Northeastern University
Khoury College of Computer and Information Sciences

ASSIGNMENT I
DATA MODELING

Name: Ravi Voleti

Class: CS5200 Database Management Systems.

SOLUTION:

STAGE I: INITIAL UNDERSTANDING OF THE PROBLEM STATEMENT.

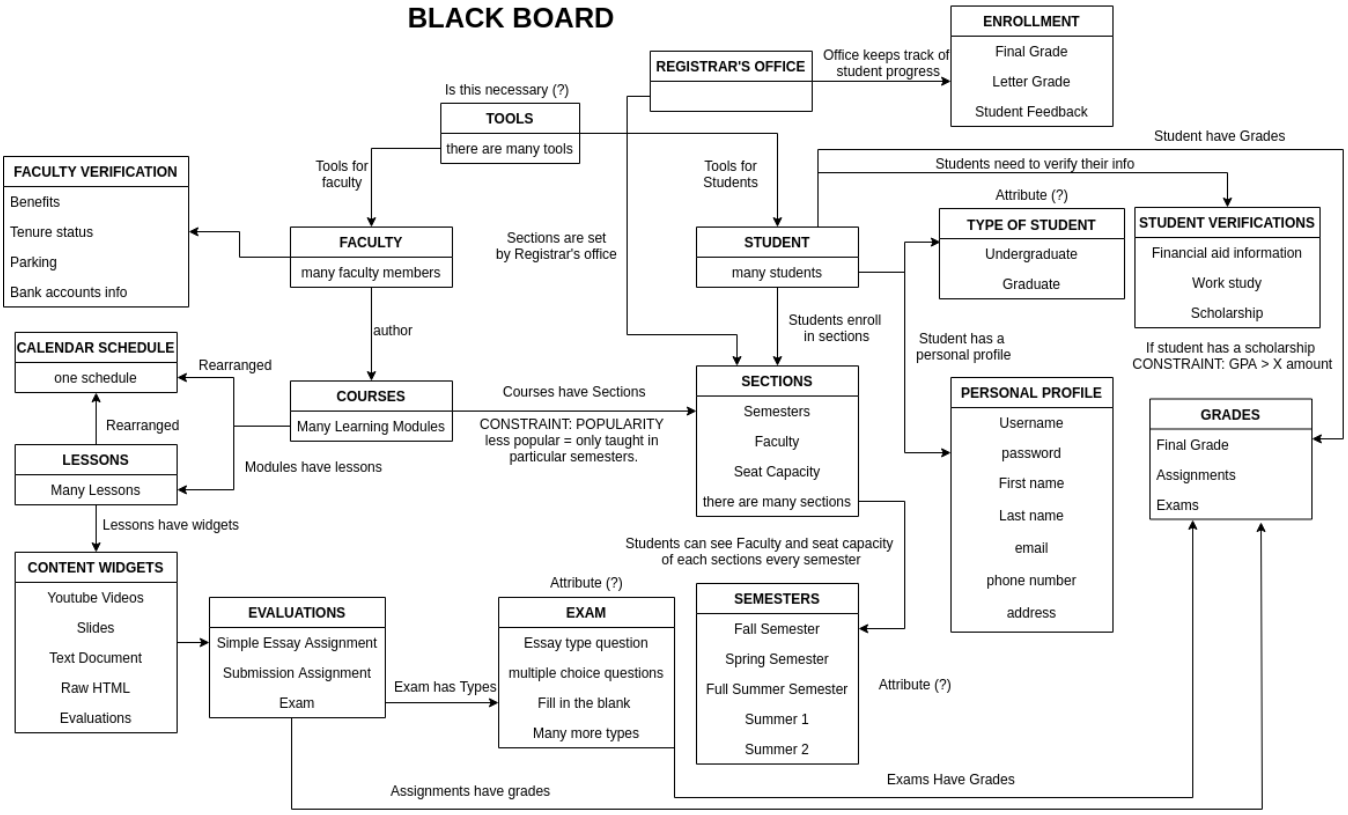


Fig.1 Initial Design UML

1. NOUNS THAT CAN BE CANDIDATE CLASSES OR ATTRIBUTES:

Student, Faculty, Tools, Courses, Sections, topic, Semesters, Undergraduate, Graduate, username, password, First name, Last name, email, Phone number, address, Seat capacity, Financial aid, Work study, Scholarship, Final Grade, Letter Grade, Student feedback, Calendar schedule, Lessons, Widgets, YouTube, Slides, Text, HTML, assignment, submission, exam, Modules, Benefits, Tenure, Parking, Bank account.

During the process of creating this UML there were many ideas that I felt were a core part of the actual design of the database system. However, it soon became obvious that some of these ideas never converged to a practical solution.

To elaborate on this, take the instance of **Fig.2.** I added Tools as a class which references Faculty and Student classes that has no meaningful impact in the long term.

