Alec Andrews

CS 121 Assignment 6

1.

a.) There are no non-trivial functional dependencies in this E-R diagram. Since it is a many to many relationship, there is no requirement or restriction between what is in a and what is in b for a given tuple in R(a, b).
b.) The non-trivial functional dependency is b→a, since there is only one value of a for each individual value of b.
c.) The non-trivial functional dependency is a→b, by the same logic as above. Each value of a maps to one particular value in b, so it logically implies it.
d.) This diagram has functional dependencies a→b and b→a since each value of a maps to one value of b, and the reverse is true as well.

2.

<u>Union Rule</u> – $If\ \alpha \to B\ holds, and\ \alpha \to \gamma\ holds, then\ \alpha \to B\gamma\ holds.$

1. Given: $\alpha \to B$
2. Given: $\alpha \to \gamma$
3. Apply the Augmentation Rule on 1 and γ: $\alpha\gamma \to B\gamma$
4. Apply the Augmentation Rule on 2 and α: $\alpha \to \alpha\gamma$
5. Apply the Transivity Rule on 4 and 3: $\alpha \to B\gamma$

<u>Decomposition Rule</u> – $If\ \alpha \to B\gamma\ holds, then\ \alpha \to B\ holds\ and\ \alpha \to \gamma\ holds.$

1. Given: $\alpha \to B\gamma$
2. Apply the Reflexivity Rule: $B\gamma \to B$
3. Apply the Reflexivity Rule: $B\gamma \to \gamma$
4. Apply the Transivity Rule on 1 and 2: $\alpha \to B$
5. Apply the Transivity Rule on 1 and 3: $\alpha \to \gamma$

<u>Pseudotransivity Rule</u> – $If\ \alpha \to B\ holds, and\ \gamma B \to \delta\ holds, then\ \alpha\gamma \to \delta\ holds.$

1. Given: $\alpha \to B$
2. Given: $\gamma B \to \delta$
3. Apply the Augmentation Rule on 1 and γ: $\gamma\alpha \to \gamma B$
4. Apply the Transivity Rule on 2 and 3: $\gamma\alpha \to \delta$

3.

R = (A, B, C, D, E)
F= {A→BC, CD→E, B→D, E→A}

a.)

E is a candidate key. We will compute $(E)^+$.

- Start with $\alpha^+ =$ E
- E→A causes $\alpha^+ =$ EA
- A→BC causes $\alpha^+ =$ EABC
- B→D causes $\alpha^+ =$ ABCDE (Thus, E is a superkey of R).

A is a candidate key. We will compute $(A)^+$.

- Start with $\alpha^+ =$ A
- A→BC causes $\alpha^+ =$ ABC
- B→D causes $\alpha^+ =$ ABCD
- CD→E causes $\alpha^+ =$ ABCDE (Thus, A is a superkey of R).

CD is a candidate key. We will compute $(CD)^+$.

- Start with $\alpha^+ =$ CD
- CD→E causes $\alpha^+ =$ CDE
- E→A causes $\alpha^+ =$ ACDE
- A→BC causes $\alpha^+ =$ ABCDE (Thus, CD is a superkey of R).

BC is a candidate key. We will compute $(BC)^+$.

- Start with $\alpha^+ =$ BC
- B→D causes $\alpha^+ =$ BCD
- CD→E causes $\alpha^+ =$ BCDE
- E→A causes $\alpha^+ =$ ABCDE (Thus, BC is a superkey of R).

b.) Functional dependencies:

- $All\ \alpha\gamma \rightarrow B\ such\ that\ \alpha\ is\ a\ candidate\ key\ outlined\ in\ part\ a.),\ \gamma \subseteq R/\alpha, and\ B \subseteq \{ABCDE\}$.
- $All\ trivial\ dependencies\ \alpha \rightarrow B\ such\ that\ \alpha \subseteq \{ABCDE\}\ and\ B \subseteq \alpha$
- B → D

4.

We know that by $A \twoheadrightarrow \mathrm{B}C$, we have for all pairs of tuples t_1 and t_2 in R such that $t_1[A] = t_2[A]$, there also exists tuples t_3 and t_4 such that:

- $t_1[A] = t_2[A] = t_3[A] = t_4[A]$
- $t_1[\mathrm{B}C] = t_3[\mathrm{B}C]$ and $t_2[\mathrm{B}C] = t_4[\mathrm{B}C]$
- $t_1[AD] = t_4[AD]$ and $t_2[AD] = t_3[AD]$

Assume that this relation R only has these types of tuples. Then, we do not have also that A→B or A→C. The reason for this is because the rule A→→BC only requires the existence of tuples t_3 and t_4 as defined above. For A→→B, there would be different requirements and this would require the existence of more tuples such that for every tuple t_1 and t_2 as defined above, there are also t_3 and t_4 such that:

- $t_1[A] = t_2[A] = t_3[A] = t_4[A]$
- $t_1[B] = t_3[B]$ and $t_2[B] = t_4[B]$
- $t_1[ADC] = t_4[ADC]$ and $t_2[ADC] = t_3[ADC]$

We do not necessarily have that $t_2[ADC] = t_3[ADC]$ with the tuples we have defined. Since we have restricted our relation to only include these tuples, A→→BC does not imply A→B and by symmetry, A→C.

