Khang Ngo
CMSC 461

CMSC 461 Project Phase A & B Report

The project is an implementation of a SQL database powered by Python and intractable by a user to perform a specified set of actions, while following a series of steps in order to ensure maximum familiarisation with the processes involved in the designing of a database.

Project Phases Analysis

Phase A of the project consists of an analysis of each phase of the project as well as analysis of the requirements of the database server to be created. By analysing the phases of the project in depth, one can learn more about and better understand the various steps of designing a database server.

Phase B of the project involves the creation of ER diagrams identifying relationships, entities, attributes and constraints that become the design of the final database's tables. This phase involves close analysis of the project requirements in order to create entities containing the correct required attributes and relationships.

Phase C of the project is the mapping of Phase B's ER model to actual tables that will be used in the database server. Normalisation skills will be applied in this phase to produces the most efficient tables possible whilst retaining satisfaction of project requirements.

Phase D is the writing of SQL scripts to create physical SQL tables based on designs made in earlier phases. This phase will test SQL writing skills and understanding of the ER model and tables designed in previous phases. The requirements must be well understood in order to write efficient scripts retrieving the correct required information from the tables.

Phase E is the development of the Python interface which will interact with the SQL database and perform required actions, as well as indexing of the database. This phase will require skills on how to properly use the SQL database now that it has been created.

Requirements Analysis

The database to be implemented is an academics-assisting database designed to process graduate applications to a college. Applications contain a variety of information and are ultimately evaluated by a professor and given a final decision. The requirements specify a list of queries the database should be able to make, and most of these queries seem to be statistical data retrievals of the applications in the database.

Data Requirements Analysis
What required attributes I determined each required entity should have, based on the project descriptions.

Applicant
- student ID
- contact information
- birthday
- name
- gender

Application
- degree program
- GRE information
- essay
- requirements answers
- semester
- year
- reference emails
- education

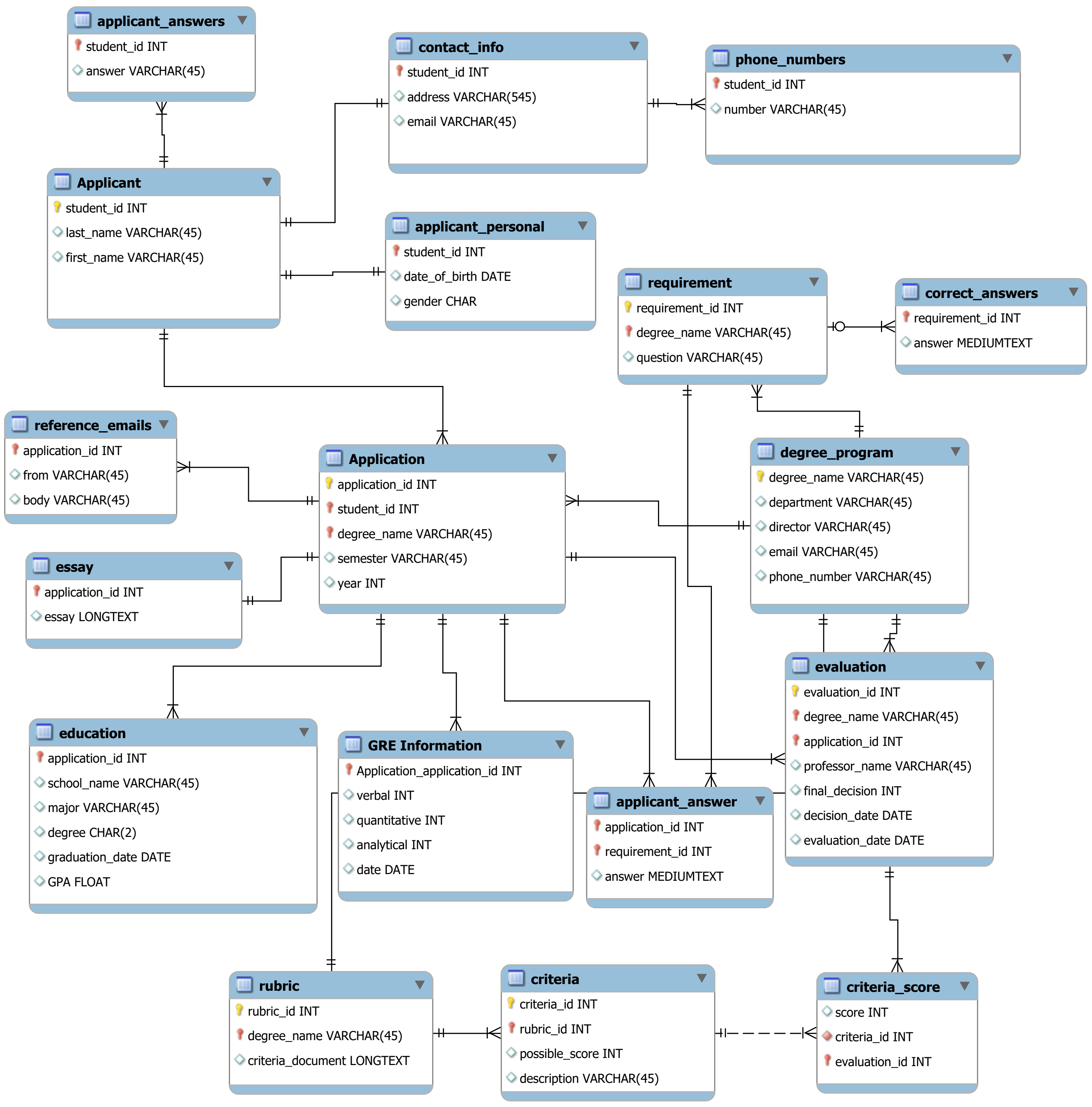
Degree Program
- name
- director
- email
- department
- phone