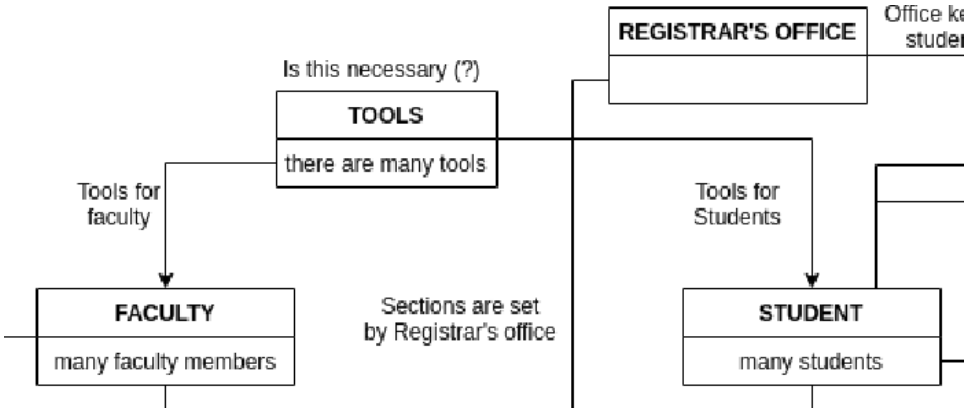


Fig.2. Tools

2. VERBS AS CANDIDATE RELATIONS BETWEEN CLASEES:

Author, rearranged, can be, evaluate, create, enroll, teach, track, verify, update, go.

Apart from these there were also a few more verbs that were used to create a logical relation with other classes such as “has a”, “have” to describe the attributes that were created separately from the class during the first visualization.

3. GENERALIZATIONS/ SPECIALIZATION:

At this stage there are no generalizations or inheritance done yet.

4. ASSOCIATIONS, AGGREGATION AND/OR COMPOSITIONS.

Some associations, aggregations and /or composition have realized at this stage but there is still a requirement for refinement.

5. CLASSES vs. ATTRIBUTE ANALYSIS

There were instances as seen in **Fig.3.** and **Fig.4.** where I created additional classes for attributes that could be merged into their corresponding classes rather than be left alone as individual classes.

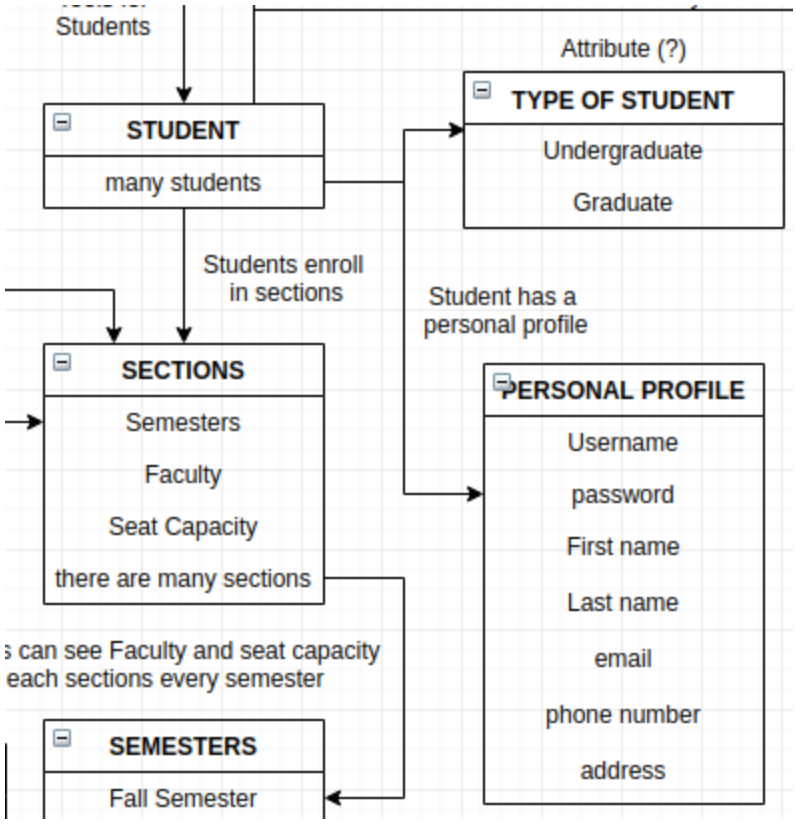


Fig.3. Additional classes for attributes (a).

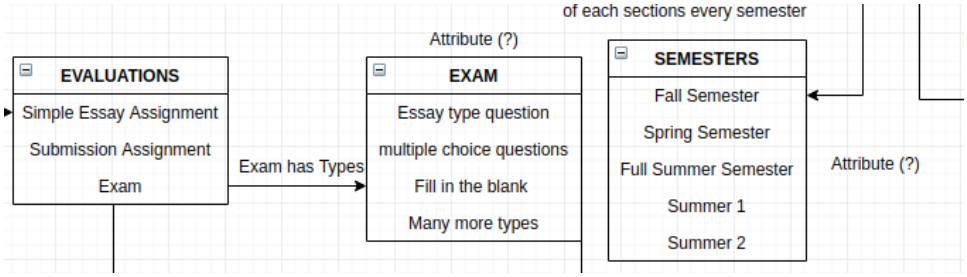


Fig.4. Additional classes for attributes (b).

6. CORRECT DATA TYPES

No data types have been assigned at this stage, however there is a clear understanding on what kind of data each class attribute can hold.

7. CARDINALITY

The type of cardinality for attributes in each class has been recognized however it can be properly established in a future stage where attributes and classes have been completely resolved.

8. REMOVAL OF REDUNDANT RELATIONS

No redundant relations have been removed yet.

9. REIFY

The classes have not been reified as of now.

