EduSpark: Your best supporter to study abroad

Liangyue Zhou, Hansheng Zhang, Hao Luo
Yuheng Liu, Xiaoyi Zheng, Yujie Peng, Ziyue Lin

May 9, 2023

1 Background and Motivation

Motivated by our own need to seek excellent graduate programs overseas, as well as the upsetting fact that notably limited information is available for appliers from Mainland China, we are interested in building a database system for providing instrumental recommendations to take use of the historical application data. This idea is envisioned to combine our knowledge concerning database system designing and our personal experiences, which is anticipated to give birth to a practically weighty and commercially successful product.

2 Introduction

Our organization is concentrated on providing educational information for Mainland China undergraduate students who wish to study abroad for graduate learning. Therefore, we designed EduSpark which is a graduate programs application database. The database should store information about universities, their programs, standardized tests, employment outcomes, and applicant details. Based on our product, students can get the desirable information and further data analysis provided by us. We are looking forward to offering application support and suggestions to students. Hence, the database should be able to manage application details, including the application status and necessary documents. We believe that with our help, there's no obstacles in graduate program application!

3 Design and Implementation

3.1 EER Diagram

The following figure demonstrates the entities and relationships of our database design. Further details will be introduced in the relational schema part. Since Mysql workbench can only convert the database to a EER diagram, we use EER instead of ER.