Admission Requirement
- question
- possible answers

Admission Rubric
- evaluation criteria
- criteria possible score and description

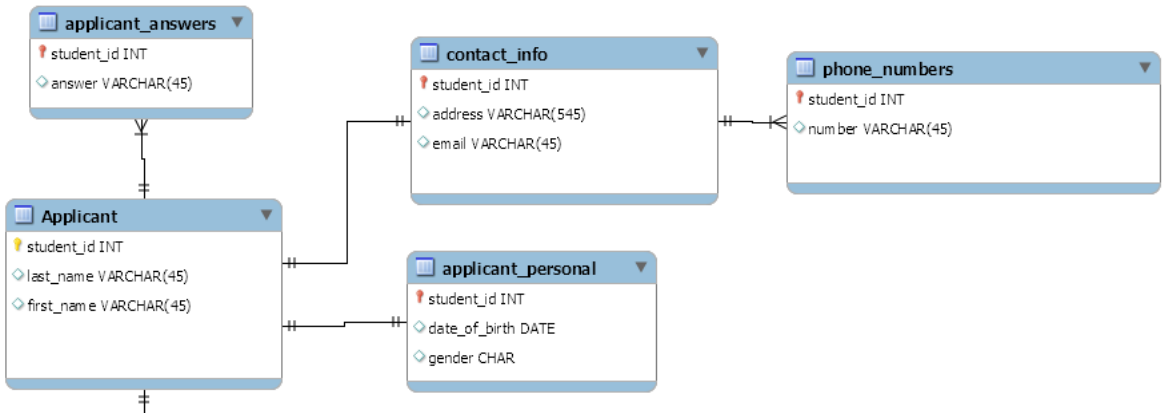
Admission Evaluation
- professor [making evaluation]
- decision
- degree program
- criteria scores
- date/time for evaluation decision and evaluation creation

