CS532 : Homework 2
ER to Relation Transform

Name : Shriram Ananda Suryawanshi (**CS532**)

B-Number: B00734421 (Spring 2018)

CS532 : Homework 2 - ER to Relation Transform

Name - Shriram Ananda Suryawanshi (**CS532**)
B-Number - B00734421 (Spring 2018)

---

1. **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, underscore the key, specify other candidate keys (if any) and foreign keys (if any), and specify the constraints associated with this relation (including all constraints that are described in the Requirements Document).**

Answer -
Please find below Relational schema for the provided E-R diagram -

- **Students** -
  - Relational Schema - Students (sid, firstname, lastname, status, gpa, email)
  - Candidate Key - email
  - Constraints - status : any one of these - {freshman, sophomore, junior, senior, graduate}
    - gpa : decimal number between 0 and 4

- **Courses** -
  - Relational Schema - Courses (dept_code, course#, title, credits, deptname)
  - Foreign Keys - deptname (Referencing relation - Departments, attribute - deptname)
  - Constraints - course# : should be of 3 digits - between 100 and 799
    - credits : can be 3 or 4
    - If, course# is between 100 and 499, then credits is 4
    - If, course# is between 500 and 799, then credits is 3

- **Prerequisites** -
  - Relational Schema - Prerequisites (dept_code, course#, p_dept_code, p_course#)
  - Foreign Keys - dept_code, p_dept_code (Referencing relation - Courses, attribute - dept_code)
    - course#, p_course# (Referencing relation - Courses, attribute - course#)
  - Constraints - courses and their pre-requisites do not form any cycle

- **Departments** -
  - Relational Schema - Departments (deptname, office, phone#)
  - Candidate Key - office

- **Faculty** -
  - Relational Schema - Faculty (fid, firstname, lastname, rank, office, email, deptname)
  - Candidate Keys - office, email
  - Foreign Key - deptname (Referencing relation - Departments, , attribute - deptname)
  - Constraints - rank : any one of these - {lecturer, assistant professor, associate professor, professor}

- **Major** -
  - Relational Schema - Major (sid, deptname)
  - Foreign Key - sid (Referencing relation - Students, attribute - sid)
    - deptname (Referencing relation - Departments, attribute - deptname)
  - Constraints - student must have atleast 1 majoring department, and atmost 2 departments

- **Classes** -
  - Relational Schema - Classes (dept_code, course#, sect#, year, semester, start_time, end_time, limit, size, capacity, classroom, fid)
  - Foreign Keys - dept_code (Referencing relation - Courses, attribute - dept_code)
    - course# (Referencing relation - Courses, attribute - course#)
    - fid (Referencing relation - Faculty, attribute - fid)
  - Constraints - semester : any one of these - {Spring, Fall, Summer 1, Summer 2}
    - size must not exceed the limit
    - limit should not exceed the classroom capacity
    - No classes of overlapping times can be assigned to the same classroom
    - No faculty member can teach classes with overlapping times

- **Days** -
  - Relational Schema - Days (dept_code, course#, sect#, year, semester, day)
  - Foreign Keys - dept_code (Referencing relation - Classes, attribute - dept_code)
    - course# (Referencing relation - Classes, attribute - course#)
    - sect# (Referencing relation - Classes, attribute - sect#)
    - year (Referencing relation - Classes, attribute - year)
    - semester (Referencing relation - Classes, attribute - semester)
  - Constraints - day : any one of these - {Monday, Tuesday, Wednesday, Thursday, Friday}

- **Enrollment** -
  - Relational Schema - Enrollment (sid, dept_code, course#, sect#, year, semester, lgrade, ngrade)
  - Foreign Keys - sid (Referencing relation - Students, attribute - sid)
    - dept_code (Referencing relation - Classes, attribute - dept_code)
    - course# (Referencing relation - Classes, attribute - course#)
    - sect# (Referencing relation - Classes, attribute - sect#)
    - year (Referencing relation - Classes, attribute - year)
    - semester (Referencing relation - Classes, attribute - semester)
  - Constraints - lgrade : any one of these - {A, B, C, D, E, F, I, null}
    - ngrade : any one of these - { 4, 3, 2, 1, 0, null}
    - Corresponding 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
- Student must have completed all the prerequisite courses with a grade of at least C
- Student can not be registered for the classes with overlapping timings
- Student can not enroll for more than 5 courses (Minimum 1, maximum 5)
- To enroll a new student, class should have room available (size < limit)
- Each class should have atleast 5 students

2. **Let A and B be the only attributes of a relation R.**
   a. **Suppose neither A nor B is a key of R. 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?**

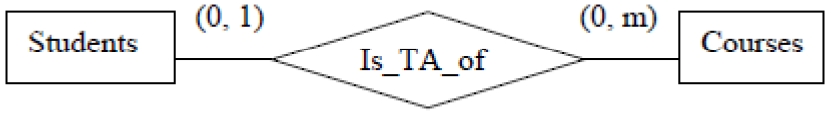
Answer -

a. Yes, the combination (A, B) forms the superkey for the relation R.
   This is because of two conditions -
   1. Relation R has only two attributes A and B, and none of it is a key.
   2. As per the Third Relational Database Rule, no two rows in the same table can be identical at any given time. That is, each tuple in a table in a unique.

