Homework 2

Fan Wu

fwz766

Q1.

a.

1. X -> Y, given;

2. Y -> Z, given;

3. X -> Z, using 1 and 2, transitivity rule;

4. X -> XY, augment 1 by X, augmentation rule;

5. XY -> YZ, augment 3 by Y, augmentation rule;

6. X -> YZ, using 4 and 5, transitivity rule.

So that a is true.

b.

1. X -> Y, given;

2. XZ -> YZ, augment 1 by Z, augmentation rule;

3. Z -> W, given;

4. YZ -> YW, augment 3 by Y, augmentation rule;

5. XZ -> YW, using 2 and 4, transitivity rule.

so that b is true.

c.

x	y	z
1	1	2
1	2	2
1	3	2

In this relation, XY -> Z, and Z -> X, but Z ->Y is not true.

Therefore, Z -> Y is not true.

(end)

Q2.

In order to examine the functional dependencies in F for violation of BCNF, calculate the attribute closure of FD:

(1) for ABH -> C

A -> ADE

B -> B

H -> H

AB -> ABDE

BH -> ABCDEFGH

AH -> ADEH

so the attribute closure of ABH is: ABH -> ABCDEFGH. It doesn't cause violation.

(2) for A -> DE

A -> ADE

so this FD causes a violation of BCNF.

(3) for BGH -> F

B -> B

G -> G

H -> H

BG -> BG

GH -> GH

BH -> ABCDEFGH

so the attribute closure of BGH is: BGH -> ABCDEFGH. It doesn't cause violation.

(4) for F -> ADH

A -> ADE

so the attribute closure of A is: F -> ADEFH. So this FD causes a violation of BCNF.

(5) for BH -> GE

BH -> ABCDEFGH. It doesn't cause violation.

We use the BCNF voilation to obtain a lossless BCNF schema:
R (A, B, C, D, E, F, G, H) use BCNF violation F -> ADH to decomposite R to
R1 (F, A, D, H) and R2(F, B, C, E, G)
and R1 and R2 are both lossless BCNF schema.

(end)

Q3.
Yes, I think the instance of the schema R always in BCNF.
In R, there are only 2 possible non-trivial FD: A -> B, B -> A.
No matter how many non-trivial FD an instance has, it will always be in BCNF because :
For A -> B, A's attribute closure is AB;
For B -> A, B's attribute closure is AB.
(end)

Q4.
The two canonical covers of F can be:
(1) A -> B, B -> C, C -> A;
(2) A -> C, B -> A, C -> B.

(1) For A -> BC, to test if attribute C is extraneous:
compute A's attribute closure using (A -> B, B -> AC, C -> AB)
1. A -> B;
2. B -> AC;
3. A -> AC, using 1 and 2;
4. A -> ABC, using 1 and 3;
4. A -> BC.
So C is extraneous.

For B -> AC, to test if attribute A is extraneous:

compute B's attribute closure using (A -> B, B -> C, C -> AB)

1. B -> C;

2. C -> AB;

3. B -> AB, using 1 and 2;

4. B -> ABC, using 1 and 3;

5. B -> AC.

So A is extraneous.

For C -> AB, to test if attribute B is extraneous:

compute C's attribute closure using (A -> B, B -> C, C -> A)

1. C -> A;

2. A -> B;

3. C -> B, using 1 and 2;

4. C -> AB, using 1 and 3.

So B is extraneous.

So that we can have one canonical cover of F:

A -> B, B -> C, C -> A

(2) For A -> BC, to test if attribute B is extraneous:

compute A's attribute closure using (A -> C, B -> AC, C -> AB)

1. A -> C;

2. C -> AB;

3. A -> AB, using 1 and 2;

4. A -> ABC, using 1 and 3;

4. A -> BC

So B is extraneous.

For B -> AC, to test if attribute C is extraneous:

compute B's attribute closure using (A -> C, B -> A, C -> AB):

1. B -> A;

2. A -> C;

3. B -> C, using 1 and 2;

4. B -> AC, using 1 and 3.

So C is extraneous.

For C -> B, to test if attribute A is extraneous:

compute C's attribute closure using (A -> C, B -> A, C -> B):

1. C -> B;
2. B -> A;
3. C -> A, using 1 and 2;
4. C -> AB, using 1 and 3.

So attribute A is extraneous.

So that we can have another canonical cover of F:

A -> C, B -> A, C -> B.

(end)

Q5.

(1) select b from R group by b having count(distinct c) > 1

This SQL query should return an empty set if the functional dependency B -> C holds in R.

(2)

```
create assertion BtoC check
(
    not exists
            (
                select b from R group by b having (distinct c ) > 1
            )
);
```

(end)

Q6.

a.

We calculate the attribute closure:

A -> ABCDE

B -> BD

C -> C

D -> D

E -> ABCDE

AB -> ABCDE

AC -> ABCDE

AD -> ABCDE

AE -> ABCDE

BC -> ABCDE

BD -> BD

BE -> ABCDE

CD -> ABCDE

CE -> ABCDE

DE -> ABCDE

So the candidate keys are:

A, E, BC, CD.

b.

According to central theorem of schema refinement:

A decomposition of R into R1 and R2 is lossless-join if

and only if at least one of the following dependencies is in F+:

$$R_1 \cap R_2 \rightarrow R_1; R_1 \cap R_2 \rightarrow R_2.$$

$$R_1 \cap R_2 = (A, B, C) \cap (A, D, E) = \{A\} \; and \; A \rightarrow ABCDE.$$

So that we have R_1, R_2 is a lossless-join decomposition of R.

(end)