**Database Systems Homework 3**

“I have done this assignment completely on my own. I have not copied it, nor have I given my solution to anyone else. I understand that if I am involved in plagiarism or cheating, I will have to sign an official form that I have cheated and that this form will be stored in my official university record. I also understand that I will receive a grade of 0 for the involved assignment and my grade will be reduced by one level (e.g., from A to A- or from B+ to B) for my first offense, and that I will receive a grade of “F” for the course for any additional offense of any kind.”

Sincerely,

Shahu S Ronghe

B00908937

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CS532: Homework 3

**Name** - Shahu S Ronghe

**B-Number** - B00908937

1. **The following are some of the relations transformed from the ER diagram for the Student Registration System (note that some changes have been made due to the creation of a single-attribute primary key for Classes):**

**Students(B#, first\_name, last\_name, level, gpa, email, bdate)**

**Courses(dept\_code, course#, title, credits, deptname)**

**Classes(classid, dept\_code, course#, sect#, year, semester, start\_time, end\_time, limit, size, room, Faculty\_B#, TA\_B#) /\* note: classid is added to serve as a single-attribute primary key \*/**

**Faculty(B#, first\_name, last\_name, title, office, email, phone#, deptname)**

**G\_Enrollment(G\_B#, classid, lgrade, score)**

**Do the following for each relation schema, identify all non-trivial functional dependencies based on the Requirements Document (but take into consideration that classid is added as the primary key for Classes). For this question, we also make the following assumptions: (1) each dept\_code corresponds to a unique department and vice versa; (2) only faculty members in the same department could share an office and a phone number; (3) each faculty office (shared or not) has one phone with a unique number. Don’t make other assumptions about the data. Use the union rule to combine the functional dependencies as much as possible to avoid having multiple functional dependencies with the same left-hand side but different right-hand side. Furthermore, try not to list redundant FDs. For example, if you already have A  BC, you don’t need to also list A  B and A  C. As another example, if you already have A  B and B  C, you don’t need to include A  C. But you are not required to eliminate all redundant FDs at this time.**

**Answer** –

1. **Student entity:**  
   B# -> firstname, lastname, level, gpa, email, bdate

Also, every student has a unique email, so email is also a candidate key. Also, there can be a relation as email -> B# which determines every attributes.

1. **Courses entity:**  
   dept\_code course# -> title  
   dept\_code course# -> credits  
   dept\_code course# -> deptname  
     
   dept\_code course# -> title credits deptname  
     
   Course number determines credits if undergraduate then course number is between 100 and 499 and for graduate course number is between 500 and 799.  
     
   dept\_code determines deptname  
   dept\_code -> deptname
2. **Classes entity:**  
     
   classid -> dept\_code, course#, sect#, year, semester, start\_time, end\_time, limit, size, room, Faculty\_B#, TA\_B#

room -> capacity   
dept\_code, course#, sect#, year, semester ->classid

1. **Faculty entity:**  
   Since B# is primary key it determines all the attributes and using union rule,  B#-> firstname, lastname, title, office, email, phone#, deptname

Using union and transitivity rule we can say that – email -> B#

1. **G\_Enrollment entity:**  
     
   G\_B# and classid are primary key.

G\_B# classid -> lgrade score  
score -> lgrade

1. **Consider the following table schema for accounts in a banking system:**

**Accounts (acct#, owner\_id, owner\_name, acct\_type, balance, interest\_rate, time\_opened)**

**It has the following functional dependencies F = {acct# -> own\_id acct\_type balance time\_opened, owner\_id -> owner\_name, acct\_type balance -> interest\_rate}.**

**(1) (5 points) Explain why the schema is not in BCNF.**

Answer: The only candidate key is acct#. But we need all left-handed variables in FD to be a key. Since the condition for the relation to be BCNF is either left hand side of every FD is super key. This relation is not BCNF.

**(2) (14 points) Use Algorithm LLJD-BCNF to decompose it. Show the steps.**

* 1. Initialization:

D = {acct# owner\_id owner\_name acct\_type balance interest\_rate time\_opened}

* 1. 1st Iteration:

D = {owner\_id owner\_name, acct# owner\_id acct\_type balance interest\_rate time\_opened}

* 1. 2nd Iteration:

D = {owner\_id owner\_name, acct\_type balance interest\_rate, acct# owner\_id acct\_type balance time\_opened}

* 1. Final Relations

D = {owner\_id owner\_name, acct\_type balance interest\_rate, acct# owner\_id acct\_type balance time\_opened}

**(3) (5 points) Is your decomposition dependency-preserving? Justify your answer.**

Every dependence must be satisfied by at least one deconstructed table in the dependency preservation.

If a relation R is split into R1 and R2, then R's dependencies must either be part of R1 or R2, or derivable from a mixture of R1 and R2's functional dependencies.

Consider the relationship R (A, B, C, D) with the functional dependence set (A-=BC). Because FD A-=BC is a component of relation R1, the relational R is split into R1 (ABC) and R2(AD), which preserves dependence (ABC).

1. **Disprove the following rule:**

**{B -> CD, AB -> E, E -> C} |= {AE -> CD}**

**To disprove a rule, construct a relation with attributes (A, B, C, D, E) and some tuples such that the tuples of the relation satisfy the functional dependencies on the left-hand side of the rule (left of |=) but do not satisfy the functional dependency on the right-hand side of the rule.**

Consider below table -

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **B** | **C** | **D** | **E** |
| a1 | b1 | c1 | d1 | e1 |
| a2 | b1 | c1 | d1 | e2 |
| a2 | b2 | c1 | d2 | e2 |
| a1 | b3 | c1 | d3 | e1 |

Now, let’s consider the right-hand side of the given rule - AE 🡪 CD

For tuple t1, a1 e1 🡪 c1 d1 but in tuple t4 it becomes a1 e1 🡪 c1 d3

For tuple t2, a2 e2 🡪 c1 d1 but in tuple t3 it becomes a2 e2 🡪 c1 d2