   Hence, to satisfy the second condition, combination of both A and B has to be unique as individually A and B can not be used as key. Hence, combination of A and B will form a superkey.

b. No, neither A nor B could be superkey in this case.
   As the key for relationship R is formed by combination of A and B, A and B together must be satisfying both the rules to form a key - uniqueness and minimality.
   So, if we consider A or B as key, then it violates the minimality constraint for the provided key (A, B).
   Hence, A or B can not be used as key.

3. Consider the following ER Diagram:



Based on the method discussed in Chapter 4 of the Lecture Notes, this 1-to-many relationship will be transformed into a foreign key attribute in relation Students and null values cannot be avoided in this attribute. If you are a CS532 student, propose two different methods to tackle this problem such that null values can be avoided in the resulted relations/attributes. If you are a CS432 student, propose just one method to tackle this problem such that null values can be avoided in the resulted relations/attributes. You can change the ER diagram, if needed, but the changed ER diagram should be logically equivalent to the original diagram.

Answer -

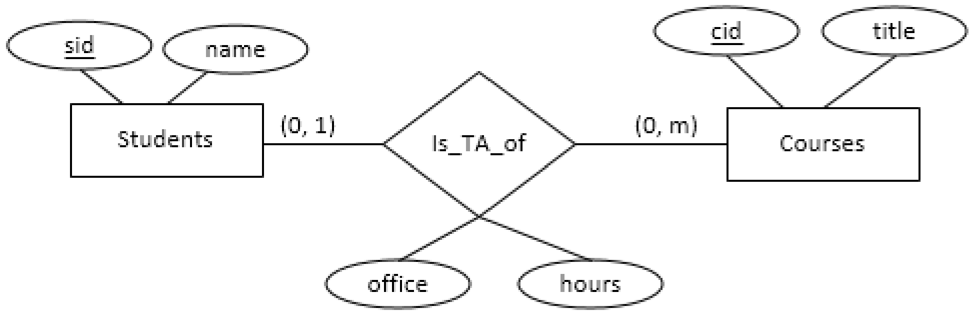
Lets assume that Students has attributes - sid, sname, gpa, office, hours and Courses has attributes - cid, title. When we will convert this ER diagram to relational schema, we get below two relation schemas -

Students (<u>sid</u>, sname, ta_cid, office, hours) {ta_cid : foreign key referencing relation - Courses, attribute - cid}

Courses (<u>cid</u>, title)

This implementation may results into many NULL values of ta_cid, office, hours in relation Students as every student is not TA. To tackle this problem, we can use any one of the below two options -

1. We can define TA specific attributes - office and hours to the relation 'Is_TA_of' as shown in below E-R diagram -



Now, the relational schema becomes -

Students (<u>sid</u>, name)

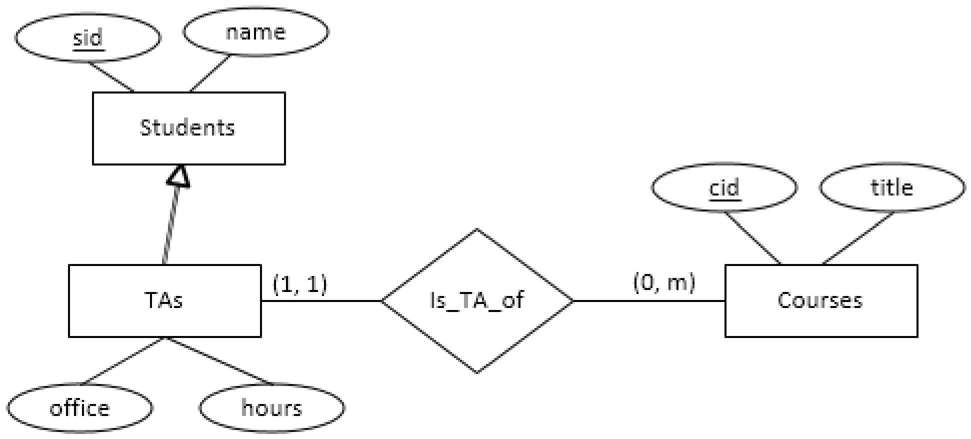
Courses (<u>cid</u>, title)

TAs (<u>sid</u>, cid, office, hours)

Here, sid and cid are foreign keys referencing Student relation key sid, and Courses relation key cid, respectively. Also, sid becomes primary key, as one student can be TA of only on course.

By this implementation, the relation schema TAs will hold only those entries of students who are TAs. Hence, it will not contain any NULL values.

2. The another way to tackle the above situation is using Specialization method. We can create Student as Superclass with TA as subclass as below -



Now, the participation of TAs entity becomes total in relation 'Is_TA_of' with courses as TAs will have only entries of students who are TA for some course.
Hence, by using Method 2 to convert IS_A hierarchy in relation schema, we get -

Students (<u>sid,</u> name)
Courses (<u>cid</u>, name)
TAs (<u>sid</u>, name, cid, office, hours)

Here, in TAs, cid is the foreign key referencing Courses relation's attribute cid.
By, this implementation as well, we can have no NULL values as every tuple in TAs will have corresponding cid.

By both the above approaches, we can avoid NULL values.
But, the approach 1 adds data redundancy, as for student who is TA, we will have details in both relations - Students and TAs. While, the approach 2 is difficult for data access as we have to use join to get any student details as if student is TA, his record will be only in TAs, and if not, details will be only in Students.

Depending upon the use of the database and the number of TAs we have, we can choose any of the above approach.