5.

R(A, B, C, D, E, G)
F = {A→E, BC→D, C→A, AB→D, D→G, BC→E, D→E, BC→A}

a.)

1. First, we set F_C = F = {A→E, BC→D, C→A, AB→D, D→G, BC→E, D→E, BC→A}
2. Apply union rule on BC→D, BC→E, BC→A and on D→G and D→E to get F_C = {A→E, BC→ADE, C→A, AB→D, D→GE }
3. E is extraneous to BC→ADE. Consider the set F_C* = {A→E, BC→AD, C→A, AB→D, D→GE}. $(BC)^+$ = {ABCDE}, which includes E. Left with F_C = {A→E, BC→AD, C→A, AB→D, D→GE }
4. D is extraneous to BC→AD. Consider the set F_C* = {A→E, BC→A, C→A, AB→D, D→GE}. $(BC)^+$ = {ABCDE}, which includes D. Left with F_C = {A→E, BC→A, C→A, AB→D, D→GE}
5. B is extraneous in BC→A (Logically implied by C→A). Left with F_C = {A→E, C→A, C→A, AB→D, D→GE}
6. Clearly, C→A is extraneous because there is a duplicate. Left with F_C = {A→E, C→A, AB→D, D→GE}

b.)

Candidate key is BC. We will compute $(BC)^+$.

- Start with $\alpha^+ =$ BC
- C→A causes $\alpha^+ =$ ABC
- AB→D causes $\alpha^+ =$ ABCD
- D→GE causes $\alpha^+ =$ ABCDE, so BC is a superkey

Now, we will look at the attribute-set closures of BC's subsets.
We will compute $(B)^+$.

- Start with $\alpha^+ =$ B
- No dependencies can expand this closure, so $(B)^+$ = B.

We will compute $(C)^+$.

- $C \rightarrow A$ causes $\alpha^+ = CA$
- $A \rightarrow E$ causes $\alpha^+ = ACE$
- No more dependencies can expand this closure, so $(C)^+ = ACE$.

Thus, BC is a candidate key, or a minimal superkey.

c.)

- $A \rightarrow E$ holds on R and isn't a trivial dependency, but A isn't a superkey ($A^+ = \{AE\}$).
- We will decompose R into two schemas: $R_1 = (A, E)$ and $R_2 = (A, B, C, D, G)$.
- $C \rightarrow A$ holds on R and isn't a trivial dependency, but C isn't a superkey ($C^+ = \{ACE\}$).
- We will decompose R_2 into two schemas: $R_3 = (C, A)$ and $R_4 = (B, C, D, G)$.
- R_1 is in BCNF with respect to F_C, because the only functional dependency from F_C that holds on R_1 is $A \rightarrow E$, which is a nontrivial dependency and A is a superkey of R_1.
- R_3 is in BCNF with respect to F_C, because the only functional dependency from F_C that holds on R_3 is $C \rightarrow A$, which is a nontrivial dependency and C is a superkey of R_3.
- R_4 is in BCNF with respect to F_C because no functional dependencies from F_C hold on R_4.
- This BCNF decomposition ($R_1 = (A, E)$, $R_3 = (C, A)$, $R_4 = (B, C, D, G)$) is unable to preserve the dependencies $AB \rightarrow D$, $D \rightarrow GE$ from F_C.

d.)

- $D \rightarrow GE$ holds on R and isn't a trivial dependency, but D isn't a superkey of R ($D^+ = \{DGE\}$).
- We will decompose R into two schemas: $R_1 = (D, G, E)$ and $R_2 = (A, B, C, D)$
- $AB \rightarrow D$ holds on R_2 and isn't a trivial dependency, but AB isn't a superkey of R_2 ($(AB)^+ = \{ABD\}$).
- We will decompose R_2 into two schemas: $R_3 = (A, B, D)$ and $R_4 = (A, B, C)$.
- $C \rightarrow A$ holds on R_4 and isn't a trivial dependency, but C isn't a superkey of R_4 ($C^+ = \{AC\}$).
- We will decompose R_4 into two schemas: $R_5 = (C, A)$ and $R_6 = (B, C)$
- R_1 is in BCNF with respect to F_C, because the only functional dependency from F_C that holds on R_1 is $D \rightarrow GE$, which is a nontrivial dependency and D is a superkey of R_1.
- R_3 is in BCNF with respect to F_C, because the only functional dependency from F_C that holds on R_3 is $AB \rightarrow D$, which is a nontrivial dependency and AB is a superkey of R_3.
- R_5 is in BCNF with respect to F_C by the same logic as R_3 in part c.
- R_6 is in BCNF with respect to F_C, because there are no functional dependencies from F_C that hold on R_6.

e.)

We will use the 3NF Synthesis Algorithm on F. First recall that F_C = {A→E, C→A, AB→D, D→GE}.

