Solution to CS532 Homework 3

1. Consider the following relation schemas 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 key for Classes):

Students(<u>sid</u>, firstname, lastname, status, gpa, email)
Courses(<u>dept_code, course#</u>, title, credits, deptname)
Prerequisites(<u>dept_code, course#, pre_dept_code, pre_dept_course#</u>)
Classes(<u>classid</u>, dept_code, course#, sect#, year, semester, start_time, end_time, limit, size, classroom, capacity, fid) /* note: classid is added to serve as a single attribute key */
Classes_days(<u>classid, day</u>)
Faculty(<u>fid</u>, name, rank, office, email, deptname)
Departments(<u>deptname</u>, chair, office)
Registration(<u>sid, classid,</u> lgrade, ngrade)
Student_majors(<u>sid, deptname</u>)

Do the following for each relation schema:

(a) (25 points) Identify all non-trivial functional dependencies. Don't make unrealistic assumptions about the data. Should use the union rule to combine the functional dependencies as much as possible. Furthermore, if a functional dependency is redundant (i.e., it can be derived from the ones you keep), it does not need to be included.

Answer:

Students(<u>sid</u>, firstname, lastname, status, gpa, email)
FDs: sid → firstname lastname status gpa email, email → sid

Courses(<u>dept_code, course#</u>, title, credits, deptname)
FDs: dept_code course# → title credits deptname, dept_code → deptname
Course# → credits

Prerequisites(<u>dept_code, course#, pre_dept_code, pre_dept_course#</u>)
FDs: No non-trivial FDs

Classes(<u>classid</u>, dept_code, course#, sect#, year, semester, start_time, end_time, limit, size, classroom, capacity, fid)
FDs: classid → dept_code course# sect# year semester start_time end_time limit size classroom fid,
dept_code course# sect# → classid, classroom → capacity

Classes_days(<u>classid, day</u>)
FDs: No non-trivial FDs

Faculty(<u>fid</u>, name, rank, office, email, deptname)
FDs: fid → name rank office email deptname, office → fid, email → fid

Departments(deptname, chair, office)
FDs: deptname → chair office, office → deptname

Registration(sid, classid, lgrade, ngrade)
FDs: sid classid → lgrade, lgrade → ngrade

Student_majors(sid, deptname)
FDs: No non-trivial FDs

(b) (25 points) Determine whether or not the schema is in 3NF or in BCNF. Need to provide justification.

Answer: Note that if a schema is in BCNF, it is also in 3NF.

Students(sid, firstname, lastname, status, gpa, email)
The schema is in BCNF because the left-hand side of each non-trivial FD, namely sid and email, is a superkey.

Courses(dept_code, course#, title, credits, deptname)
The schema is in not in 3NF because dept_code → deptname, we have a non-prime non-trivially depending on a non-superkey. Course# → credits also makes the schema not in 3NF.

Prerequisites(dept_code, course#, pre_dept_code, pre_dept_course#)
The schema is in BCNF because no non-trivial FD exists.

Classes(classid, dept_code, course#, sect#, year, semester, start_time, end_time, limit, size, classroom, capacity, fid)
The schema is not in 3NF because in classroom → capacity, we have a non-prime non-trivially depending on a non-superkey.

Classes_days(classid, day)
The schema is in BCNF because no non-trivial FD exists.

Faculty(fid, name, rank, office, email, deptname)
The schema is in BCNF because for each non-trivial FD, the left-hand side is a superkey.

Departments(deptname, chair, office)
The schema is in BCNF because for each non-trivial FD, the left-hand side is a superkey.

Registration(sid, classid, lgrade, ngrade)
The schema is not in 3NF because in lgrade → ngrade, we have a non-prime non-trivially depending on a non-superkey.

Student_majors(sid, deptname)
The schema is in BCNF because no non-trivial FD exists.

(c) (25) For each schema that is not in 3NF, decompose it into 3NF schemas using Algorithm LLJD-DPD-3NF. Show the result after each step of the algorithm. Are they decomposed schemas in BCNF? Justify your answer.

Answer: Only three schemas are not in 3NF and they are Courses, Classes and Registration.

Decompose Courses:

(1) All candidate keys: dept_code course#

(2) Minimal cover: dept_code course# → title, course# → credits, dept_code → deptname

(3) Decomposition: R1 = <u>dept_code course#</u> title
R2 = <u>course#</u>, credits
R3 = <u>dept_code</u> deptname

(4) No change because R1 contains the candidate key

Decompose Classes:

(5) All candidate keys: classid, dept_code course# sect#

(6) Minimal cover: classid → dept_code, classid → course#, classid → sect#, classid → year, classid → semester, classid → start_time, classid → end_time, classid → limit, classid → size, classid → classroom, classid → fid, dept_code course# sect# → classid, classroom → capacity

(7) Decomposition: R1 = <u>classid</u> dept_code course# sect# year semester start_time end_time limit size classroom fid
R2 = <u>classroom</u> capacity

(8) No change because R1 contains the candidate key

Decompose Registration:

(1) All candidate keys: sid classid

(2) Minimal cover: sid classid → lgrade, lgrade → ngrade

(3) Decomposition: R1 = <u>sid classid</u> lgrade
R2 = <u>lgrade</u> ngrade

(4) No change because R1 contains the candidate key

All decomposed schemas are in BCNF. All R2's have two attributes and they are in BCNF as a result. In all R1's, in each non-trivial FD, the left-hand side is a superkey.

2. (15 points) Prove or disprove the following rules:

(a) {X → Y, Z → W} |= {XZ → YW}

(b) {X → Y, YZ → W} |= {XY → W}

When proving a rule, use Armstrong's Axioms (i.e., reflexivity rule, augmentation rule, and transitivity rule) only. To disprove a rule, construct a relation with appropriate attributes and tuples such that the tuples of the relation satisfy the functional dependencies on the left of the rule but do not satisfy the functional dependency on the right of the rule.

Answer: (a) Prove: From X → Y (given) and IR2 (augmentation rule), we have XZ →YZ.

From Z → W (given) and IR2 (augmentation rule), we have YZ →YW.
From XZ → YZ, YZ → YW and IR3 (transitivity rule), we have XZ → YW.

(b) Disprove: Consider the following table:

X	Y	Z	W
a1	b1	c1	d1
a1	b1	c2	d2

Clearly based on the tuples in the table, X → Y is true, YZ → W is also true (the values under YZ are different), but XY → W is not true. This means that we cannot derive XY → W from X → Y and YZ → Z.

3. (10 points) Let R be a relation schema and K is a subset of the attributes of R. Prove that if K functionally determines all attributes in R – K, then K is a superkey of R. Note that you are not allowed to cite Theorem 3 in Chapter 5 of the Lecture Notes for the proof. You need to prove it based on the definitions of functional dependency and superkey.

Answer: Let X be the complete set of attributes of R. Since K → K is true (reflexivity rule) and K → R – K (given), then K → X is true (union rule). K → X means that, by the definition of FD, if two tuples in R have the same values under the attributes in K, then the two tuples will have the same values under all attributes, namely, these two tuples will be identical. Since a relation cannot have identical tuples due to the Unique Row Rule, this means no two tuples can have identical values under the attributes in K. This in turn means that K must be a superkey.