Alex Janosi

CS121 Problem Set #6

1a. No dependencies since one value of a doesn't map to one value of b and vice versa

1b. b $\rightarrow$ a since a can have multiple b values but not vice versa

1c. a $\rightarrow$ b since same rational as part b

1d. a $\rightarrow$ b and b $\rightarrow$ a since each pair is unique

2. Union rule: If $\alpha \rightarrow \beta$ holds, and $\alpha \rightarrow \gamma$ holds, then $\alpha \rightarrow \beta\gamma$ holds.

Given $\alpha \rightarrow \beta$, $\alpha\alpha \rightarrow \beta\alpha$ by the Augmentation rule

Given $\alpha \rightarrow \gamma$, $\alpha\beta \rightarrow \gamma\beta$ by the Augmentation rule

Since $\alpha\alpha \rightarrow \beta\alpha$ and $\alpha\beta \rightarrow \gamma\beta$, $\alpha\alpha \rightarrow \gamma\beta$ by the Transitivity rule

Thus, $\alpha \rightarrow \gamma\beta$ since $\alpha\alpha$ is two identical sets

Decomposition rule: If $\alpha \rightarrow \beta\gamma$ holds, then $\alpha \rightarrow \beta$ holds and $\alpha \rightarrow \gamma$ holds.

Given $\beta\gamma$, $\beta\gamma \rightarrow \beta$ and $\beta\gamma \rightarrow \gamma$ by the Reflexivity rule

Given $\alpha \rightarrow \beta\gamma$ and $\beta\gamma \rightarrow \beta$, $\alpha \rightarrow \beta$ by the Transitivity rule

Given $\alpha \rightarrow \beta\gamma$ and $\beta\gamma \rightarrow \gamma$, $\alpha \rightarrow \gamma$ by the Transitivity rule

Pseudotransitivity rule: If $\alpha \rightarrow \beta$ holds, and $\gamma\beta \rightarrow \delta$, then $\alpha\gamma \rightarrow \delta$ holds.

Given $\alpha \rightarrow \beta$, $\alpha\gamma \rightarrow \beta\gamma$ by Augmentation rule

Given $\alpha\gamma \rightarrow \beta\gamma$ and $\gamma\beta \rightarrow \delta$, then $\alpha\gamma \rightarrow \delta$ by Transitivity rule

3a. Must find candidate keys that are a superkey for R:

NOTE: $\alpha \rightarrow \alpha$ for any set by Reflexivity

Given A $\rightarrow$ BC, A $\rightarrow$ B and A $\rightarrow$ C by Decomposition

Since B $\rightarrow$ D and A $\rightarrow$ B, A $\rightarrow$ D by Transitivity

Given A $\rightarrow$ CD by Union (A $\rightarrow$ C and A $\rightarrow$ D) and CD $\rightarrow$ E, A $\rightarrow$ E by Transitivity

Thus, A $\rightarrow$ ABCDE by Union so A is a candidate key

Since A $\rightarrow$ ABCDE and E $\rightarrow$ A, E $\rightarrow$ ABCDE by Transitivity

So, E is also a candidate key

Since CD → E and E → ABCDE, CD → ABCDE by Transitivity

CD is a candidate key

Given B → D, BC → CD by Augmentation

Given BC → CD and CD → E, BC → ABCDE by Transitivity

BC is a candidate key

Thus, (A, E, BC, CD) are the candidate keys

3b. We can use the candidate keys to find all dependencies:

A → ABCDE and all dependencies generated from this by applying the Decomposition rule

E → ABCDE and all dependencies generated from this by applying the Decomposition rule

BC → ABCDE and all dependencies generated from this by applying the Decomposition rule

CD → ABCDE and all dependencies generated from this by applying the Decomposition rule

Adding to this, we can create a ton of function dependencies by putting a candidate key and any set of attributes from R on the left and any subset of R on the right. Since the candidate key being paired with other sets in R will create a ton of other dependencies.