Phase B Diagram



Design Decisions and Assumptions

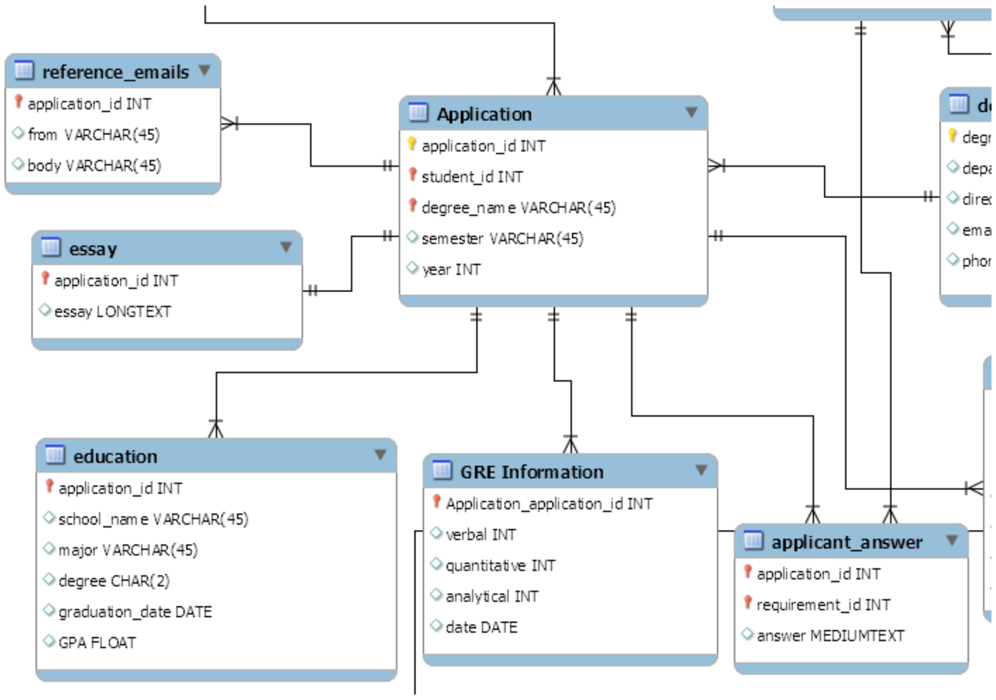
Applicant



Assumptions: An applicant can make multiple applications. An applicant can only have one email and address, but multiple phone numbers.

I decided to split out certain parts of the applicant entity, such as the date of birth, gender, address, and email, as they did not seem to be as important based on the required queries. I decided to put phone numbers and applicant answers in their own tables as they may number an unknown amount. They are both findable via the student ID of the applicant entity.

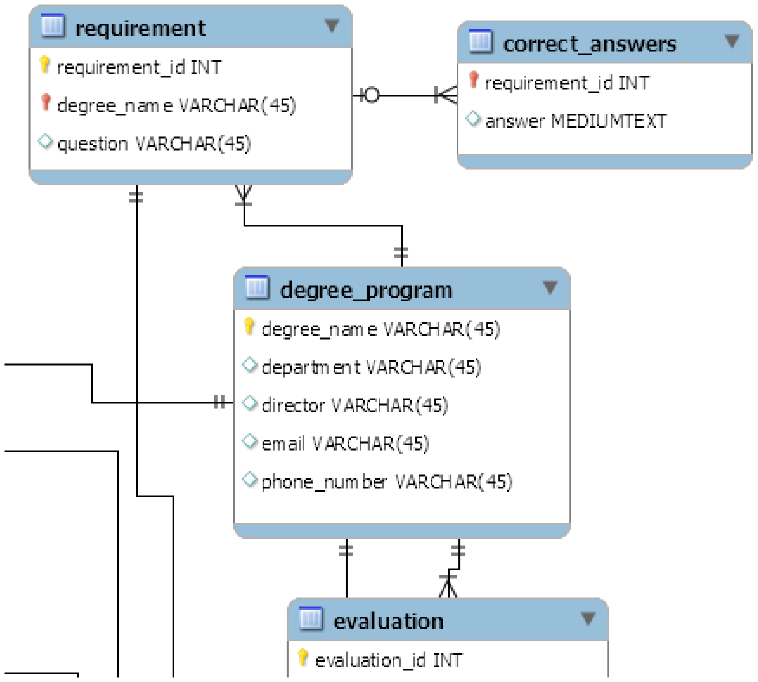
Application



Assumptions: An applicant can make multiple applications, an application can include multiple GRE test results, an application can include multiple education entries, and application can only include 1 essay.

The application entity ended up being pretty complex and I intend on having it be one of the more core parts of the database. A lot of the required queries will end up using this as a dependency to access other data, and as such I wanted to keep it as light as possible. I will probably end up moving semester and year out as well, if necessary. I decided to keep essay and reference emails in their own table as they may end up having potentially large bodies, and are not accessed as much. GRE Information and education made sense as their own entity, and, as such, I made them so. I assumed that it would be possible for an application to include more than one education and GRE entry, as an applicant may have gone to multiple schools are taken multiple tests. I decided not to attach this information directly to the applicant, as the applicant may wish to attach specific education entries to certain applications, but not others. Also, the requirements state that the application itself should include the education and GRE information, and so that furthered my decision. Since the requirements said an application would have "a" essay, I decided that an application would be limited to one essay.