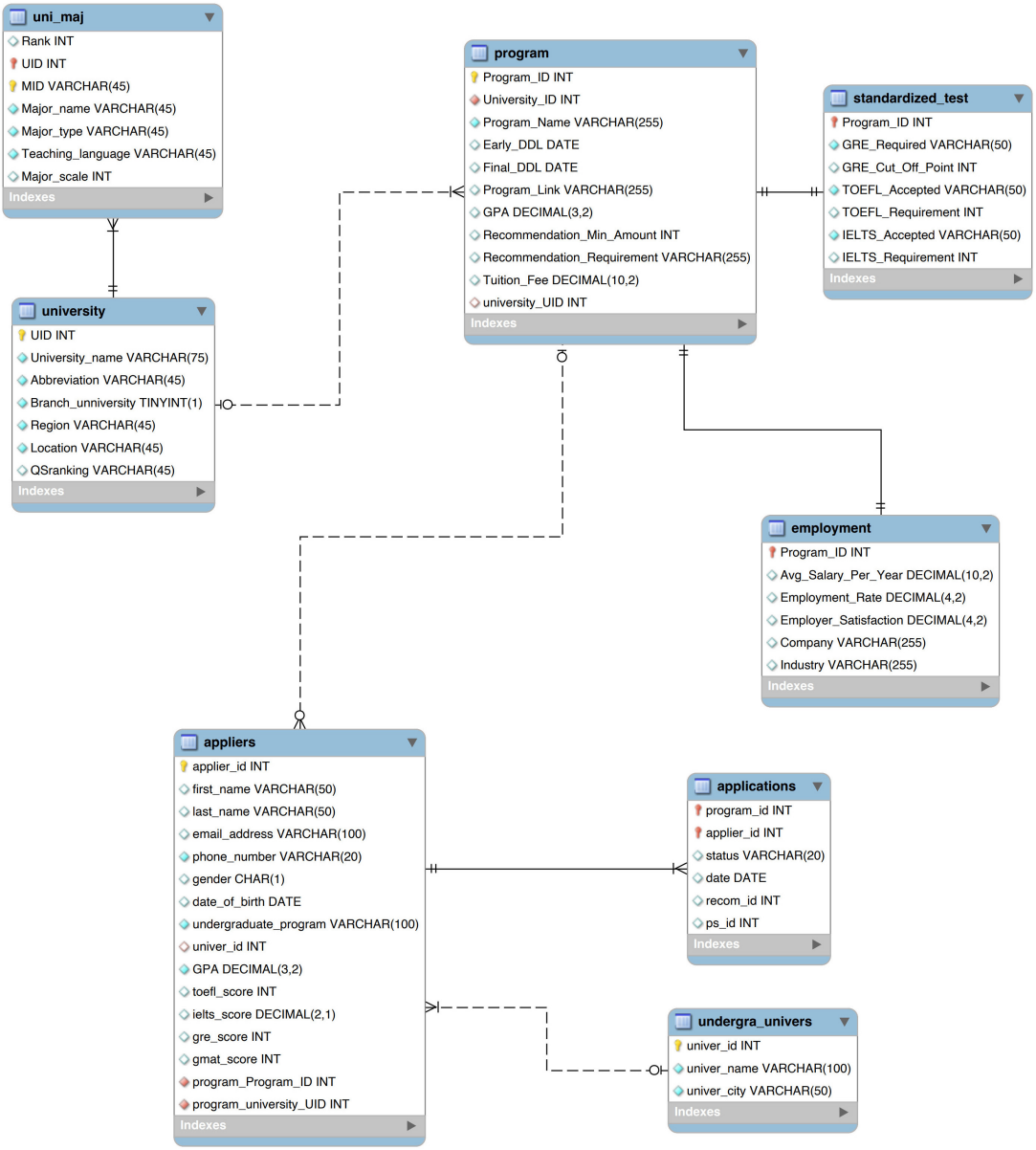

Figure 1: EER Diagram

Legend:

1. Dashed lines connect strong entities; solid lines connect strong and weak entities.
2. || indicates mandatory (i.e., must have at least one relationship); o indicates non-mandatory (i.e., can have 0 relationships).
3. $- - --$ represents 1:1; $- - - <$ represents 1:n.

3.2 Relational Schema

3.2.1 Assumptions

The relational schema of our database system are based on following assumptions:

• Each applier can apply different programs in one university, but he or she can only apply a specific program one time in a year.

• One applier can only have one applier_id, which will not change in different applications.

• One applier have only one undergraduate university.

3.2.2 Attributes explanation

• University:

- UID: Unique identifier for each university.
- University_name: The name of the university.
- Abbreviation: The abbreviation commonly used for the university.
- Branch_unniversity: A boolean flag indicating whether the university is a branch campus (True) or not (False).
- Region: The region in which the university is located.
- Location: The specific location or address of the university.
- QSranking: The university's QS World University Ranking, if available.

• Uni_maj (University Majors):

- Rank: The rank of the major within the university.
- UID: The unique identifier of the university offering the major (foreign key referencing the University table).
- MID: Unique identifier for the major within the university.
- Major_name: The name of the major.
- Major_type: The type of major, such as arts or sciences.
- Teaching_language: The language in which the major's courses are taught.
- Major_scale: The number of students enrolled in the major.

• Program:

- Program_ID: Unique identifier for the program.
- University_ID: Unique identifier of the university offering the program (foreign key referencing the University table).
- Program_Name: The name of the program.
- Early_DDL: The early deadline for applying to the program.

- Final_DDL: The final deadline for applying to the program.
- Program_Link: A URL link to the program's official webpage.
- GPA: The minimum GPA required for admission to the program.
- Recommendation_Min_Amount: The minimum number of recommendation letters required for the program.
- Recommendation_Requirement: Additional requirements or guidelines for recommendation letters.
- Tuition_Fee: The tuition fee for the program.

• Standardized_Test:
- Program_ID: Unique identifier for the program (foreign key referencing the Program table).
- GRE_Required: Indicates whether the GRE test is required for the program.
- GRE_Cut_Off_Point: The minimum GRE score required for the program, if applicable.
- TOEFL_Accepted: Indicates whether the TOEFL test is accepted for the program.
- TOEFL_Requirement: The minimum TOEFL score required for the program, if applicable.
- IELTS_Accepted: Indicates whether the IELTS test is accepted for the program.
- IELTS_Requirement: The minimum IELTS score required for the program, if applicable.

• Employment:
- Program_ID: Unique identifier for the program (foreign key referencing the Program table).
- Avg_Salary_Per_Year: The average yearly salary for graduates of the program.
- Employment_Rate: The percentage of graduates who find employment within a specified period after graduation.
- Employer_Satisfaction: A measure of employer satisfaction with graduates from the program.
- Company: Representative companies that hire graduates from the program.
- Industry: The industry in which graduates from the program typically find employment.

• Undergra_univers (Undergraduate Universities):
- univer_id: Unique identifier for the undergraduate university.
- univer_name: The name of the undergraduate university.
- univer_city: The city in which the undergraduate university is located.

• Appliers:
- applier_id: Unique identifier for each applicant.
- first_name: The first name of the applicant.
- last_name: The last name of the applicant.
- email_address: The email address of the applicant.

- phone_number: The phone number of the applicant.
- gender: The gender of the applicant (M, F, or other).
- date_of_birth: The date of birth of the applicant.
- undergraduate_program: The undergraduate program the applicant completed.
- univer_id: The unique identifier of the undergraduate university the applicant attended (foreign key referencing the Undergra_univers table).
- GPA: The Grade Point Average of the applicant.
- toefl_score: The TOEFL score of the applicant, if applicable.
- ielts_score: The IELTS score of the applicant, if applicable.
- gre_score: The GRE score of the applicant, if applicable.
- gmat_score: The GMAT score of the applicant, if applicable.