Then, we must consider the last restriction in F:

B → D

B → BD by Augmentation and Union

B → B by Decomposition

BD → BD by Augmentation of B → B

BD → B and BD → D by Decomposition

Lastly, we need the remaining trivial dependencies:

C → C and D → D by Reflexivity

All the mentioned dependencies make up F^+.

4. Given R(A, B, C, D), does A →→ BC logically imply A →→ B and A →→ C?

No, given A →→ BC, only A→→ B or A →→ C can be satisfied, not both. Take for example this table:

A	B	C	D
1	1	1	3
1	1	2	4
1	1	1	4
1	1	2	3

This satisfies A →→ BC since t1[A] = t2[A] = t3[A] = t4[A], t1[BC] = t3[BC] and t2[BC] = t4[BC], t1[R – BC] = t4[R – BC] and t2[R – BC] = t3[R – BC] (by definition). We can also infer by that definition that A →→ C by the same definition. However, we see that this does not hold for A →→ B since the last part of the definition does not hold (t1[R – B] = t4[R – B] and t2[R – B] = t3[R – B]). If we create a similar table having all the C values equal (e.g. flip the B and C columns above), we get A →→ B but not A →→ C.

5a. F = { A → E, BC → D, C → A, AB → D, D → G, BC → E, D → E, BC → A }

BC → D is extraneous in F since C → A becomes BC → AB by Augmentation

And BC → AB with AB → D becomes BC → D by Transitivity

{A → E, C → A, AB → D, D → G, BC → E, D → E, BC → A}

Union rule on D → E and D → G

{A → E, C → A, AB → D, D → EG, BC → E, BC → A}

BC → E is extraneous in F since BC → A and A → E by Transitivity

{A → E, C → A, AB → D, D → EG, BC → A}

B is extraneous in BC → A since C → A and join duplicates of C → A

F_C = { A → E, C → A, AB → D, D → EG }

5b. Compute BC^+ since BC is common

BC^+ = BC and BC → D, BC → E, BC → A

BC^+ = ABCDE and D → G

BC^+ = ABCDEG so BC is a superkey

Next, find B^+ and C^+:

B^+ = B and there's nothing else we can do

C^+ = C and C → A

C^+ = CA and A → E

C^+ = CAE and that is it

Since attribute-set closure of all subsets is not R, BC is a candidate key.

5c. We consider each dependency in F: A → E, C → A, D → E, D → G, AB → D are not in BCNF while BC → D, BC → E, BC → A are in BCNF since BC is a candidate key.

Take out A → E first:

R_1 = (A, E) where A is primary key of R_1 so in BCNF

R_2 = (A, B, C, D, G) take out D → G

R_2 = (D, G) where D is primary key of R_2 so in BCNF

R_3 = (A, B, C, D) take out C → A

R_3 = (C, A) where C is primary key of R_3 so in BCNF

R_4 = (B, C, D) which is in BCNF because BC is candidate key

So, final answer:

R_1 = (A, E) where A is primary key of R_1 (A → E)

R_2 = (D, G) where D is primary key of R_2 (D → G)

R_3 = (C, A) where C is primary key of R_3 (C → A)

R_4 = (B, C, D) since BC is candidate key (BC → D is valid)

Dependencies lost: AB → D, D → E

5d. Pick D $\rightarrow$ EG first:

R_1 = (D, E, G) where D is primary key of R_1 so in BCNF

R_2 = (A, B, C, D) take out C -> A

R_2 = (C, A) where C is the primary key of R_2 so in BCNF

R_3 = (C, B, D) and BC is candidate key of R given F so in BCNF

So, final answer:

R_1 = (D, E, G) where D is primary key of R_1 (D $\rightarrow$ EG)

R_2 = (C, A) where C is the primary key of R_2 (C $\rightarrow$ A)

R_3 = (C, B, D) where BC is candidate key (BC $\rightarrow$ D is valid)

Lost Dependencies: A $\rightarrow$ E and AB $\rightarrow$ D

5e. Using F_C and adding a candidate key since rest do not have one:

R_1(A, E) where A is the primary key of R_1 (A $\rightarrow$ E)

R_2(C, A) where C is the primary key of R_2 (C $\rightarrow$ A)

R_3(D, E, G) where D is the primary key of R_3 (D $\rightarrow$ EG)

R_4(A, B, D) where AB is the primary key of R_4 (AB $\rightarrow$ D)

R_5(B, C) where BC is a candidate key

6. Let R{ course_id, section_id, dept, units, course_level, instructor_id, term, year, meet_time, room, num_students } = R { A, B, C, D, E, G, H, I, J, K, L }

6a. Given F = { A $\rightarrow$ CDE, ABHI $\rightarrow$ JKLG, KJHI $\rightarrow$ GAB }

Take $\{KJHI\}^+$ = KJHI and KJHI $\rightarrow$ GAB so $\{KJHI\}^+$ = KJHIGAB

A $\rightarrow$ CDE so $\{KJHI\}^+$ = KJHICDEGAB

ABHI $\rightarrow$ JKLG so $\{KJHI\}^+$ = ABCDEGHIJKL

KJHI is a candidate key

$\{ABHI\}^+$ = ABHI and ABHI $\rightarrow$ JKLG

$\{ABHI\}^+$ = ABHIJKLG and A $\rightarrow$ CDE

$\{ABHI\}^+$ = ABHIJKLGCDE = ABCDEGHIJKL

ABHI is a candidate key

Thus, { course_id, section_id, term, year } and { room, meet_time, term, year } are candidate keys

6b. Looking at the last two dependencies, we can see that the G on the right hand side is extraneous. Thus, our two canonical covers are (only difference is instructor_id on right side of second and third dependencies):

{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 }

And

{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 }

It would make more sense to have instructor_id in the second dependency. The second dependency gives information about a course at a time, which makes sense to have the instructor_id in. The last dependency describes a location and time for a class and section, which does not need the instructor_id.

6c. BCNF decomposition: We have F_C = { A → CDE, ABHI → JKLG, KJHI → AB}

Start with (A, B, C, D, E, G, H, I, J, K, L) and take out A → CDE

R_1(A, C, D, E) where A is primary key of R_1

(A, B, G, H, I, J, K, L) take out ABHI → JKLG and KJHI → AB

R_2(A, B, H, I, J, K, L, G) where ABHI is candidate key and KJHI is a candidate key

3NF: Using F_C and adding candidate keys:

R_1(A, C, D, E) where A is primary key of R_1

R_2(A, B, H, I, J, K, L, G) where ABHI is candidate key and KJHI is a candidate key

These are the same thing. Thus, we must look at the perks of each given the size. Since this is a relatively small database, we do not have to worry about runtime. Thus, we care about the correct and complete representation in the database. For this reason, the 3NF version would be better.

7. Let emails(email_id, send_date, from_addr, to_addr, subject, email_body, attachment_name, attackment_body) = emails(A, B, C, D, E, G, H, I):

We have A → BCEG, AH → I, and A→→D

NOTE: if X →Y then X→→Y

Start with R: (A, B, C, D, E, G, H, I) take out A → BCEG:

(A, B, C, E, G) where A is the primary key

(A, D, H, I) take out A→→ D

(A, D) where AD is the primary key

(A, H, I) where AH is the primary key

Final:

sent_email(email_id, send_date, from_addr, subject, email_body) where email_id is primary key

send_to(email_id, to_addr) where email_id, to_addr is primary key of relation and email_id foreign key to sent_email

add_attachment(email_id, attachment_name, attachment_body) where email_id, add_attachment is primary and email_id is foreign key to sent_email

sent_email is in 4NF because email_id is a superkey for sent_email which sprouts from the first dependency listed.

send_to is in 4NF because both are primary keys which makes a trivial multivalued dependency which sprouts from the second dependency listed.

add_attachment is in 4NF because email_id is a superkey for add_attachment which sprouts from the last dependency.