Examples and Exercises of BCNF

Question 1. Consider the following collection of relations and dependencies. For each relation, please (a) determine the candidate keys, and (b) if a relation is not in BCNF then decompose it into a collection of BCNF relations.

1. R1(A,C,B,D,E), A → B, C→ D

Answer: First compute the keys for R1. The attributes A, C, E do not appear on right hand side of any functional dependency therefore they must be part of a key. So we start from {A, C, E} and find out that this set can determine all features. So the key is {A, C, E}

We have dependencies A → B and C → D so the table is not BCNF. Applying the BCNF decomposition algorithm, the non-BCNF dependency is A → B, therefore create two relations (A, C, D, E) and (A, B). The first relation is still not in BCNF since we have a non-BCNF dependency C→ D. Therefore decompose further into (A, C, E) and (C, D). Now all relations are in BCNF and the final BCNF scheme is (A, C, E), (C, D), (A, B).

1. R2(A,B,F), AB→ F, B→ F

Answer: First compute keys for R2. Note that AB -> F is totally redundancy since we already have B-> F. A,B do not appear on right side of any dependency, so start by computing attribute set closure of {AB}. Since AB→ F, we have {AB}+ = {ABF} and therefore {AB} is the key. Since we have B → F, i.e., F is partially dependent on the key, the relation is not in BCNF. During BCNF decomposition, we have B → F as the non-BCNF relation therefore create new schema (A,B) (B,F). Both are in BCNF.

1. R5(A,B,C,D,E) with functional dependencies D → B, CE → A.

Answer: The key for R5 is {CDE} since we have {CDE}+ = {ABCDE}. R5 is not in BCNF and not in 3NF. Applying the BCNF decomposition, we pick D → B which is not BCNF and decompose into (D, B) and (ACDE). But (ACDE) is still not in BCNF since CE→ A is not BCNF. Therefore decompose into (D, B), (A,C,E) and (D,C,E) which is now in BCNF.

1. R6(A,B,C,D,E) with functional dependencies A→ E, BC→ A, DE→ B.

Answer: Since D and C do not appear on right hand side, all keys must contain C and D. The set {C,D} by itself is not a key since {C,D}+= {C,D}. However, all three element sets containing {C,D} are keys – i.e., {A,C,D} is a key, {B,C,D} is a key, and {C,D,E} is a key. So all attributes are prime attributes. Therefore the schema is in 3NF. However, the left hand side of the FDs is not a key therefore it is not in BCNF. One decomposition will start by removing A → E to form (A,E) and (A,B,C,D). But (A,B,C,D) is not in BCNF since BC is not the key. Therefore decompose further to get (A,E), (BCA) and (BCD). This is in BCNF since the only dependency in (BCD) is the trivial dependency BCD → BCD.

1. Consider a table (S, T, V, C, D,P) with the FDs as follows

S → T

V → SC

SD→ PV

Determine the key(s) for this relation (assuming all attributes are in one table to begin

with).

The set of attributes are: (S, T, V, C, D,P).

Examining the FD’s, we see that D does not appear on the right hand side of any dependency, therefore it must be part of any key. So the smallest set that can be a key is {D}. However, D+={D}. So we examine two element sets. Picking {S,D}, we get {S,D}+={S,D,T,P,V,C} and therefore it is a key. Similarly, {V,D}+={S,C,P,V,T,D} and therefore it is a key.

Sets {D,C}, {D,T} and {D,P} will not give us a key. Note that C, P and T do not appear in any of the left side of a FD. So adding them into the keys will not let you derive new properties.

The non-BCNF dependencies are S → T and V → SC

A lossless join decomposition into BCNF is ST, VC, VS, VDP, another is ST, VSC, VDP.

1. **R(A,B,C,D,E); D->B, CE->A**

DCE is a minimal key

R violates BCNF since D (in itself) is not a key

We choose D-> B and we can split the table into (**DB)**, (DCEA)

DCEA violates BCNF since CE is not a key.

We choose CE->A and split the table into (**CEA)**, (**CED)**

 **Answer:** **DB, CEA, CED**

1. **S(A,B,C,D,E); A->E, BC->A, DE->B**

DCE, DCB, DCA are minimal keys

S violates BCNF since A is not a key

We choose A->E and split the table into (**AE), (**ABCD)

ABCD violates BCNF since BC is not a key

We choose BC-> A and split the table into (**BCA), (BCD)**

 **Answer: AE, BCA, BCD**