• Applications:
- program_id: The unique identifier of the program to which the applicant is applying (foreign key referencing the Program table).
- applier_id: The unique identifier of the applicant (foreign key referencing the Appliers table).
- status: The current status of the application (e.g., submitted, accepted, rejected, waitlisted).
- date: The date the application was submitted.
- recom_id: The identifier for the recommendation letter(s) submitted by the applicant.
- ps_id: The identifier for the personal statement submitted by the applicant.

3.2.3 Relationships

• University (1) −− (N) Uni_maj
• University (1) −− (N) Program
• Program (1) −− (1) Standardized_Test
• Program (1) −− (1) Employment
• Undergra_univers (1) −− (N) Appliers

3.2.4 Relational Schema

The relational schema of our database system is exhibited below. The primary key of each table is underlined.

• `university(UID, University_name, Abbreviation, Branch_university, Region, Location, QSranking)`

FD: The primary key is UID, and all other attributes are dependent on it.
No MVD.

• uni_maj(Rank, UID, MID, Major_name, Major_type, Teaching_language, Major_scale)
Foreign key: UID, which references the primary key "UID " in table "university"
FD: The primary key is UID and MID, and all other attributes are dependent on it.
No MVD.

• program(Program_ID, University_ID, Program_Name, Early_DDL, Final_DDL, Program_Link, GPA, Recommendation_Min_Amount, Recommendation_Requirement, Tuition_Fee)
FD: The primary key is Program_ID, and all other attributes are dependent on it.
No MVD.

• standardized_Test(Program_ID, GRE_Required, GRE_Cut_Off_Point, TOEFL_Accepted, TOEFL_Requirement, IELTS_Accepted, IELTS_Requirement)
FD: The primary key is Program_ID, and all other attributes are dependent on it.
No MVD.

• employment(Program_ID, Avg_Salary_Per_Year, Employment_Rate, Employer_Satisfaction, Company, Industry)
FD: The primary key is Program_ID, and all other attributes are dependent on it.
No MVD.

• undergra_univers(univer_id, univer_name, univer_city)
FD: The primary key is univer_ID, and all other attributes are dependent on it.
No MVD.

• appliers(applier_id, first_name, last_name, email_address, phone_number, gender, date_of_birth, undergraduate_program, univer_id, GPA, toefl_score, ielts_score, gre_score, gmat_score)
FD: The primary key is applier_ID, and all other attributes are dependent on it.
No MVD.

• applications(program_id, applier_id, status, date, recom_id, ps_id) FD: The primary key is applier_ID and program_id, and all other attributes are dependent on it.
No MVD.

Normal form: according to the above functional dependencies, the design is in 3NF, as there are no partial or transitive dependencies.

3.2.5 Functional Dependency Analysis

Each table is in third functional dependency. Moreover, our system does not involve multi-valued dependencies.

3.3 Indexing

Indexing and hashing are a couple of standard approaches which will speed up queries filtering or other operations. A number of ways to combine the idea of indexing with our database system designing is illustrated below.

University(UID): As the primary key, indexing the UID will speed up queries that filter or join based on the university ID. Additionally, it will be helpful for optimizing queries that involve grouping or aggregations based on university information.

Program(Program_ID): As the primary key, indexing the Program_ID will improve performance for queries that analyze data across multiple programs, such as comparing application rates or admission requirements.

Appliers(applier_id): As the primary key, indexing the applier_id will speed up queries that filter or join based on the applicant ID. This index will also be beneficial for queries analyzing applicant demographics, educational backgrounds, or standardized test scores.

Applications(program_id, applier_id): As primary key attributes, indexing both program_id and applier_id will enhance the performance of queries that involve filtering or joining based on program_id and/or applier_id.

