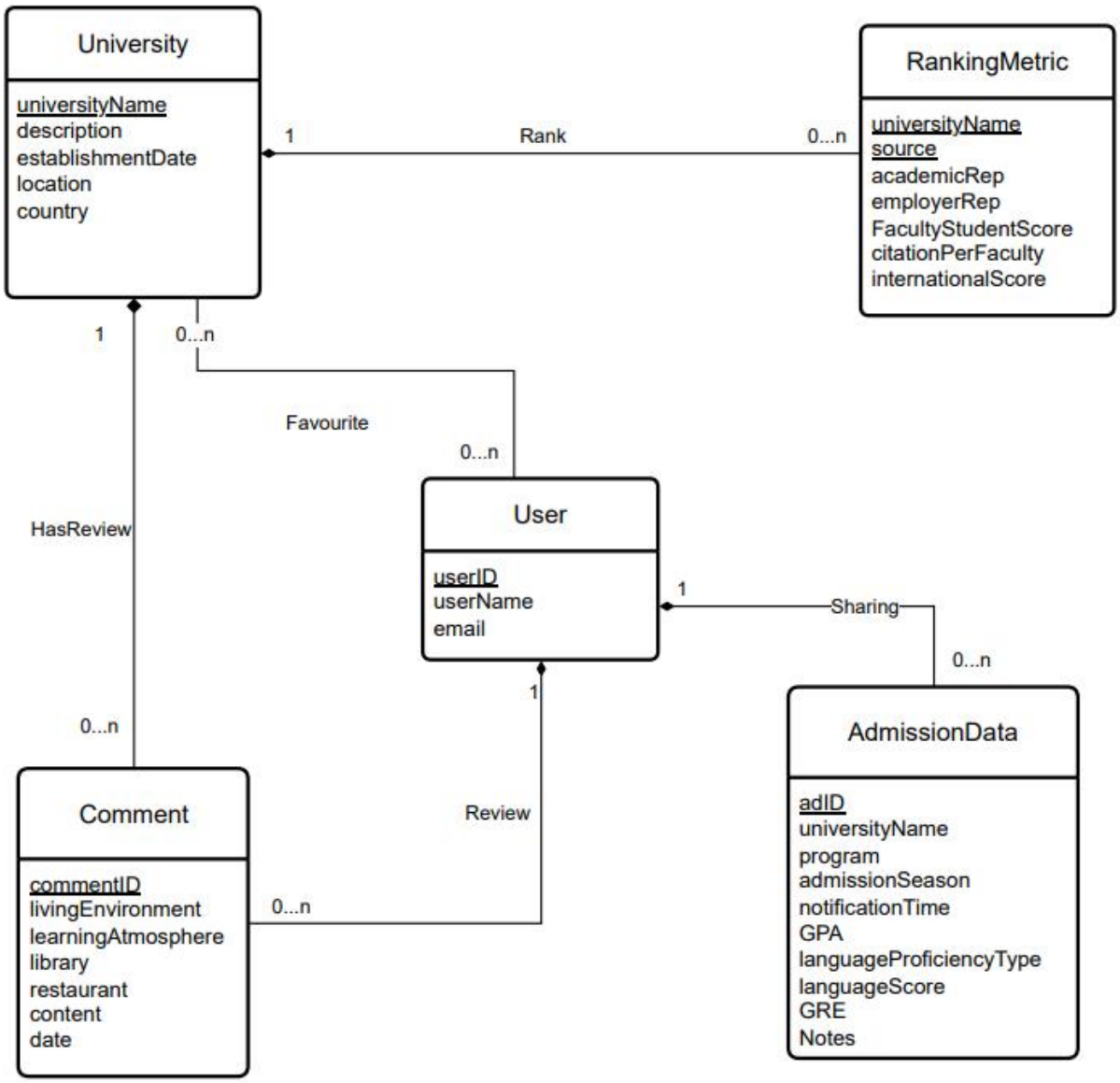
UML graph:



1. Assumptions

Entities:

User: This is an entity storing necessary user information. Because the information might be used widely, so we extract according information and include them as an entity.

University: University is an entity for basic university information. These information will be related to comments and favorites. So although there are some similarities between university and ranking information, we separate the two parts. In this case the university basic information can contribute to the review and favorite part better.

