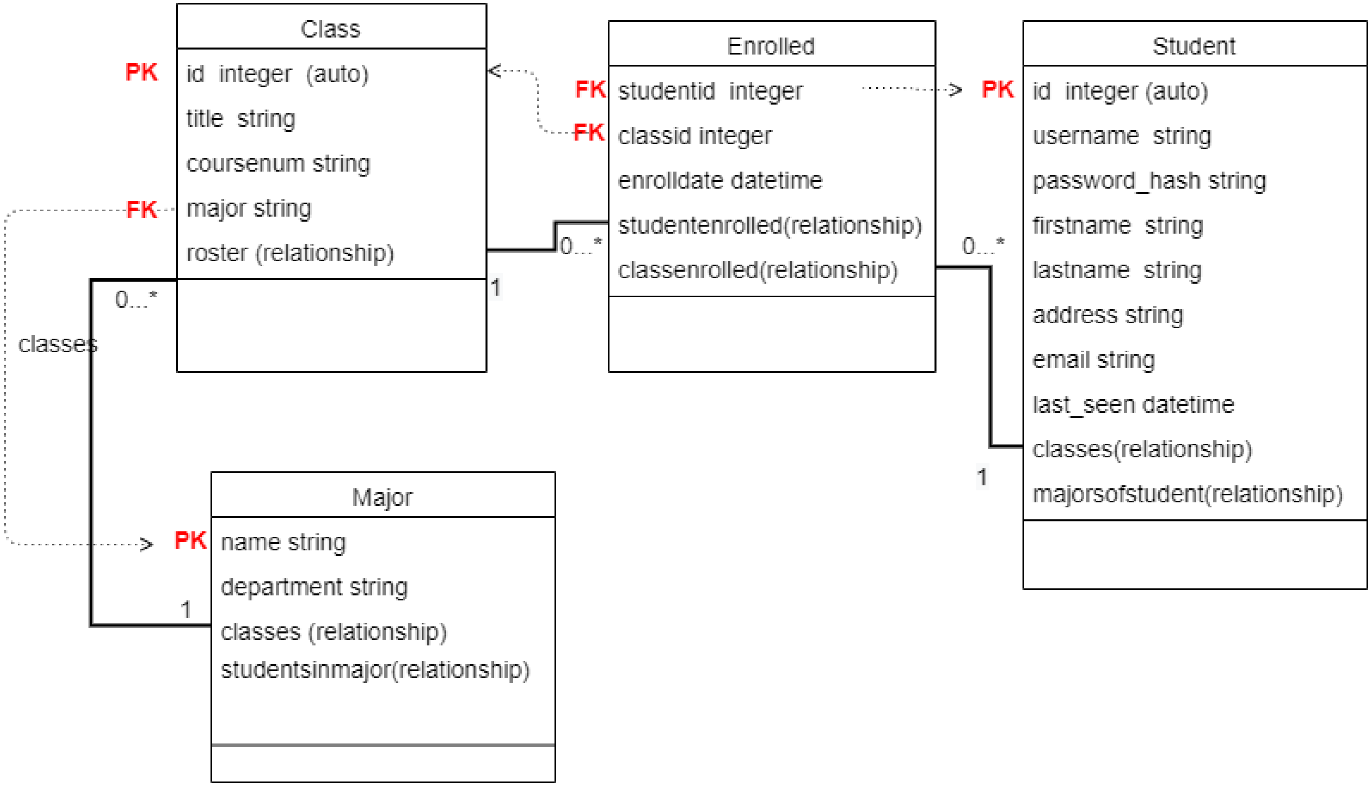
- Pros:
  - Core table object;
  - Easy to implement; easy to maintain (insertions and deletions); easy to query
- Cons:
  - Association table can’t have attributes beyond the foreign keys to participating entities
    - For example: can’t store the date of enrollment in the “enrolled” association table.
- Demo!

Student App – Database Schema



- Add the many-to-many relationship “enrolled“ between Classes and Students (association object solution)

UML Class Diagram for Student App Database -- Version 3



Class
- id integer (auto)
- title string
- coursenum string
- major string
- roster (relationship)

Enrolled
- studentid integer
- classid integer
- enrolldate datetime
- studentenrolled(relationship)
- classenrolled(relationship)

Student
- id integer (auto)
- username string
- password_hash string
- firstname string
- lastname string
- address string
- email string
- last_seen datetime
- classes(relationship)
- majorsofstudent(relationship)

Major
- name string
- department string
- classes (relationship)
- studentsinmajor(relationship)

Many-to-many relationships in SQLAlchemy

- Option 2: Use a SQLAlchemy “association object”

```
class Enrolled(db.Model):
    studentid = db.Column(db.Integer, db.ForeignKey('student.id’),
                                      primary_key=True)
    classid = db.Column(db.Integer, db.ForeignKey('class.id’),
                                      primary_key=True)
    enrolldate = db.Column(db.DateTime, default=datetime.utcnow)
    studentenrolled = db.relationship('Student')
    classenrolled = db.relationship('Class')
```

- Pros:
  - Association table can have attributes beyond the foreign keys to participating entities
    - E.g., can store the date of enrollment in the “enrolled” association object.
- Cons:
  - Queries on enrollment data will become more complex
    - E.g., to retrieve the students in class, need to first fetch the enrollment objects associated for the class, then retrieve the student associated with that enrollment.
- Demo!