CS532 : Homework 3

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

B-Number: B00734421 (Spring 2018)

CS532 : Homework 3

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

---

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 key for Classes):**

Students(<u>sid</u>, firstname, lastname, status, gpa, email)
Courses(<u>dept_code, course#,</u> title, credits, deptname)
Classes(<u>classid</u>, dept_code, course#, sect#, year, semester, start_time, end_time, limit, size, classroom, capacity, fid)
Faculty(<u>fid</u>, firstname, lastname, rank, office, email, deptname)
Enrollments(<u>sid, classid</u>, lgrade, ngrade)

Do the following for each relation schema:

(a) Identify all non-trivial functional dependencies. Don't make unrealistic assumptions about the data. 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), don't include it.

Answer -

Please find below functional dependencies according to provided relations -

- **Students** -
  - sid → firstname lastname status gpa email
  - email → sid
- **Courses** -
  - dept_code course# → title credits deptname
  - course# → credits
  - dept_code → deptname
- **Classes** -
  - classid → dept_code course# sect# year semester start_time end_time limit size classroom capacity fid
  - classroom → capacity
  - dept_code course# sect# year semester → classid
- **Faculty** -
  - fid → firstname lastname rank office email deptname
  - email → fid
  - office → fid
- **Enrollments** -
  - sid classid → lgrade ngrade
  - lgrade → ngrade

(b) Determine whether or not the schema is in 3NF or in BCNF. Justify your conclusion.

Answer -

Based on functional dependencies mentioned in above answer, please find below conclusion for schema being 3NF or BCNF -

- **Students** -
  - BCNF - Yes, the schema is in BCNF, because -
    The left hand side of each non-trivial FD (sid and email) is a superkey
  - 3NF - Yes, as this schema is in BCNF, it is in 3NF as well

- **Courses** -
  - BCNF - No, the schema is not in BCNF, because -
    In second and third non-trivial FD 'course#' and 'dept_code' does not form a superkey.
  - 3NF - No, the schema is not in 3NF, because -
    The non-prime attributes of 'deptname' and 'credits' are non-trivially depend on the non-superkey attributes 'dept_code' and 'course#', respectively.

- **Classes** -
  - BCNF - No, the schema is not in BCNF, because -
    The left hand side 'classroom' of second non-trivial FD doesn't form a superkey.
  - 3NF - No, the schema is not in 3NF, because -
    The non-prime attribute of FD 'capacity' is non-trivially depends on non-superkey attribute 'classroom'.

- **Faculty** -
  - BCNF - Yes, the schema is in BCNF because -
    The left hand side of each non-trivial FD (fid, office and email) is a superkey
  - 3NF - Yes, as this schema is in BCNF, it is in 3NF as well

- **Enrollments** -
  - BCNF - No, the schema is not in BCNF, because -
    The left hand side 'lgrade' of second non-trivial FD doesn't form a superkey.
  - 3NF - No, the schema is not in 3NF, because -
    The non-prime attribute of FD 'ngrade' is non-trivially depends on the non-superkey attribute 'lgrade'.

(c) 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 the decomposed schemas in BCNF? Justify your answer.

Answer -

Please find below decomposition of relational schemas to 3NF which were not in 3NF -

- **Courses** -
  - Candidate Keys - dept_code course#
  - Fmin - { dept_code course# → title,
    course# → credits,
    dept_code → deptname }
  - D - { D1 (dept_code, course#, title)
    D2 (course#, credits)
    D3 (dept_code, deptname) }
  - The above decomposed schemas are in BCNF, because if we consider the FDs of each schema, the left hand side of each non-trivial FD (dept_code course#, course#, dept_code) is a superkey.

- **Classes** -
  - Candidate Keys - classid
    - dept_code course# sect# year semester
  - Fmin - { 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# year semester → classid,
    classroom → capacity }
  - D - { D1 (classid, dept_code, course#, sect#, year, semester, start_time, end_time, limit, size, classroom, fid)
    D2 (classroom, capacity)
  - The above decomposed schemas are in BCNF, because if we consider the FDs of each schema, the left hand side of each non-trivial FD (classid, classroom) is a superkey.

