CSCI 5333.1 DBMS Fall 2021

Suggested Solution for HW #8

[1] F = {B->A, AC->D, CD->F, F->E}

To prove BC-> E

Proof. For example:

[1] B->A (given)
[2] AC->D (given)
[3] BC->D (pseudo-transitivity rule on [1] and [2]).
[4] CD->F (given)
[5] BCC -> F (pseudo-transitivity rule on [4] and [3]).
[6] BC-> F (simplification of [5])
[7] F -> E (given)
[8] AE-> F (transitivity of [6] and [7]).

[2] Consider F = {A->B, BC->DE, AB->E, DE->C, AE->CD}

(a) A+=ABCDE, B+=B, C+= C, D+=D, E+=E
(b) The candidate key is A
(c) Prime: A, non-prime: BCDE
(d) {A->BC, BC->DE, DE->C}
(e) 2NF since BC->D or BC->E violates 3NF: D and E non-prime and BC is not a superkey. DE->C is also a violation.
(f) No, the best decomposition has a relation not in BCNF:
R1(A,B,C) {A->BC} in BCNF
R2(B,C,D,E) {BC->DE, DE->C} is in 3NF but not in BCNF.

[3] F = {CD->B, BC->D, D->A, F->DE, FDE->AC, B->F}

(a) A+=A, B+=ABCDEF, C+=C, D+=AD, E+=E, F+=ABCDEF
(b) The candidate keys are B, F and CD
(c) Prime: BCDF; non-prime: AE
(d) {B->F, D->A, CD->B, F->CDE}
(e) 1NF. D->A is a violation of 2NF.
(f) The decomposition into R1(B,C,D,E,F) {B->F, CD->B, F->CDE} and R2(A,D) {D->A} are lossless and FD preserving. R1 and R2 are in BCNF. Further decomposition of R1(B,C,D,E,F) is also acceptable but not preferrable.

[4]

(a) R(A,B,C,D) {A->C}: Highest NF: 1NF; CK: [1] ACD; A->C violates 2NF.

(b) R(A,B,C,D) {A->B, B->A, A->C, C->D, D->AB}: Highest NF: BCNF; CK: [1] A, [2] B, [3] C, [4] D.

(c) R(A,B,C,D,E) {AB->CD, C->ABE}: Highest NF: BCNF; CK: [1] AB, [2] C.

(d) R(A,B,C,D,E) {ABC->D, E->D}: Highest NF: 1NF; CK: [1]ABCE; Both FDs violate 2NF.

(e) R(A,B,C,D,E) {ABC->D, D->E}: Highest NF: 2NF; CK: [1] ABC; D->E violates 3NF

(f) R(A,B,C,D,E) {ABCE->D, D->BE}: Highest NF: 3NF; CK: [1] ABCE, [2] ACD; D->BE violates BCNF.

[5] R(A,B,C,D,E) {AB->C, A->D, BE->A, AD->CE} is decomposed into R1(A,B,C), R2(A,C,D,E) and R3(A,B,E).

The decomposition is lossless. We start by obtaining a canonical cover of F:

{AB->C, A->CDE, BE->A}

The decomposition of R into R1(A,B,C) and R'(A,B,C,D,E) is lossless since the common attributes are ABC, which is a superkey in R1. The further decomposition of R' into R2(A,C,D,E) and R3(A,B,E) is also lossless since the common attributes are AE and AE->CD in R2. Thus, the overall decomposition is lossless.

Using the chase matrix algorithm:

We use the canonical form: {AB-> C, A->CDE, BE->A}

Step 1. Create a table of 5 columns (number of columns and 4 rows (number of relations). Populate it with b[i,j].

Relation	**A**	**B**	**C**	**D**	**E**
R1	b[1,1]	b[1,2]	b[1,3]	b[1,4]	b[1,5]
R2	b[2,1]	b[2,2]	b[2,3]	b[2,4]	b[2,5]
R3	b[3,1]	b[3,2]	b[3,3]	b[3,4]	b[3,5]

Step 2. For each relation Ri, set all attribute Aj that appears in Ri from b[i,j] to a[j].

Relation	**A**	**B**	**C**	**D**	**E**
R1	*a[1]*	*a[2]*	*a[3]*	b[1,4]	b[1,5]
R2	*a[1]*	b[2,2]	*a[3]*	*a[4]*	*a[5]*
R3	*a[1]*	*a[2]*	b[3,3]	b[3,4]	*a[5]*

Step 3. While changes can be made with a FD X-> Y, with two rows in the table having the common X values in the following manner:

for every attribute W in Y:

- If one cell is an a and the other cell is an b, change the b to the a.
- If both cells are b's, change them to the same b.

Applying A->CDE

Relation	**A**	**B**	**C**	**D**	**E**
R1	a[1]	a[2]	a[3]	*a[4]*	*a[5]*
R2	a[1]	b[2,2]	a[3]	a[4]	a[5]
R3	a[1]	a[2]	*a[3]*	*a[4]*	a[5]

Since there are rows with all a's, the algorithm halts and declares that the decomposition is lossless.

[6]

[a] Maximum = 24 (e.g., when the two CKs are [1] A, and [2] B) and minimum = 3 (e.g., when the two CKs are [1] ABCD and [2] ABCE.

[b] Yes, it is also in BCNF.
Proof. Suppose we can find such a relation R that is in 3NF but not in BCNF. Since R is not in BCNF, there must be a non-trivial FD X->Y in a minimal cover of R such that it violates BCNF but not 3NF. In other words, X must not be a SK but Y is prime. Let K be the CK that contains Y (and thus making Y a prime attribute). Note that X and Y are disjoint. Thus, K ∪ X – Y -> K U X -> R. Thus, K ∪ X – Y is a SK without Y. It is now possible to find a second CK from it that is distinct from K, violating the assumption that there is only one CK.

[c] No. The following two example scenarios result in exactly five superkeys but they have different numbers of candidate keys.
[1] CK: [1] ABC, [2] ABDE => SK: [1] ABC, [2] ABCD, [3] ABCE, [4] ABCDE, [5] ABDE.
[2] CK: [1] ABCD, [2] ABCE, [3] ABDE, [4] ACDE => SK: [1] ABCD, [2] ABCE, [3] ABDE, [4] ACDE, [5] ABCDE