CS 121 Assignment 6

1.

1. There are no functional dependencies
2. B → A
3. A → B
4. A → B, B → A

2.

**Union**

We want to prove if 𝞪 → 𝞫 holds and 𝞪 → 𝞬 holds, then 𝞪 → 𝞫𝞬 holds.

By augmentation of the first dependency with 𝞪 we can get 𝞪 → 𝞫𝞪 holds

By augmentation of the second dependency with 𝞫 we can get 𝞫𝞪 → 𝞫𝞬 holds

Thus, by transitivity of the two augmented dependencies we have 𝞪 → 𝞫𝞬 holds as desired

**Decomposition**

We want to prove if 𝞪 → 𝞫𝞬 holds, then 𝞪 → 𝞫 holds and 𝞪 → 𝞬 holds.

By reflexivity we have 𝞫𝞬 → 𝞫 holds because 𝞫 is clearly a subset of 𝞫𝞬.

Thus, by transitivity we have 𝞪 → 𝞫 holds

Similarly, by reflexivity we have 𝞫𝞬 → 𝞬 holds

So, by transitivity we have 𝞪 → 𝞬 holds

Thus, we have proved decomposition.

**Pseudotransitivity**

We want to show if 𝞪 → 𝞫 holds and 𝞫𝞬 → 𝞭 holds, then 𝞪𝞬 → 𝞭 holds.

By augmenting the first equation with 𝞬, we have 𝞪𝞬 → 𝞫𝞬 holds

Thus, by transitivity we have 𝞪𝞬 → 𝞭 holds as desired.

3.

1. A is a candidate key. We will show it’s a superkey. We’ll start with a+ = A

From A → BC and decomposition, a+ = ABC. From B → D and CD → E, we get A+ = ABCDE. Thus, A is a superkey. It’s also obviously a candidate key because it is minimal.

E is also a candidate key. E → A so from the above implications we can see that E+ = ABCDE. E is also clearly a candidate key.

BC is also a candidate key (bc)+ = BCD from B → D. Then from CD → E, (bc)+ = BCDE. Finally from E → A, (BC)+ = ABCDE. Thus, BC is a superkey. BC is a candidate key because B+ = BD and C+ = C.

CD is also a candidate key. CD → E so (cd)+ = CDE. E → A so (cd)+ = ACDE. A → BC so by decomposition, (CD)+ = ABCDE. Thus. CD is a super key. D+ = D and C+ = C so CD is a candidate key.

b) All trivial dependencies 𝞪 → 𝞫 where 𝞪 is any subset of ABCDE and 𝞫 is a subset of 𝞪.

All dependencies 𝞬𝞪 → 𝞫 where 𝞪 is one of the candidate keys, 𝞬 is a subset of

[ABCDE - 𝞪] and 𝞫 is a subset of ABCDE. We also have B → D and B → BD that are

non-trivial dependencies. This is F+.

4. We will show A →→ BC doesn’t imply A →→ B and A →→ C by counter example

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| tuple | A | B | C | D |
| t1 | a1 | b1 | c1 | d1 |
| t2 | a1 | b2 | c2 | d2 |
| t3 | a1 | b1 | c1 | d2 |
| t4 | a1 | b2 | c2 | d1 |

This clearly satisfies the definition of A →→ BC. However, for A →→ B we would need t2[R - 𝞫] = t3[R - 𝞫] and this isn’t true in the above table because C for t2 and C for t3 aren’t equal. This argument also holds for A →→ C. Thus, A →→ BC doesn’t imply A →→ B and A →→ C.

5.

1. F = {A → E, BC → D, C → A, AB → D, D → G, BC → E, D → E, BC → A }

First we will apply the union rule to BC → D, BC → E, and BC → A to get:

F = {A → E, BC → DEA, C → A, AB → D, D → G, D → E }

Now we will apply the union rule to D → G and D → E to get:

F = {A → E, BC → DEA, C → A, AB → D, D → GE}

D is extraneous in BC → DEA because C → A and AB → D so BC → D is implied by pseudotransitivity by F. So we’re left with:

F = {A → E, BC → EA, C → A, AB → D, D → GE}

E is extraneous in BC → EA because C → A and A → E so C → E by transitivity so BC → E is implied by F. So we’re left with:

F = {A → E, BC → A, C → A, AB → D, D → GE}

B is extraneous in BC → A because C → A is obviously implied elsewhere in F. Thus, we are left with:

Fc = {A → E, C → A, AB → D, D → GE}

b) We have (bc)+ = (BCA) because C → A and A → E so (bc)+ = (ABCE) and AB → D

and D → GE so using decomposition we get (BC)+ = (ABCDEG). Thus, BC is a superkey.

To show it’s a candidate key we will show B+ and C+ don’t contain R. B+ = B and C+ = ACE

so, BC is a candidate key.

c) C → A is a non trivial dependency and C is not a superkey so we will we will replace R

with two schemas: R1 = CA and R2 = BCDEG. D → GE is also a nontrivial dependency

and D isn’t a superkey for R2. Thus, we will further split R2. So we have R1 = CA and

R2 = DEG and R3 = BCD. These schema are clearly in BCNF because now we only have

2 non-trivial functional dependencies left, C → A and D → GE and C is a superkey for R1

because C+ = CA and D is a superkey for R2 because D+ = DGE. R3  clearly has no

non-trivial functional dependencies. Thus, the schemas satisfy the conditions to be in

BCNF form. Note that AB → D and A → E are not preserved by this BCNF

decomposition.

d) AB → D is a non trivial dependency and AB is not a superkey so we will we will replace R

with two schemas: R1 = ABD and R2 = ABCEG. C → A is also a nontrivial dependency

and C isn’t a superkey for R2. Thus, we will further split R2. So we have R1 = ABD and

R2 = CA and R3 = BCEG. These schema are clearly in BCNF because now we only have

2 non-trivial functional dependencies left, C → A and AB → D and C is a superkey for R2

because C+ = CA and AB is a superkey for R1 because (AB)+ = ABD. R3  clearly has no

non-trivial functional dependencies. Thus, the schemas satisfy the conditions to be in

BCNF form. Note that DE → G and A → E are not preserved by this BCNF

decomposition.

e) Using the 3NF algorithm we get the following schemas.