… continue to SECTION II.

STAGE II: REFINING THE UML.

POTTENTIAL CLASSES AND ATTRIBUTES USED TO CREATE UML:

STUDENT (***student id***, type of student, username, password, first name, last name, financial aid information, work-study, scholarship).

EMAIL (***email id***, ***student id***, email).

NUMBER (***number id***, ***student id***, phone number).

ADDRESS (***address id***, ***student id***, address).

ENROLL (***enroll id***, ***student id***, ***section id***).

SECTION (***section id***, ***course id***, ***faculty id***, type of semester, seat capacity, year offered).

FACULTY (***faculty id***, faculty name, benefits, tenure status, parking, bank accounts information).

ASSISTANT (***assistant id***, ***student id***, ***faculty id***, ***course id***, office hours, office location).

COURSE (***course id***, ***faculty id***, course name, ***module id,*** module name, popularity).

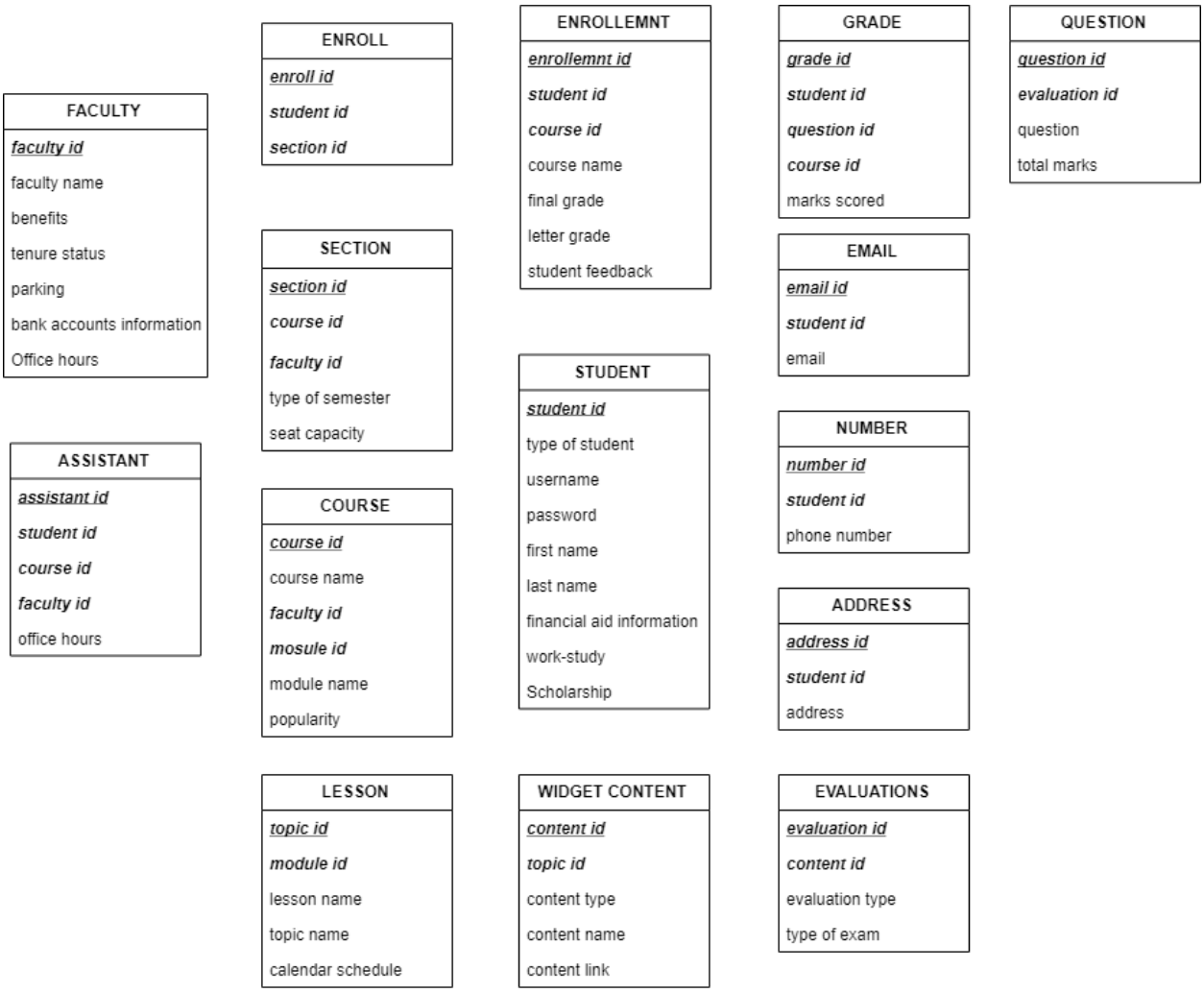
LESSON (***topic id***, module id, lesson name, topic name, calendar schedule).

WIDGET CONTENT (***content id***, ***topic id***, content Type, content name, content link).

EVALUATION (***evaluation id***, ***content id***, Evaluation type, type of exam).

ENROLLMENT (***enrollment id***, ***student id***, ***course id***, final grade, letter grade, student feedback).

GRADE (***grade id***, ***student id***, ***course id, evaluation id,*** question link, attempted answer link, marks scored, total marks, grade received).