- **Enrollments** -
  - Candidate Keys - sid classid
  - Fmin - { sid classid → lgrade,
    lgrade → ngrade }
  - D - { D1 (sid, classid, ngrade),
    D2 (lgrade, ngrade }
  - The above decomposed schemas are in BCNF, because if we consider the FDs of each schema, the left hand side of each non-trivial FD (sid classid, lgrade) is a superkey.

2. Prove or disprove the following rules:

(a) {B → CD, AB → E, E → C} |= {AE → CD}

(b) {B → CD, AD → E} |= {AB → E}

When proving a rule, you can use all the six inference rules (i.e., reflexivity rule, augmentation rule, transitivity rule, decomposition rule, union rule and pseudo transitivity rule). 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) -

The given rule is not valid, hence we are disproving 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

The above table satisfies all the FDs mentioned on the left-hand side of the rule.

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

This is not valid as per the definition of FD, i.e. if X → Y, then for any t1, t2 in r(R), if t1 and t2 have the same X-value, then t1 and t2 also have the same Y-value.

Hence, above rule is not valid as per the functional dependencies.

(b) -

The given rule is valid, hence we are proving the rule.

Proof -

1. Given FDs from left hand side of the rule - 1. B → CD 2. AD → E
2. Using Decomposition rule on FD 1 - B → C and B → D
3. Now, from FD 1 and 2 we have now - B → D and AD → E
4. By using Pseudotransitivity rule - AB → E

   OR

   By using Augmentation rule - B → D can be AB → AD, and we have AD → E

   Hence, AB → E

This is the right side of the given rule.

Hence, we are able to derive the right hand side of the rule from left hand side, the rule is valid and proved.

3. **Write relational algebra expressions to answer the following query statements based on the tables given in page 3 of this document. For each query, show the result that would be obtained if the query were actually executed against the tables. It is OK to answer a query in multiple steps. You need to make sure that your relational algebra expressions are reasonably optimized as we discussed in class, i.e., conditions involving a single table should be specified on that table directly and Cartesian products should be replaced by joins whenever possible.**

   a. **Find the dept_code, course# and title of each course that was offered in the Spring semester of 2014.**

   - Expression -

$$\pi_{\text{Courses.Dept\_code, Courses.Course\#, Courses.Title}} (\text{Courses} \bowtie \sigma_{\text{Classes.Year = '2014' and Classes.Semester = 'Spring'}} (\text{Classes}))$$

   - Expected Output -

Dept_code	**Course#**	**Title**
CS	432	database systems
CS	240	data structure
CS	532	database systems
Math	221	calculus I

   b. **Find the first name of each student who has taken at least one CS course and at least one Math course.**

   - Expression -

$$\pi_{\text{Students.Firstname}} (\text{Students} \bowtie (\pi_{\text{sid}} (\sigma_{\text{Classes.Dept\_code = "Math'}} (\text{Classes}) \bowtie \text{Enrollments}) \cap \pi_{\text{sid}} (\sigma_{\text{Classes.Dept\_code = "CS'}} (\text{Classes}) \bowtie \text{Enrollments}))$$

   - Expected Output -

Firstname
Anne

c. Find the dept_code and course# of each course that was not offered in 2013.

- Expression -

$$\pi_{\text{Courses.Dept\_code, Courses.Course\#}} (\text{Courses}) - \pi_{\text{Classes.Dept\_code, Classes.Course\#}} (\sigma_{\text{Classes.Year = '2013'}} (\text{Classes}))$$

- Expected Output -

Dept_code	**Course#**
CS	240
Math	221
CS	532
CS	552
BIOL	425

d. Find the sid and first name of every student who has taken all CS classes offered in Spring 2014.

- Expression -

$$\pi_{\text{Students.Sid, Students.Firstname}} (\text{Students} \bowtie (\pi_{\text{Classes.Sid, Classes.Classid}} (\text{Enrollments}) \div \pi_{\text{Classes.Classid}} (\sigma_{\text{Classes.dept\_code = 'CS' and lasses.Year = '2014' and Classes.Semester = 'Spring'}} (\text{Classes}))))$$

- Expected Output -

Sid	**Firstname**
B001	Anne