- R_1 = (A, E), since no other schema contains (A ∪ E)
- R_2 = (C, A), since no other schema contains (C ∪ A)
- R_3 = (A, B, D), since no other schema contains (A ∪ B ∪ D)
- R_4 = (D, G, E), since no other schema contains (D ∪ G ∪ E)
- R_5 = (B, C), since no schema contains a candidate key for R.
- 3NF schema = (R_1, R_2, R_3, R_4, R_5)

6.

R = (course_id, section_id, dept, units, course_level, instructor_id, term, year, meet_time, room, num_students)

a.) We will refer to the functional dependencies that hold on R as F1, F2, F3 in accordance with the order they appear in the question.

- Candidate key: (course_id, section_id, term, year)
  - Start with $\alpha^+ =$ (course_id, section_id, term, year)
  - F1 causes $\alpha^+ =$ (course_id, section_id, term, year, dept, units, course_level)
  - F2 causes $\alpha^+ =$ R
- Candidate key: (room, meet_time, term, year)
  - Start with $\alpha^+ =$ (room, meet_time, term, year)
  - F3 causes $\alpha^+ =$ (room, meet_time, term, year, instructor_id, course_id, section_id)
  - F2 causes $\alpha^+ =$ (room, meet_time, term, year, instructor_id, course_id, section_id, num_students)
  - F1 causes $\alpha^+ =$ (room, meet_time, term, year, instructor_id, course_id, section_id, num_students, dept, units, course_level)

b.)

- First Canonical Cover
  - Start with F_C = {course_id→(dept, units, course_level), (course_id, section_id, term, year) → (meet_time, room, num_students, instructor_id), (room, meet_time, term, year) → (instructor_id, course_id, section_id)}
  - Instructor_id is extraneous in (course_id, section_id, term, year) → (meet_time, room, num_students, instructor_id). Consider the set F_C* = {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)}. (course_id, section_id, term, year)$^+$ = R, which includes instructor_id. Left with F_C = {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)}.
- Second Canonical Cover

- Start with F_C = {course_id→(dept, units, course_level), (course_id, section_id, term, year) → (meet_time, room, num_students, instructor_id), (room, meet_time, term, year) → (instructor_id, course_id, section_id)}
- Instructor_id is extraneous in (room, meet_time, term, year) → (instructor_id, course_id, section_id). Consider the set F_C* = {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)}. (room, meet_time, term, year)$^+$ = R, which includes instructor_id. Left with {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)}.

- I believe the most appropriate canonical cover is the second one. This one relates an instructor with a course and a section rather than a room, meeting time, and number of students. This is because the course id and section id are much less likely to change while the room, meeting time, and number of students can be variable if students drop class or class gets moved. When one is looking up a course, that is when they are looking for an instructor, rather than when they are looking up a room and a meeting time.

c.) The best normal form would be 3NF. This is because 3NF won't lose any of the functional dependencies required. BCNF can have lossy decompositions, so it becomes expensive to enforce some functional dependencies that are lost in the decomposition. 4NF and 5NF are not needed because there are no multivalued dependencies. Also, since this database is for a single college, it won't be that large, so the redundant information from 3NF as opposed to BCNF isn't as costly.

Now, we will use the 3NF Synthesis Algorithm to come up with a decomposition. Recall that F_C = {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)}.

- R_1 = (course_id, dept, units, course_level) since no other schemas contain (A U B) for the first functional dependency A→B in F_C.
- R_2 = (course_id, section_id, term, year, meet_time, room, num_students, instructor_id) since no other schemas contain (A U B) for the second functional dependency A→B in F_C.
- R_3 = (room, meet_time, term, year, course_id, section_id) since no other schemas contain (A U B) for the third functional dependency A→B in F_C.
- No other schemas are necessary because R_2 and R_3 contain candidate keys for R.

7.

We will construct schemas for these functional dependencies. We know that if A→B, then also A→→B.

- emails = (email_id, send_date, from_addr, subject, email_body). The functional dependency represented by this schema is email_id → (send_date, from_addr, subject, email_body). Thus, email_id →→ (send_date, from_addr, subject, email_body). However, (email_id) U (send_date, from_addr, subject, email_body) is a superkey for emails, so emails is in 4NF with respect to D (the set of functional and multivalued dependencies given to us by the problem), and this multivalued dependency is trivial per the index at the bottom of the set.
- recipients = (email_id, to_addr). The multivalued dependency represented by this schema is email_id →→to_addr. Since (email_id) U (to_addr) = R for recipients, this is a superkey for recipients, making this a trivial multivalued dependency by the index at the bottom of the set. Since the only multivalued dependency that holds on recipients is trivial, recipients is in 4NF.
- attachments = (email_id, attachment_name, attachment_body). The functional dependency represented by this schema is (email_id, attachment_name) → attachment_body. This is also a multivalued dependency, (email_id, attachment_name) →→ attachment_body. However, (email_id, attachment_name) U attachment_body is a superkey for attachments, making it a trivial multivalued dependency by the index at the bottom of set . Since all multivalued dependencies in D that hold on attachments are trivial, attachments is also in 4NF.