1. NOUNS THAT CAN BE CANDIDATE CLASSES OR ATTRIBUTES:

Student, email, number, address, enroll, section, faculty, course, lesson, widget content, widget YouTube, widget slides, text document, evaluation, popularity, enrollment, grade, question, username, password, first name, last name, financial aid information, work-study, scholarship, email, phone number, address, type of semester, seat capacity, faculty name, benefits, tenure status, parking, bank accounts information, course name, module name, popularity, topic name, calendar schedule, content type, type of exam, final grade, letter grade, student feedback, marks scores, total marks, assistant, office hours, grade received, question link, attempted answer link, office location, grade received, GPA.

2. VERBS AS CANDIDATE RELATIONS BETWEEN CLASEES:

faculty **author** courses, student **enroll** sections, faculty **teach** section, students **watch** GPA (final grade), faculty **have** assistants, students **go** to office, students **evaluate** questions, student **have** emails, phone numbers and addresses, sections **have** students, faculty **have** assistants, courses **have** grades, courses **have** questions, marks, widgets **have** evaluations.

3. CLASSES vs. ATTRIBUTE ANALYSIS

Looking at the initial table we can immediately say that there are attributes that can be merged into an already existing class. Class such as **TOOLS** can be removed as it does not affect the final output in any meaningful way.

The Student class can include all unique attributes from existing **PERSONAL PROFILE** and **TYPE OF STUDENT.**

STUDENT (student id, type of student, username, password, first name, last name).

TYPE OF STUDENT can include Undergraduate or Graduate.

Additionally, we can add verification details to **STUDENT** class as they are unique values too.

STUDENT (student id, type of student, username, password, first name, last name, financial aid information, work-study, scholarship).

To accept multiple field entries for each student we can create new classes called **EMAIL, NUMBER, ADDRESS** which can hold multiple values of email, phone number and address.

EMAIL (student id, email).

NUMBER (student id, phone number)

ADDRESS (student id, address)

Students can enroll into a different section of a course, so we need to create a class **ENROLL** which keeps track of **STUDENT** and **SECTION**.

ENROLL (student id, section id).

FACULTY class can contain verification details like **STUDENT** as the details are unique to each member.

FACULTY (faculty id, benefits, tenure status, parking, bank accounts information).

FACULTY can author specific courses therefore course is an important member. Therefore, we need to include some way to incorporate courses. This can be done by adding the course id to **FACULTY**.

FACULTY (faculty id, benefits, tenure status, parking, bank accounts information)

COURSE has many modules which in turn have many lessons. And faculty can author courses which can change every year and therefore, to store every individual record of **COURSE** and **FACULTY** I have decided to keep faculty id in **COURSE** rather than course id in **FACULTY**.

Since there are some courses which are more popular than others, we need to impose some constraints by adding popularity attribute in the class **COURSE**.

COURSE (course id, faculty id, course name, module id, module name, popularity).

LESSON is a class with multiple lessons with respect to each module and can be rearranged using a calendar schedule.

LESSON (module id, lesson name, topic id, topic name, calendar schedule).

Each **LESSON** has multiple topics with different types widget content.

WIDGET CONTENT (topic name, content Type, content name, content id).

In **WIDGET CONTENT** there can be multiple content types: YouTube, slides, text document, Raw HTML, EVALUATION and there can be multiples of each of these content types. So, I thought we could have multiple Classes which can represent this data.

WIDGET CONTENT (content id, topic id, content Type, content name).

WIDGET YOUTUBE (content id, serial number, video link, video name).

WIDGET SLIDES (content id, serial number, slide name).

TEXT DOCUMENT (content id, serial number, document name).

RAW HTML (content id, serial number, html file link).

However, due to an afterthought I realized that this approach is incorrect, this table can be simplified by adding content link in the class **WIDGET CONTENT** and making another class **EVALUATION** which can be mapped using the unique key content id.

WIDGET CONTENT (content id, topic id, content Type, content name, content link).

EVALUATION can have evaluation id, Evaluation type (which is comprised of simple essay assignment, a submission assignment or an exam) and type of exam (which has multiple attributes such as Essay type question, multiple choice question, fill in the blanks, etc.). This attribute type of exam can also be null if evaluation type is not an exam.

EVALUATION (content id, evaluation id, Evaluation type, type of exam).

Office creates a section for a course for a given semester. Each **SECTION** can have different types of semesters such as fall, spring, full summer, summer 1 and summer 2.

SECTION (section id, course id, faculty id, type of semester, seat capacity, year offered).

Registrar's office keeps track of student final grade in enrollment. Since Students can see their final grades for a **COURSE** they were enrolled in, we need to identify each student's grade uniquely in **ENROLLMENT**.

