HW3
Wanzhu Chen

Problem1:

(a) a→b: **No,** since when a=1, b could be 2 or 3 or 4.
(b)a→c: **Yes**. Since for each a, c only has one value.
(c) b→a: **Yes**. Although when b=2 or 3 or 4, a has the same value 1, for each b, a only has a specific value, no multiple choices, so it satisfies.
(d) b→c: **Yes**. The same reason as (c)
(e) c→a: **Yes**. The same reason as (c)
(f) c→b: **No**. Since when c=2, b could be 2 or 3 or 4 or 5.
(g) ab→c: **Yes**. For each combination of ab, there is a specific value of c.
(h) ac→b: **No**. Since when a=1, c=2, b could be 2 or 3 or 4. It is not a single result for b.
(i) bc→a: **Yes**. Since a only has one value for each combination of bc.

Problem2:

ca, cb, cd and ce are the candidate keys.
(1) ca has a closure including {abcde}, ca is candidate key.
(2) cb has a closure including {cbdea}, cb is candidate key.
(3) Since cd→e, e→a, a→b, b→d, with Transitivity rule from Armstrong's axioms, cd→ e,a,b,d. Thus cd is the candidate key.
(4) Since e→a, a→b, b→d, with Transitivity rule, e→abd.
Then apply Augmentation rule by adding c to both sides at the same time, so I have ce→cabd. Thus ce is one candidate key.

Problem3:

(a) **af is the candidate key.**

Reason:

a→bcd-----------(FD1)
ad→e------------(FD2)
efg→h-----------(FD3)
f→gh------------(FD4)

Start with af, put af into its closure:{af}.

Firstly, considering a→bcd, and f→gh, I can safely add bcd and gh into the closure: {abcdfgh}

Next consider ad→e. Since both ad is in the closure of af, add e into the closure: {abcdefgh}, which is R.

Thus, af is the candidate key.

(b) **efg→h can be removed.**

Reason:

f→gh could separate as f→g and f→h using Decomposition Rule.

Also, when compare f→g, f→h to efg→h, I could find e and g are extraneous attributes. If efg→h is deleted, it will make no differences for F+.

So, efg→h can be removed.

Problem4:

(a) **The only possible candidate key is abe.**

Start with {abe} as the closure of abe itself.

Firstly adding c and d into the closure after considering a→c and b→d: {abecd};

Then since I have de→f and both d and e are in the closure, I could safely add f into the closure:{abecdf}, which is R.

Thus, abe is candidate key.

(b) **Add a→e as a new FD.**

Since a→b, b→d, with Transitivity rule, a→d.

Since a→d and a→e, a→de.

Sine a→de, de→f, so a→f.

Since a→c, a→b(additional), a→d, a→f, a→e(added), with Union rule a→bcdef, which means a is the candidate key.

Problem 5:

(a) Nontrival functional dependencies:

a→b
b→a
c→b
ac→b

(b) **It is not BCNF. it is a→b, b→a that violate BCNF.**

Since the candidate key is *c*, and *a* or *b* are not superkeys while they are determinants in some of the FDs(a→b, b→a). Thus, it is a→b, b→a that violate BCNF.

(c) **Valid decomposition:**

R1(<u>a</u> <u>b</u>), R2(a b <u>c</u>), and a, b in R2 are foreign keys reference to a, b in R1 respectively.

Problem 6:

(a) **ab, bc, bde are candidate keys.**

ab→c, so given ab, I could have abc to closure of ab.
bc→d, so, add d to closure of ab.
cd→e, so add e to closure of ab.
Thus, ab is a candidate key, since its closure is {abcde} that covers all the attributes of R.

bc→d, so add d to closure of bc. Now the closure of bc is {bcd}
cd→e, so add e to closure of bc. Now the closure of bc is {bcde}
de→a, so add a to closure of bc. Now the closure of bc is {abcde}, which is R.
Thus, bc is a candidate key.

de→a, so add a to closure of bde. Now the closure of de is {abde}
ab→c, so add c to closure of bde. Now the closure of de is {abcde}, which is R.
Since neither bd or be could has a closure containing all attributes of R, bde is a candidate key.

(b) **cd→e and de→a** violates BCNF, since neither cd nor de is a superkey,

(c) **Decomposition1: R1(<u>c d</u> e), R2(<u>b c</u> d) , R3(<u>a</u> <u>b</u> c)**
Decomposition2: R1(<u>c d</u> e), R2(<u>b c</u> d) , R3(<u>d</u> <u>e</u> a)

Problem7

(a)

FD1: course_id, sec_id, semester, year$\rightarrow$building, room_no
FD2: course_id, sec_id, semester, year$\rightarrow$time_slot_id
FD3: course_id, sec_id, semester, year, s_ID$\rightarrow$grade
FD4: course_id, sec_id, semester, year$\rightarrow$i_ID

(b) **Candidate key: {course_id, sec_id, semester, year, s_ID}**

(c) **No, it is not BCNF.**

The reason is that on the left hand side of FD1, FD2, FD4, they only have {course_id, sec_id, semester, year} and it is not a superkey. Also, FD1, FD2 and FD4 all three are not trivial.