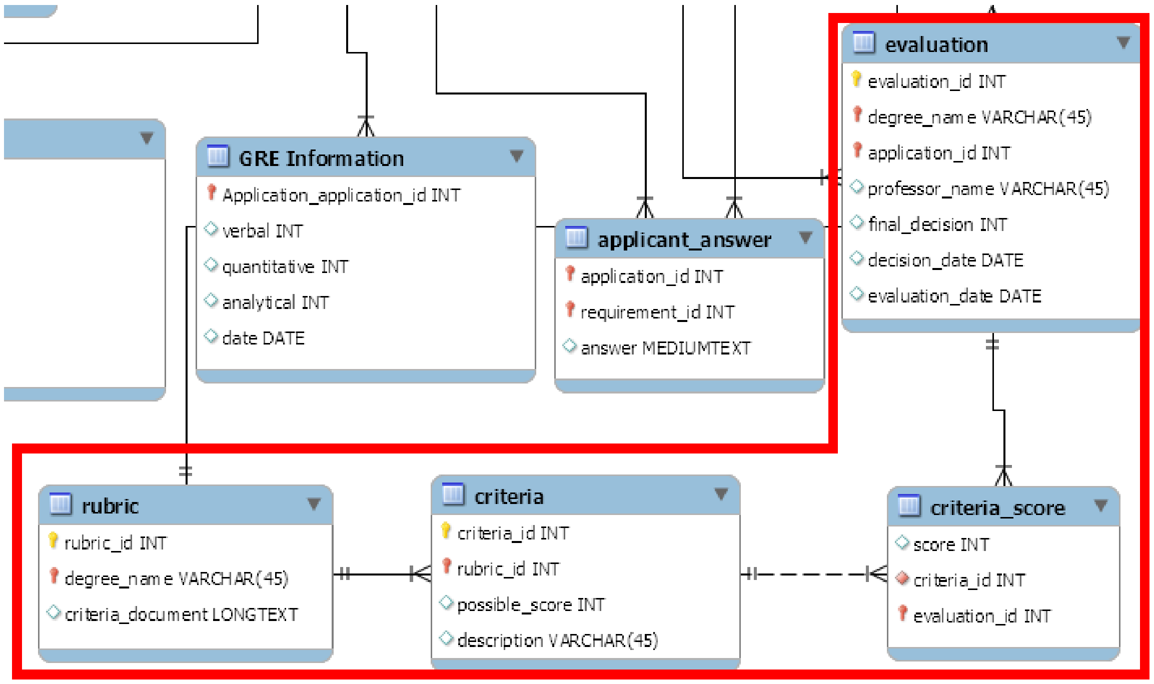
Degree Program and Admissions Requirements



Assumptions: A degree program can only have one department, director, email, and phone number, unlike an applicant, which could have multiple phone numbers. The degree_name is unique (as stated by the requirements) and as such should be usable as a primary key.

I decided to not split up the attributes of degree program much, as it seems the most important piece of data in the degree program entity is its primary key, the degree_name, which will be present as a foreign key in any other elements that may need to reference this entity. Requirements is its own entity as it appears to be a required entity, and as such has its own requirement ID. Although not visible in the picture above, the answers to requirements for any given applicant will reference the requirement object by ID, ensuring a connection without having to search through degree program.

Admission Rubric



Assumptions: Each degree program can only have one rubric, rubric criteria scores are simple integers, there can only be one professor per evaluation, but there can be multiple evaluations per applicant (as stated by the requirements).

It is not exactly clear what the requirements define a "rubric" as. Although the requirements do list a few things that can make up criteria, "(undergraduate education, prerequisite knowledge, letters, ect.)", I thought to would make more sense to have the rubric be just a document professors can read off while evaluating an applicant, instead of manually making columns for each of the listed things. I also thought that each degree should be able to have different criteria, so criteria has been split into its own entity. I decided to give criteria an ID despite it not being a required entity, as that allows criteria scores entity of an evaluation entity to easily connect with the criteria entity the score is of. For the evaluation entity, I decided to not split up much of the attributes, as for now it does not seem like the entity will be used greatly.