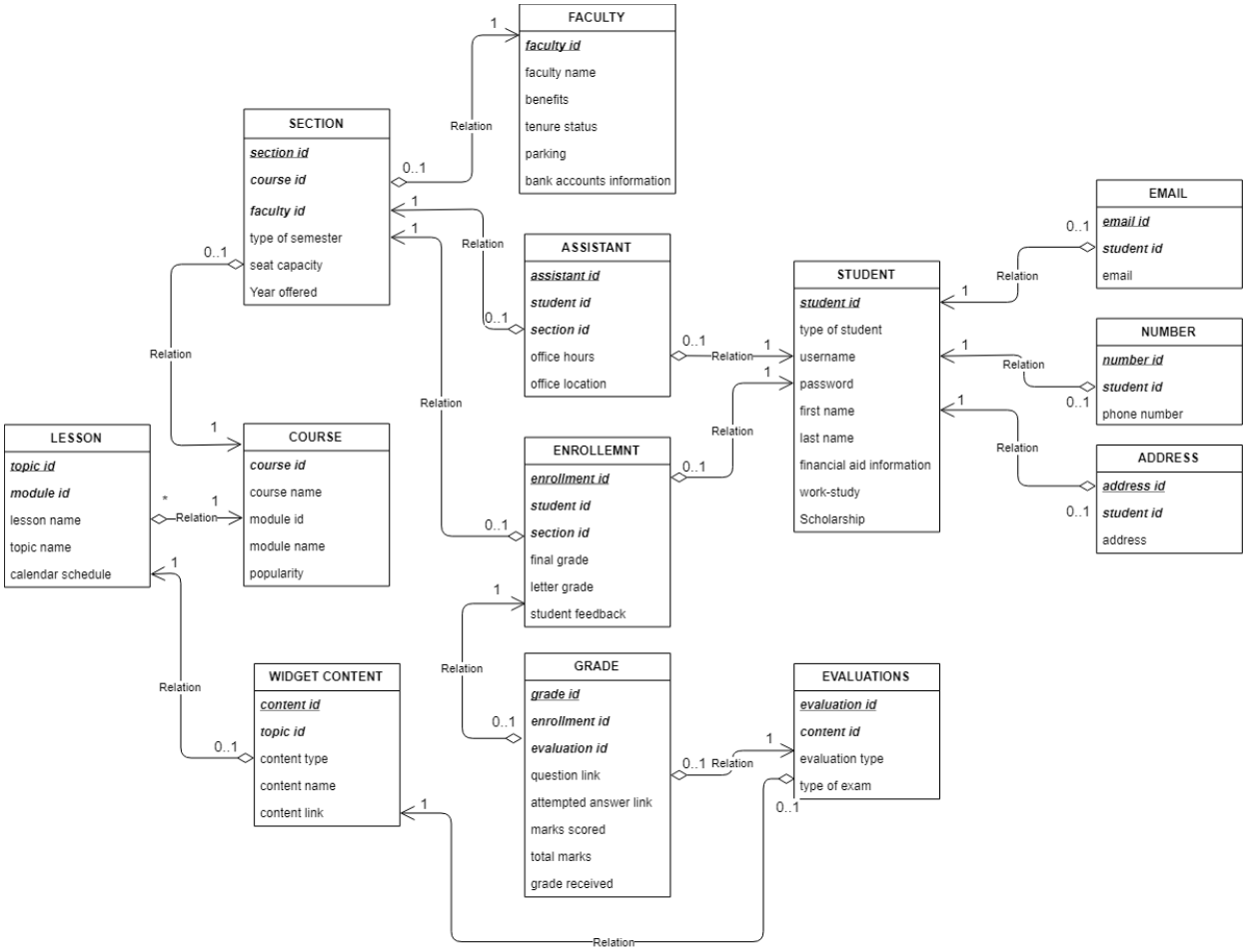
ENROLLMENT (enrollment id, student id, course id, final grade, letter grade, student feedback).

STUDENT can see every question of all assignments submitted. So, we need to track the marks scored for all types of evaluations, exact question and the answer attempted by the student.

GRADE (grade id, student id, course id, evaluation id, question link, attempted answer link, marks scored, total marks, grade received, GPA).

Since, students can go during office hours to review an evaluation in **GRADE**, therefore I created another class called **ASSISTANT** which tracks the professor and their student assistants for each course. With this we can assert the office timings and its location of both the teaching assistant and the professor.

ASSISTANT (assistant id, student id, faculty id, course id, office hours, office location).



4. REMOVAL OF REDUNDANT RELATIONS

Now paying close attention to all the classes we can remove some classes and change them into attributes, for example:

1. From **COURSE** I removed section id because it was redundant.
2. Remove **ENROLL** class and add its contents to **ENROLLMENT** remove course id due to redundancy, add section id so that it can connect to the **COURSE** and **FACULTY**.
3. **ASSISTANT** is can be directly connected to **SECTION** rather than to **COURSE** and **FACULTY** by removing course id and faculty id.
4. I have removed **EVALUATION** and added its contents to the class **GRADE** as EVALUATION was only providing redundant values rather than proving useful for the overall design.

5. REIFY

I have added another class called **MODULE** would help simplify things, as there can be multiple modules with different lessons. These lessons can have topics which in turn have widgets.

Between **LESSON** and WIDGET CONTENT I have added TOPIC class to further have a clear idea.

6. GENERALIZATIONS/ SPECIALIZATION:

STUDENT has **EMAIL, PHONE NUMBER** and **ADDRESS. STUDENT** enrolls in **ENROLLMENT.**

ENROLLMENT has **GRADE** and **SECTION**

SECTION has **ASSISTANT, FACULTY** and **COURSES**

COURSE has **MODULE**

MODULE has **LESSONS**

LESSONS have **TOPIC**

TOPIC have **WIDGET CONTENT**

7. CORRECT DATA TYPES

STUDENT
type of student: String varchar
username: String varchar
password: String varchar
first name: String varchar
last name: String varchar
financial aid information: Numeric float.
work-study: Binary.
Scholarship: Numeric float.