Decomposition:

R1(course_id, sec_id, semester, year, building, room_no)
R2(course_id, sec_id, semester, year, time_slot_id)
R3(course_id, sec_id, semester, year, i_ID)
R4(course_id, sec_id, semester, year, s_ID, grade)

(d)

update anomalies:

It may contain records in takesSecTaughtBy table that have the same course_id, sec_id, semester, year, building, room_no while have different s_IDs, which reflects one section is scheduled in exactly one building and room and can have many students enrolled. Then if I want to change the place of this section, I have to update the building and room_no values in all tuples that are related to this section.

delete anomalies:

If I have to delete one section, which has many tuples in takesSecTaughtBy table, since there may be many students register it, then I have to delete all the records. It is really not efficient to do such redundant work like this. Also, when I delete this section, I have to delete other information like who have registered this section at the same time.

insert anomalies:

If I want to insert a new section with only course_id, sec_id, semester and year value while who registered for this section is not set, then it may be no way to insert a new record like this, since the candidate key is {course_id, sec_id, semester, year, s_ID}, and the candidate key can't have null value.

(e)

Based on the requirements, the FDs are:

FD1: course_id, sec_id, semester, year$\rightarrow$building, room_no

FD2: course_id, sec_id, semester, year→time_slot_id
FD3: course_id, sec_id, semester, year, s_ID→grade
FD4: course_id, sec_id, semester, year→i_ID

FD5: s_ID, i_ID→course_id
FD6: s_ID, course_id→i_ID
FD7: course_id→i_ID

Simplify these FDs by deleting FD4 and FD6. So I have:
FD1: course_id, sec_id, semester, year→building, room_no
FD2: course_id, sec_id, semester, year→time_slot_id
FD3: course_id, sec_id, semester, year, s_ID→grade
FD5: s_ID, i_ID→course_id
FD7: course_id→i_ID

(f) **Candidate keys are:**
{course_id, sec_id, semester, year, s_ID} and
{s_ID, i_ID, sec_id, semester, year}

(g) **No, it is not BCNF,** since FD1, FD2, FD5, FD7 are not trivial and don't have superkeys on the left hand side.

Decomposition to BCNF:
R1(<u>course_id, sec_id, semester, year</u>, building, room_no)
R2(<u>course_id, sec_id, semester, year,</u> time_slot_id)
R3(<u>course_id, sec_id, semester, year, s_ID,</u> grade)
R4(<u>course_id,</u> i_ID)
R5(<u>course_id, sec_id, semester, year, s_ID</u>)

Problem 8

(a)

Answer:

It is neither BCNF nor 3NF.

Decompostition: R1(b c), R2(a b). It is lossless join and dependency preserved.

Analysis:

Since a→b, b→c, so a is the only candidate key. Thus, it is not BCNF since b→c violates BCNF rule that b is not a superkey.

Since b→c violates 3NF that b is not a superkey while c on the right hand side is not part of candidate key, it is not 3NF.

Decomposition: R1(b c), R2(a b)

R1 ∩ R2 = b, which is the primary key of R1, so it is lossless join decomposition.

Checking each FD founds that every FD holds on either R1 or R2, thus it is dependency-preserved.

(b)

Answer:

It is not BCNF but it is 3NF.

Analysis:

Candidate keys are ab, cb, db.

Since either c or d is a superkey while both of them are on the left hand side of c→d and d→a, so c→d, d→a violates BCNF. It is not BCNF.

However, since the right hand side of c→d, d→a are d and a, and both of them are part of candidate keys, specifically part of db or ab, thus it is a 3NF.

(c)

Answer:

It is neither BCNF nor 3NF.

Decomposition: R1(b d), R2(b c), R3(a b), and it is a dependency-preserved and lossless join decomposition.

Analysis:

Candidate key is ab.

Both FDs violates BCNF, since b itself is not a superkey.

Also, the right hand side of these two FDs are c and d, but neither of them are part of candidate key, so it is not 3NF.

Decomposition: R1(b d), R2(b c), R3(a b)

It is a lossless join decomposition, since

R1 ∩ R2 = b and b is the candidate key of R1;

R2 ∩ R3 = b and b is the candidate key of R2;

R1 ∩ R3 = b and b is the candidate key of R1.

It is a dependency-preserved decomposition since each FD holds on one of the relationships.

(d)

Answer:

It is BCNF. It is 3NF.

Analysis:

Candidate keys are ab, bc, ad and cd.

For those four FDs, they all have superkeys on the left hand side, thus it is BCNF. There is no doubt that if it is BCNF, it is 3NF.