3.4 Sample Data (see the excel files in appendices)

3.5 Sample queries

Here we only cover some samples. Others see the appendices.

```
-- 1. Region of universities
SELECT university.region, COUNT(*) AS num
```

```
FROM university, program, applications
WHERE university.UID = program.University_ID AND program.
   Program_ID = applications.program_id AND applications.status =
    'Accepted' AND applications.applier_id IN (
    SELECT applier_id
    FROM appliers
    WHERE GPA < 3.3
)
GROUP BY Region
-- 2. Get the average GPA, GRE score and TOEFL score of appliers
   who are accepted by CSE@MIT in 2022
SELECT AVG(appliers.GPA), AVG(appliers.GRE_score), AVG(appliers.
   TOEFL_score)
FROM appliers
JOIN applications
ON appliers.applier_id = applications.applier_id
JOIN Program
ON applications.program_id = Program.Program_ID
JOIN undergra_univers
ON appliers.univer_id = undergra_univers.univer_id
WHERE Program_Name = 'CSE@MIT' AND status = 'Accepted' AND
   univer_name = 'Chinese University of Hong Kong, Shenzhen' AND
   YEAR(applications.date) = 2022;
```

3.6 Data Mining

We utilize decision tree model to realize data mining in support of providing generalized positioning for our customer students. Decision tree model is a traditional and classical data mining approach which is perfectly suitable for our context of providing generalized positioning for our customer students. It is capable of reading the data with a number of features at one time, integrating and analyzing them, and extracting the hidden logic which may be challenging to uncover. Python code are shown in the appendices. Part of a sample trained model is shown below.

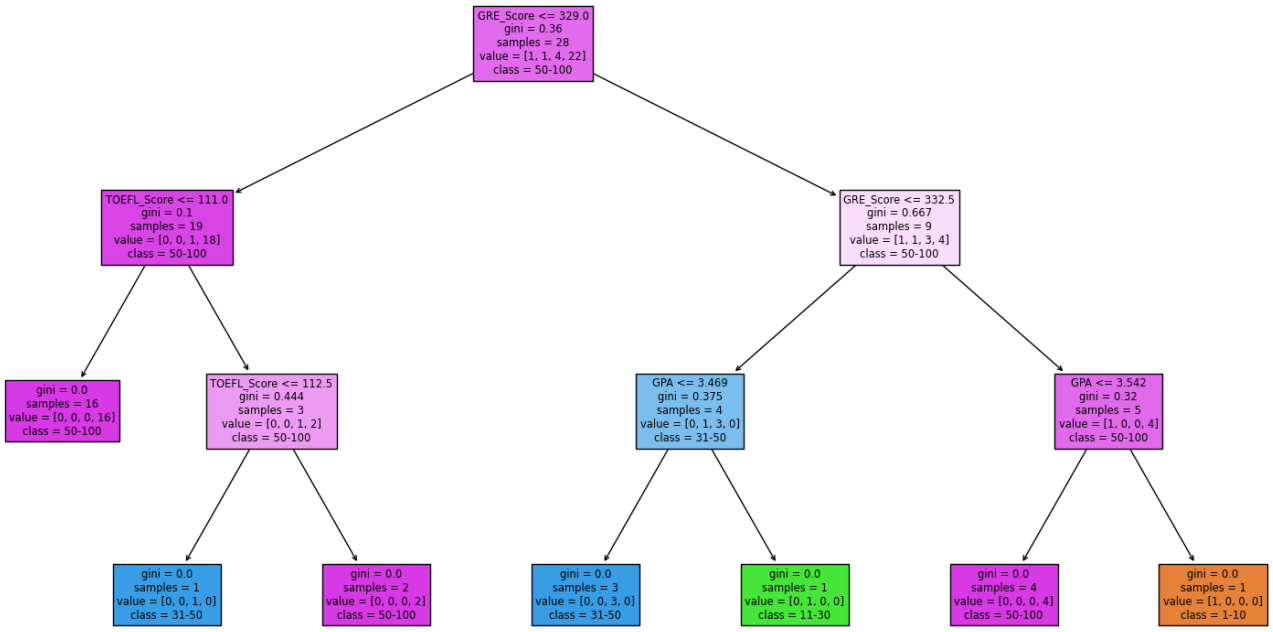

Figure 2: Decision Tree

4 Conclusion and Self-Evaluation

With the data stored in our database system with choreographed relational schema and our data mining approach, we believe that the designed system involves notable educational and commercial significance where we are able to provide our customer students with meaningful services. To finish the project, all group members have contributed greatly and equally.

From this project, we learned about designing a well-structured relational database for an educational organization, including identifying entities, relationships, and constraints. We encountered several problems, including how to determine normalization levels, how to set foreign key constraints and how to create ER diagram. Limitations include the potential for outdated or incomplete information, and the need to maintain referential integrity.

5 References

- Data from: https://opencs.app

6 Appendices

1. Sample data: see excel files in folder "Data". We use python to insert those .xlsx files to the database. Insertion of data: inser data.py
2. SQL codes: creation of the database is in create.sql. Queries are in query.sql

3. PPT: EduSpark.pptx
4. Decision tree: decision tree.py