EMAIL
Email id: Numeric int
Student id: Numeric int
Email: String varchar.

NUMBER
Number id: Numeric int
Student id: Numeric int
Phone number: Numeric int.

ADDRESS
Address id: Numeric int
Student id: Numeric int
Address: String text

FACULTY
Faculty id: Numeric int
Faculty name: String char
Benefits: String varchar
Tenure status: Date
Parking: Numeric int
Bank account details: Numeric int

ASSISTANT
Assistant id: Numeric int
Student id: Numeric int
Section id: Numeric int
Office hours: Time.
Office location: String varchar.

ENROLLMENT
Enrollment id: Numeric int
Student id: Numeric int
Section id: Numeric int
Final grade: String char
Letter grade: String char
Student feedback: String text.

GRADE
Grade id: Numeric int
Enrollment id: Numeric int
Content id: Numeric int
Question link: String varchar
Attempted answer link: String varchar
Marks scored: Numeric int
Total marks: Numeric int
Grade received: String char
Evaluation type: enumeration
Type of exam: enumeration

SECTION
Section id: Numeric int
Course id: Numeric int
Faculty id: Numeric int
Type of semester: enumeration
Seat capacity: Numeric int
Year offered: Date

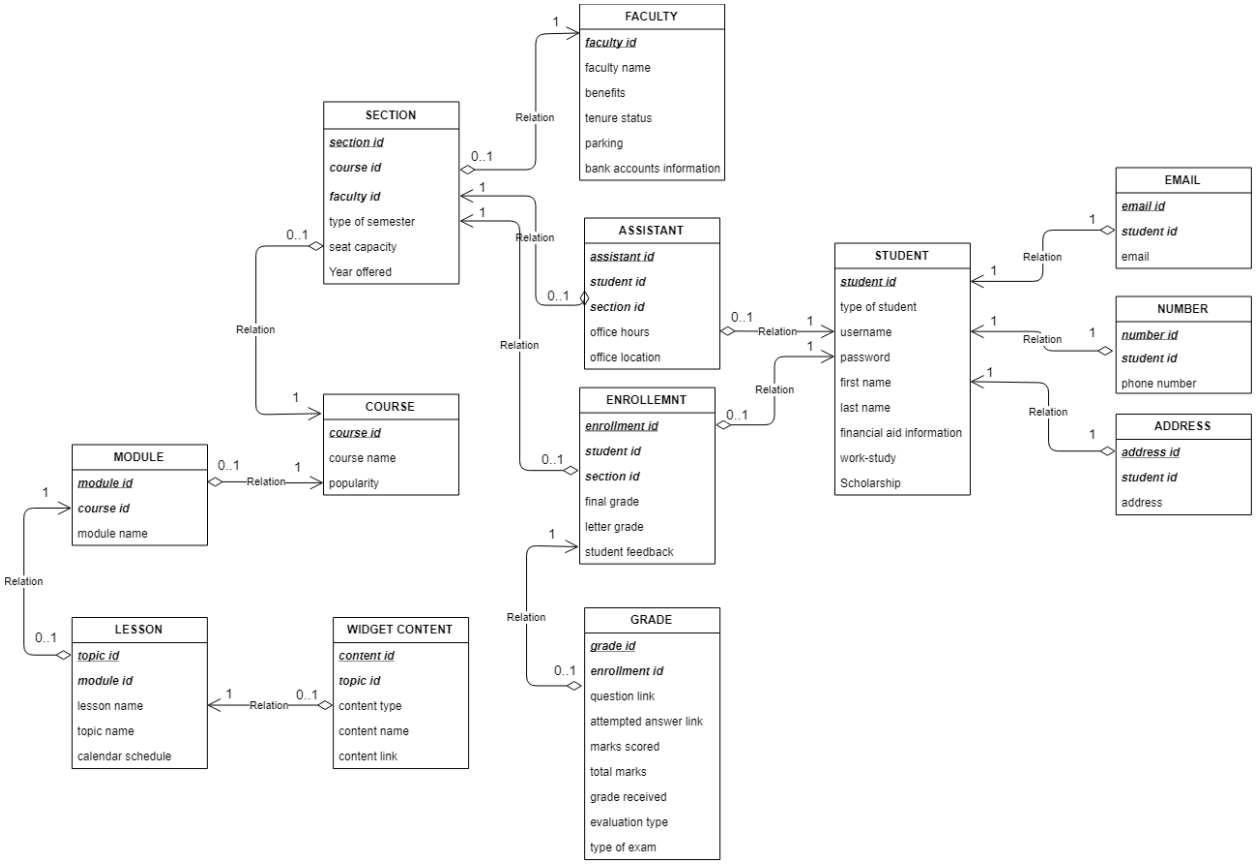
COURSE
Course id: Numeric int
Course name: String varchar
Popularity: Numeric int

WIDGET CONTENT
Content id: Numeric int
Topic id: Numeric int
Content type: enumeration
Content name: String varchar
Content link: String varchar

MODULE
Module id: Numeric int
Course id: Numeric int
Module name: String varchar

LESSON
Lesson id: Numeric int
Module id: Numeric int
Lesson name: String varchar
Calendar schedule: Date

TOPIC
Topic id: Numeric int
Lesson id: Numeric int.
Topic name: String varchar



8. ASSOCIATIONS, AGGREGATION AND/OR COMPOSITIONS.

There are many associations and compositions in this UML.

