CS 432/532 Homework 2
ER to Relation Transform

Name: Jigeesha Sanjeev Jain
B number: B00928112

1. [70%] Transform the provided ER diagram for the Student Registration System to relations using the techniques discussed in class. For composite attributes, use Method 1 (i.e., use the more specific attributes only) to perform the transformation. For each relation obtained, do the following:
   - underscore the key
   - specify other candidate keys, if any, and foreign keys, if any
   - specify the constraints associated with this relation, including all the constraints that are described in the Requirements Document.

Answer:

1.Students (sid, firstname, lastname, status, gpa, email)
The other candidate key is email.

Foreign key: there are no foreign keys in students

Constraints: specific values for status whether student is freshman, sophomore, junior, senior.
GPA is a decimal number between 0 and 4.

2.Courses (cid, dept_code, course#, title, credits, deptname)
The are no candidate keys here

Foreign keys: Courses.deptname is the foreign referencing the Departments.deptname

Constraints: Here the course number must be 3 digit. It should be in between 100 and 499 for undergraduate courses and in between 500 and 799 for graduate courses.
Also, the credits i.e. the credit hours must be 3 for graduate and 4 for undergraduate.

3.Classes (cid, sec#, year, semester, start_time, end_time, limit, size, capacity, classroom, fid)
There are no candidate keys

Foreign keys: Classes.fid is the foreign key referencing the Faculty.fid

Constraints:
The values of semester are limited to {Spring, Fall, Summer 1, Summer 2}.

The actual size of a class must not exceed the limit of the class.
The limit of a class must not exceed the capacity of the assigned classroom.
No classes of overlapping times can be assigned to the same classroom

4.Classes_days (cid, sec#, year, semester, days) :
There are no candidate keys

Foreign keys: Classes_days.cid, Classes_days.sec#, Classes_days.year, Classes_days.semester are the foreign key referencing to Classes.cid,

Constraints:
The values of days are limited to {Monday, Tuesday, Wednesday, Thursday, Friday}

5.Faculty (fid, firstname, lastname, rank, office, email)

Candidate keys: offices, email

There are no foreign keys

Constraints:
Valid values for faculty rank are {lecturer, assistant professor, associate professor, professor}.
No faculty member can teach classes with overlapping times.

6.Departments (deptname, office, phone#)
Candidate keys: office
There are no foreign keys
There are no constraints

7.Enroll_in (sid, cid, sec#, year, semester, lgrade, ngrade)
Candidate keys: No such candidate keys

Foreign keys: Enroll_in.sid is the foreign key referencing Students.sid
Enroll_in.cid is the foreign key referencing Courses.cid
Enroll_in.sec#,year,semester are foreign keys referencing Classes.sec#, Classes.year, Classes.semester

Constraints:
Only {A, B, C, D, F, I, null} are valid letter grades
Corresponding number grades are: A ←→ 4, B ←→ 3, C ←→ 2, D ←→ 1, F ←→ 0 and I ←→ null.
No student is allowed to enroll in different classes of the same course more than once.

The student has not been registered in a different section of the same course
The student has completed all the prerequisite courses with a grade of at least C.

8.Major_in (sid,deptname)
Candidate key: No candidate keys
Foreign Key: Major_in.sid is a foreign key referencing Student.sid
Constraints: Each student is associated with either one or two departments, i.e., the majoring departments.

9.belong_to(deptname,fid)
Candidate key: No such candidate key

Foreign keys: belong_to.deptname is the foreign key referencing Departments.deptname
belong_to.fid is the foreign key referencing Faculty.fid

Constraints: Each faculty member belongs to one to three departments, and a department may have one or more faculty members.

10.prerequisite_of (cid, prerequisite_of_cid)

Candidate key: No such candidate key

Foreign keys: Prerequisite_of.cid is the foreign key referencing Courses.cid

Constraints: The student has completed all the prerequisite courses with a grade of at least C.
Courses and their prerequisite courses do not form cycles.

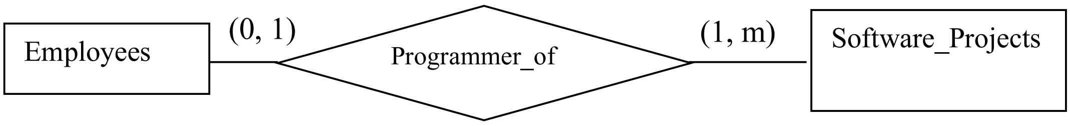
2. [10%] Let A and B be the only attributes of a relation R. Assume that neither A nor B is a key of R. Given that, answer the questions below.
   (a) Does the combination of these two attributes, (A, B), form a key of R? Why or why not?
   (b) Suppose the combination of these two attributes, (A, B), is a key of R. Can either A or B be a superkey of R? Why or why not?

Answer:

a. Initially, in this scenario it is mentioned that A and B are the attributes of relation R but neither of them is a key of R. A key is the one which is uniquely identified and has which follows minimality principle. It means that a key should be able to identify every row in that relation uniquely and not only that even in future for every potential tuple of the relation. Also, if the relation can be uniquely identified by just A or B then together A and B cannot form keys as it would result into redundancy and further increase inconsistency. As neither of these two are keys of R we will have to check properly with every scenario that the combination of (A, B) has unique values and if yes then they would form a key.

b. In this scenario, the two attributes (A,B) form a key of R which mean that both A and B together can uniquely identify the relation R and satisfy the minimality criteria. A superkey has only one requirement that it should be unique but as the combination of A,B is the key and not individual attribute A and attribute B are keys so there is no such mention that attributes A or attribute B is unique hence either A or B **cannot** be superkeys.

3. [20%]



Clearly and briefly describe two methods to convert the ER diagram above to relations introducing no null values. Don’t create any foreign key in Software Projects because it will introduce a lot of redundancies. If necessary, you can modify the ER diagram as long as the resulting ER diagram is logically equivalent to the given one.

Answer:

a) Method 1 [10%]:

Here in Employees entity, the maximum cardinality is 1 hence the foreign key will be created in Employees as it will not introduce any null values. The foreign key referencing the Software_Projects will be added to Employees. If the foreign key is added to Software_Projects it will unnecessarily introduce lot of redundancies.

b) Method 2 [10%]:

In the method 2, we would transform the relationship to a separate table which would contain two foreign keys that will be referencing both the entities this will avoid creation of any null values. Also, the values will be such that only when Employee is a programmer then only those values will be present for software projects and eventually null values will be avoided.