R1 = (A, E)

R2 = (C, A)

R3 = (A, B, D)

R4 = (D, G, E)

R5 = (B, C)

6.

1. {course\_id, section\_id, term, year} is a candidate key. We will show it’s a superkey.

{course\_id, section\_id, term, year}+ =

(course\_id, section\_id, term, year, meet\_time, room, num\_students, instructor\_id) because {course\_id, section\_id, term, year} → {meet\_time, room, num\_students, instructor\_id}. Finally, {course\_id, section\_id, term, year}+ =

(course\_id, section\_id, term, year, meet\_time, room, num\_students, instructor\_id, dept, units, course\_level) because {course\_id } → { dept, units, course\_level }. Thus, {course\_id, section\_id, term, year} is a superkey so we say it’s a candidate key.

{room, meet\_time, term, year } is also a candidate key. We will show it’s a superkey. {room, meet\_time, term, year }+ = (room, meet\_time, term, year, instructor\_id, course\_id, section\_id) because {room, meet\_time, term, year } → {instructor\_id, course\_id, section\_id}. Now we have {room, meet\_time, term, year }+ = (course\_id, section\_id, term, year, meet\_time, room, num\_students, instructor\_id) because {course\_id, section\_id, term, year} → {meet\_time, room, num\_students, instructor\_id}. Finally we get {room, meet\_time, term, year }+ = (course\_id, section\_id, term, year, meet\_time, room, num\_students, instructor\_id, dept, units, course\_level) because {course\_id } → { dept, units, course\_level }. Thus, {room, meet\_time, term, year } is a superkey so we say it’s a candidate key.

b) instructor\_id is extraneous in the second dependency. This is because when we take it out of the relation and make F’ = ({course\_id } → { dept, units, course\_level }, {course\_id, section\_id, term, year} → {meet\_time, room, num\_students}, {room, meet\_time, term, year } → {instructor\_id, course\_id, section\_id}). By transitivity we have {course\_id, section\_id, term, year} → {instructor\_id, course\_id, section\_id}. Thus, by decomposition we have {course\_id, section\_id, term, year} → {instructor\_id}. Thus, Fc = ({course\_id } → { dept, units, course\_level }, {course\_id, section\_id, term, year} → {meet\_time, room, num\_students}, {room, meet\_time, term, year } → {instructor\_id, course\_id, section\_id}).

Instructor\_id is extraneous in the third relation. We have F’ = ({course\_id } → { dept, units, course\_level }, {course\_id, section\_id, term, year} → {meet\_time, room, num\_students, instructor\_id}, {room, meet\_time, term, year} → {course\_id, section\_id}). Thus, by transitivity we have {room, meet\_time, term, year} → {meet\_time, room, num\_students, instructor\_id} because {room, meet\_time, term, year} → {course\_id, section\_id}. So, by decomposition {room, meet\_time, term, year} → {instructor\_id}. Therefore, instructor\_id is extraneous and Fc = ({course\_id } → { dept, units, course\_level }, {course\_id, section\_id, term, year} → {meet\_time, room, num\_students, instructor\_id}, {room, meet\_time, term, year} → {course\_id, section\_id})

The second Fc which has instructor\_id removed from schema 3 is most appropriate because it makes more sense because when you use the database you’ll choose a course and a section number (and obviously a term and year) and see who is teaching a class. You wouldn’t choose a room and time and then see who is teaching in that room and time.

c) 3NF is the best normal form for this. This is because there are no multivalued dependencies, so 4NF and 5NF are ruled out. It is already in 1NF as is. BCNF isn’t the best because of lossy decomposition. So, 3NF is going to be the best choice.Thus, the schemas we have are

R1 = (course\_id, dept, units, course\_level)

* (course\_id) is a primary key

R2 = (course\_id, section\_id, term, year, meet\_time, room, num\_students)

* (course\_id) is a foreign key that references R1
* (course\_id, section\_id, term, year) is a candidate key and a primary key

R3 = (room, meet\_time, term, year, instructor\_id, course\_id, section\_id)

* room, meet\_time, term, year, course\_id, and section\_id are foreign keys that reference R2
* (room, meet\_time, term, year) is a candidate key and a primary key
* (course\_id, section\_id, term, year) is a candidate key

7. email\_info = (email\_id, send\_date, from\_addr, subject, email\_body)

* email\_id →→ send\_date, from\_addr, subject, email\_body is the relevant dependency for this schema
* (email\_id) is clearly a superkey for this relation so it’s in 4NF
* (email\_id) is the primary key

send\_to = (email\_id, to\_addr)

* email\_id →→ to\_addr is the relevant dependency for this schema
* The multivalued dependency is clearly trivial in this schema so it’s in 4NF
* (email\_id, to\_addr) is the primary key
* (email\_id) is a foreign key referencing email\_info

attachment = (email\_id, attachment\_name, attachment\_body)

* email\_id, attachment\_name →→ attachment\_body is the relevant dependency for this schema
* (email\_id, attachment\_name) is clearly a superkey for this relation so this relation is in 4NF
* (email\_id, attachment\_name) is the primary key
* (email\_id) is a foreign key referencing email\_info