COMPOSITIONS:

1. **EMAIL** and **STUDENT.**
2. **ADDRESS** and **STUDENT.**
3. **PHONE NUMBER** and **STUDENT.**
4. **ENROLLMENT** and **GRADE**
5. **SECTION** and **COURSE**
6. **SECTION** and **ASSISTANT**
7. **COURSE** and **MODULE**
8. **MODULE** and **LESSON**
9. **LESSON** and **TOPIC**
10. **TOPIC** and **WIDGET CONTENT**

These classes cannot exist without the other therefore the compositions are necessary.

ASSOCIATIONS:

11. **STUDENT** and **ENROLLMENT**
12. **ENROLLMENT** and **SECTION**
13. **SECTION** and **FACULTY**

9. CARDINALITY:

STUDENT can have many **ADRESS, EMAIL** and **NUMBER.**

One **STUDENT** has many **ENROLLMENT.**

One **ENROLLMENT** has many **GRADE.**

One **ENROLLMENT** has many **SECTION.**

One **SECTION** has many **ASSISTANT.**

Many **SECTION** has one **FACULTY**.

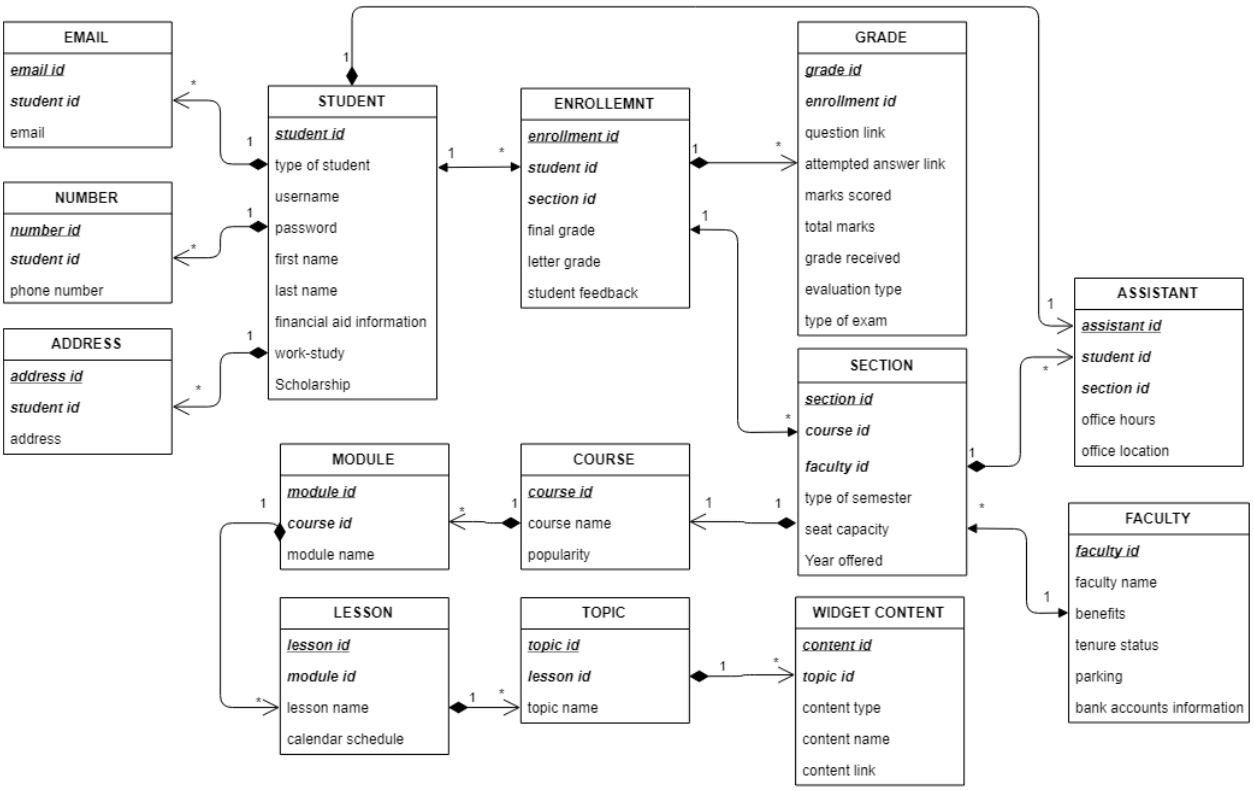
One **SECTION** has one **COURSE**.

One **COURSE** has many **MODULE**.

One **MODULE** has many **LESSON**.

One **LESSON** has many **TOPIC**.

One **TOPIC** has many **CONTENT WIDGETS**.



In **ENROLLMENT** ***student id*** and ***section id*** can form a compound key to identify each record uniquely.

ENROLLEMENT class can be an association class to **STUDENT** and **SECTION**

FINAL CLASSES:

STUDENT (***student id***, type of student, username, password, first name, last name, financial aid information, work-study, Scholarship).

EMAIL (***email id***, ***student id*****,** email)

NUMBER (***number id***, ***student id***, phone number).

ADDRESS (***address id***, ***student id***, address).

