Name :Ankita Patra
Bnumber : B01101280
Mail: apatra@binghamton.edu

CS 432/532 Homework 2
ER to Relation Transform

Max total: 4 points

Before you begin, for academic honesty, please read the paragraph below and sign there.
I have done this assignment completely on my own. I have neither copied nor shared my solution with anyone else. I acknowledge that if I engage in plagiarism or cheating, I will sign an official form admitting to the violation, which will be added to my official university record. I also understand that a first offense will result in a grade of 0 for the assignment and a one-level reduction in my course letter grade, and any subsequent offense of any kind will result in a grade of 'F' for the course.
Name:**Ankita Patra**

_____

Signature:**Ankita Patra** _____

1. [3 points] 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.

Ans:

1. **Students** → _sid_, firstname, lastname, status, gpa, email, deptname
- **Primary Key**: _sid_
- **Foreign Key:** deptname → Departments(deptname)
- **Constraints**:
  - gpa is a decimal (0-4)
  - status must be one of →{ freshman, sophomore, junior, senior, graduate }
  - email is unique for every student

2. **Courses** → _cid_, dept_code, course#, title, credits
- **Primary Key**: _cid_
- **Foreign Key:** dept_code → Departments(dept_code)
- **Constraints**:
  - course# must be → 100-499 (undergrad) or 500-799 (graduate)
  - credits = 4 for undergrad, credits = 3 for graduate
  - A course must have a department

3. **Departments** → _deptname_, phone#, office
- **Primary Key**: _deptname_
- **Constraints**:
  - - A department must have at least one faculty
  - office is unique
  - phone # might have a format constraint.

4. **Faculty** → _fid_, firstname, lastname , office, rank, email, deptname
- **Primary Key**: _fid_
- **Foreign Key**: deptname → Departments(deptname)
- **Constraints**:
  - rank must be one of → lecturer, assistant professor, associate professor, professor
  - email is unique
  - A faculty belongs to → 1-3 departments

5. **Classes** → _cid_, _sect#_, _year_, _semester_, limit, size, classroom, capacity, days, start_time, end_time, fid
- **Primary Key:** _cid_, _sect#_, _year_, _semester_
- **Foreign Keys**:
  - cid → Courses(cid)
  - fid → Faculty(fid)
  - classroom → Classrooms(classroom)
- **Constraints**:
  - size ≤ limit ≤ capacity
  - days must be → Monday, Tuesday, Wednesday, Thursday, Friday
  - semester must be → Spring, Fall, Summer 1, Summer 2
  - A class must have at least 5 students
  - No overlapping classes in the same classroom
  - No faculty member can teach overlapping classes

6. **Enrollments** → _sid_, _cid_, _sect#_, _year_, _semester_, lgrade, ngrade
- **Primary Key** →_sid_, _cid_, _sect#_, _year_, _semester_
- **Foreign Keys** →
  - sid → Students(sid)
  - cid, sect#, year, semester → Classes(cid, sect#, year, semester)
- **Constraints**:
  - lgrade ∈ {A, B, C, D, F, I, NULL}
  - ngrade mapping: A → 4, B → 3, C → 2, D → 1, F → 0, I → NULL
  - A student must enroll in 2 to 5 classes per semester
  - A student cannot enroll in different sections of the same course
  - The student must pass prerequisites (C or better)
  - The class must not be full
  - The student must not have time conflicts

7. **Prerequisites** → _cid_, prerequisite_of(prerequisite_cid)
- **Primary Key** → _cid_, prerequisite_of (prerequisite_cid)
- **Foreign Keys** →
  - cid → Courses(cid)
  - prerequisite_cid → Courses(cid)
- **Constraints** →
  - No prerequisite cycles allowed

8. **Major_In** → _sid_, _deptname_
- **Primary Key** → _sid_, _deptname_
- **Foreign Keys** →
  - sid → Students(sid)
  - deptname → Departments(deptname)
- **Constraints** →
  - A student can have 1 or 2 majors

9. **Faculty_Department** → _fid_, _deptname_
- **Primary Key** →_fid_, _deptname_
- **Foreign Keys**:
  - fid → Faculty(fid)
  - deptname → Departments(deptname)
- **Constraints**:
  - A faculty must belong to 1-3 departments

10. **Course_Offering** → _cid_, _deptname_
- **Primary Key** → _cid_, _deptname_
- **Foreign Keys**
  - cid → Courses(cid)
  - deptname → Departments(deptname)
- **Constraints**:
  - A course must belong to exactly one department

2. [0.5 points] 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) [0.25 points] Does the combination of these two attributes, (A, B), form a key of R? Why or why not?
(b) [0.25 points] 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?

Ans:

a) (A, B) forms a key if no two tuples have the same (A, B) values. Otherwise, it does not. yes. (A, B) forms a key only if the combination of A and B uniquely identifies each tuple in the relation. If there are duplicate (A, B) pairs, then (A, B) is not a key. It could be a superkey, however. A superkey is any set of attributes that contains a key.

Note (explanation):

- A key is a minimal set of attributes that uniquely identify a tuple in a relation.
- Since A and B alone are not keys, it is possible that their combination (A, B) forms a key if it uniquely determines all tuples in R.
- However, we cannot be 100 percent certain without additional information. The combination of (A, B) would be a key only if no two tuples have the same values for both A and B.
- If duplicates exist in (A, B), then (A, B) is not a key
- **In a nutshell,** If (A, B) uniquely identifies each row, then yes, it is a key.

b)

No. If (A, B) is a key, then neither A nor B can be a superkey. A superkey must uniquely identify each tuple. If A alone were a superkey, then (A, B) would not be minimal and could not be a key. The same logic applies to B. A key is a minimal superkey (you can't remove any attributes and still have a superkey).therefor , **Neither A nor B can be a superkey** because they do not uniquely determine tuples on their own.

Note(explanation) :

- Definition of a superkey: A superkey is an attribute (or set of attributes) that uniquely identifies all tuples in the relation.
- If (A, B) is a key, it means that neither A nor B alone can uniquely determine all tuples.
- A superkey must be able to uniquely identify a tuple, and since A alone or B alone could not do that (otherwise (A, B) wouldn't be a minimal key), neither A nor B can be a superkey.
- **Therefor,** No, neither A nor B can be a superkey if (A, B) is a key.

3.

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 if the resulting ER diagram is logically equivalent to the given one.

Ans:

a) [0.25 points] Method 1:

Method 1

————————————————————

- Create a relation for Employees with EmpID as the primary key.
- Create a separate relation for Software_Projects without a foreign key.
- Introduce a Programmer_of table { EmpID{pk} , ProjectID} to maintain the many-to-many relationship.

Therefor , We can add **software_proj_id** as a **foreign key** in the **Employees** table. This means each employee will be linked to a project using its **unique ID** from the **Software_Projects table**. Since each employee is associated with at most one project, this setup works without issues.

b) [0.25 points] Method 2:

Method 2

Assignment (or Programmer_Of) → _EmpID_, _ProjectID_
- Primary Key: (EmpID, ProjectID)
- Foreign Keys:
  - EmpID → Employees(EmpID)
  - ProjectID → Software_Projects(ProjectID)

- Modify the ER model to introduce a new Assignment table with a composite key (EmpID, ProjectID).

- This avoids nulls in Software_Projects while maintaining logical equivalence.

Therefore , We can create a relation **programmer_of** consisting of **employee_id** from employees relation and **software_project_id** from **software_projects relation** as the **primary key.**

Thank You