RankingMetric: This is an entity includes a lot of necessary metric data for ranking. Since the metric data are based on real datasets, so there could be multiple metric data coming from different lists for a particular university. For less redundancy, we extract these data into this entity rather than treating them as the attributes of University.

Comment: We encourage users to make comments on their university, so that potential applicants can get to know the detail of a university better. The content of this entity is independent so we make it as an entity.

AdmissionData: We encourage people to share their application data to help others get to know a program. This is an independent part so there is no place to store according information as attributes. So we make it as an entity.

Relationships:

Rank: This is a many to one relationship. Because the ranking system will be based on basic information of a university, so there must be one university for metrics. Because there could be multiple lists for metrics and some universities may not be included in particular lists, so for a university there could be zero to many rankingMetrics. We will use the attribute source to tell the origin of a particular metric record.

Favorite: This is a relationship for users to collect universities. Every user can collect as many universities as they want, and a universities can be collected by different users. So this is a many to many relationship.

HasReview: A university can have many comments. But a comment will definitely belong to a university, so this is a many to one relationship.

Review: This relationship represents that a user's review history. A user can have many comments, and every comment must belong to a particular user. So this is a many to one relationship.

Sharing: This relationship is used to represent the post action of the admission data sharing. A user can share many admission data to give other people insights. And every admission data will belong to the according user. So this is a many to one relationship.

2. Relational Schema

According to the UML graph, we translate the UML model into following relational schemas.

University(universityName:VARCHAR(255)[PK],description:VARCHAR(255),establishmentDate:
VARCHAR(255), location: VARCHAR(255), country: VARCHAR(255))
The FD for this schema is: universityName -> description, establishmentDate, location, country. Since the universityName is the superkey, this schema adheres to BCNF.

RankingMetric(universityName:VARCHAR(255)[PK][FK to University.universityName], source[PK]:VARCHAR(100), academicRep:DOUBLE, employerRep:DOUBLE, facultyStudentScore:DOUBLE, citationPerFaculty:DOUBLE, internationalScore:DOUBLE)
The FD for this schema is universityName, source ->academicRep, employerRep, facultyStudentScore,

citationPerFaculty, internationalScore. Since the left side of the FD is the superkey, this schema adheres to BCNF. The schema includes the foreign key for University schema, so it translates the relationship Rank naturally.

User(userID:INT[PK], userName:VARCHAR(255), email:VARCHAR(255))
The FD for this schema is userID -> userName, email; email -> userID, userName. Since the left sides of the FD are all superkey, this schema adheres to the BCNF.

Favourite(universityName:VARCHAR(255)[PK][FK to University.universityName], userID:INT[PK][FK to User.userID])
This is a relationship schema for University and User. Since there are only two columns, this table adheres to the BCNF naturally.

Comment(commentID:INT[PK], universityName:VARCHAR(255)[FK to University.universityName], userID:INT[FK to User.userID], livingEnvironment:INT, learningAtmosphere:INT, library:INT, restaurant:INT, content:VARCHAR(255), date:VARCHAR(255))
This schema includes both the information for the comment and two foreign keys. The two foreign keys are used to translate the relationship between schema Comment and University and User. The FD of this schema is commentID -> universityName, userID, livingEnvironment, learningAtmosphere, library, restaurant, content, date. The commentID is the key of this entity, so it naturally can be used to determine according comment information. And because there must be a university and user matched to a comment, a commentID can determine a userID and an according universityName. So, the above FD works. And the left side of the FD is a superkey, so this schema adheres to BCNF.

AdmissionData(adID:INT[PK], userID:INT[FK to User.userID], universityName:VARCHAR(255)[FK to University.universityName], program:VARCHAR(255), admissionSeason:VARCHAR(255), notificationTime:VARCHAR(255), GPA:DOUBLE, languageProficiencyType:VARCHAR(255), languageScore:DOUBLE, GRE:DOUBLE, Notes:VARCHAR(255))
This schema includes basic information for an admission data and a foreign key for university. The funcation dependency is adID -> userID, universityName, program, admissionSeason, notificationTime, GPA, languageProficiencyType, languageScore, GRE, Notes. The adID is the key of the entity, so it can determine the basic info for the admission data naturally. Because one admission data must belong to one user, so the adID can also tell the userID, thus the above FD works. The left side of the FD is the superkey, so this schema adheres to BCNF.