ENROLLMENT (***enrollment id***, **{*student id*, *section id*}**, final grade, letter grade, student feedback).

GRADE (***grade id***, ***enrollment id,*** question link, attempted answer link, marks scored, total marks, grade received, evaluation type, type of exam).

SECTION (***section id***, ***course id***, ***faculty id***, type of semester, seat capacity, year offered).

FACULTY (***faculty id***, faculty name, benefits, tenure status, parking, bank accounts information).

ASSISTANT (***assistant id***, ***student id***, ***section id***, office hours, office location).

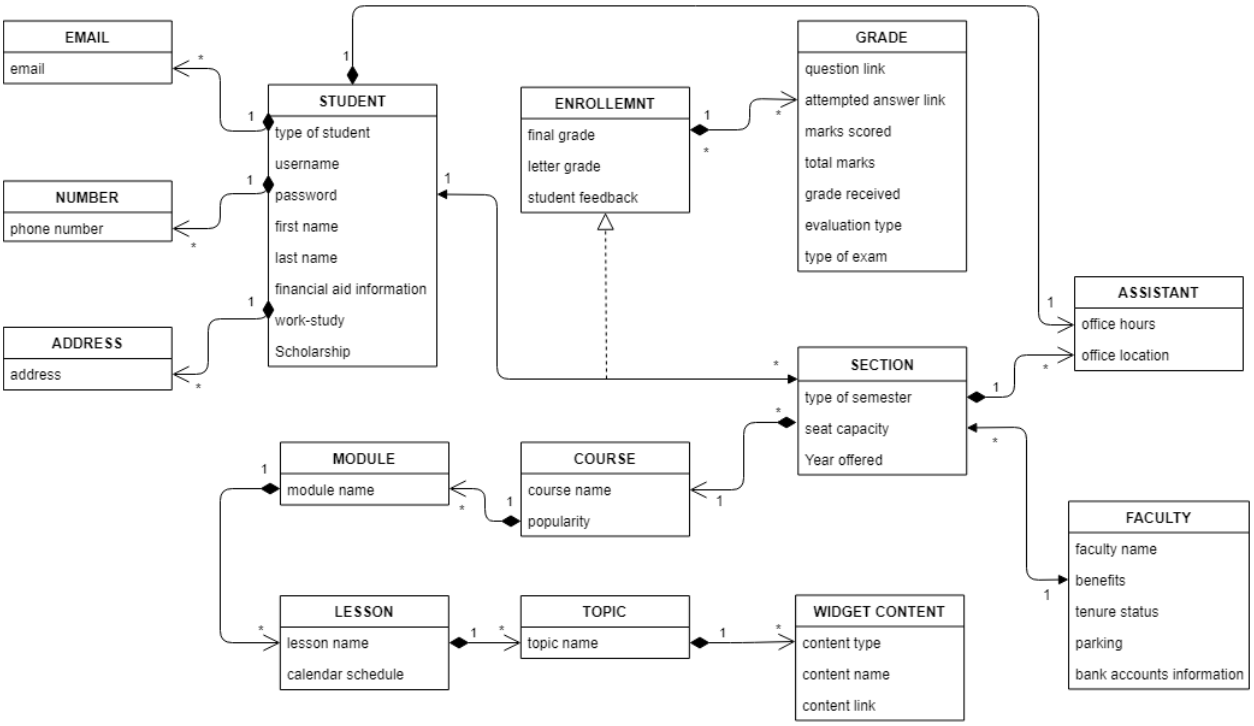
COURSE (***course id***, course name, popularity).

MODULE (***module id***, course id, module name).

LESSON (***lesson id***, ***module id***, lesson name, calendar schedule).

TOPIC (***topic id***, ***lesson id***, topic name)

WIDGET CONTENT (***content id***, ***topic id***, content Type, content name, content link).



FINAL UML DIAGRAM

FINAL DEFENCE:

The final UML in my opinion is a good solution to the given problem.

Faculty can have many sections which can have one course. Therefore, faculties can author courses. Modules and lessons have an attribute calendar schedule in LESSON which allows rearrangement. WIDGET CONTENT has an attribute called content type which can be like YouTube videos, slides, text documents, raw HTML, evaluations, and many more. Evaluation have been moved to GRADE, having another class for exam did

not make sense in evaluation to me. So, what I did was create an attribute called evaluation type which can be an exam and if it is an exam the type of exam can be an essay questions, multiple choice questions, fill in the blank questions, and many more types of questions. To apply popularity constraint, I have added an attribute called popularity in COURSE which is like an upvote system. The more upvotes a course has the more popular it is. Using the popularity many sections can be created for a course for a semester. SECTION has an attribute called type of semester which is of data type enumeration. There are 5 types of semesters: fall, spring, full summer, summer 1 and summer 2. Students can enroll in different sections for a course through ENROLLMENT. Therefore, ENROLLEMENT is an association class to STUDENT and SECTION. STUDENT and registrar's office can check their grades in ENROLLEMENT. STUDENT and FACULTY are master classes that hold information about their respective domains which need to be updated regularly. In GRADE there are all kinds of information about question, attempted answer to the question, marks received which student can check with the ASSISTANT during office hours. ASSISTANT is a new class which keeps track of office information. It also is a part SECTION which is also an associated with FACULTY so each faculty can have multiple sections which can have multiple assistants.