**CSC343F Assignment 3**

**Part 2**

**Question 1:**

**a)**

Since a BCNF requires for all FDs in W, if X determines some Y, then Y is the set of all other attributes in V. We will determine the result based on a series of closure tests:

does not determine MNO, thus violates BCNF

does not determine MNOPQ, thus violates BCNF

does not determine NPQRST, thus violates BCNF

does not determine PQ, thus violates BCNF

Therefore, based on the closure tests above, all of the FDs in W violates BCNF property.

**b)**

We will perform the BCNF decomposition on the FD **.**

**Step 1:**

First replace V into two relations V1 and V2 such that:

**Step 2**: Perform projection of W onto V1 with initial **.**

→ For each attribute in the closure, if it exists in V1, add it to T1.

Note here I omitted the other subsets of V1 because those attributes are irrelevant to the LHS in W.

Then **.** By closure test we can confirm that V1 is in BCNF.

**Step 3:** Perform project of W onto V2 with initial

→ For each attribute in the closure, if it exists in V2, add it to T2.

Note here I omitted the other subsets of V2 because those attributes are irrelevant to the LHS in W.

Then **.** By closure test V2 is not in BCNF as T2 does not determine all attributes in V2.

**Step 4:** Perform decomposition on V2, with the following two new relations:

First project W onto V21 with initial :

Note here I omitted the other subsets of V2 because those attributes are irrelevant to the LHS in W.

Then  and by closure test we confirm that V21 is in BCNF.

Second, we project W onto V22 with initial

Note here I omitted the other subsets of V2 because those attributes are irrelevant to the LHS in W.

Then  as none of the closure attributes (except for LHS itself) are in V22, and then it follows vacuously V22 is in BCNF.

Then in result, we have decomposed V into BCNF relations . Note by performing the BCNF decomposition algorithm we are guaranteed with no redundancy and lossless join.

**Question 2:**

a)

To find the minimal basis, we first split each FDs with a single RHS and remove any redundant attributes/FDs:

Note that I have made a shortcut in directly stating the summaries because the closure tests results can be followed trivially by comparing the LHS and RHS of the FDs,

|  |  |
| --- | --- |
| AB→C | Removing A will result in B→C, which already exists in T, can remove from T |
| AB→D | Removing A will result in B→D, which already exists in T, can remove from T |
| ACDE→B | Since removing any attribute in LHS will no longer depends B, keep this FD |
| ACDE→F | Removing A will result in CDE→F, which already exists in T, can remove from T |
| B→A | No more attributes can be removed |
| B→C | No more attributes can be removed |
| B→D | No more attributes can be removed |
| CD→A | Removing any attribute in LHS will no longer depends A, keep this FD |
| CD→F | Removing any attribute in LHS will no longer depends F, keep this FD |
| CDE→F | Removing E will result in CD→F, which already exists in T, can remove from T |
| CDE→G | Removing any attribute in LHS will no longer depends G, keep this FD |
| EB→D | Removing E will result in B→D, which already exists in T, can remove from T |

After eliminating redundant attributes and FDs, we result in the following new set of FDs

**T’ = {ACDE→B, B→A, B→C, B→D, CD→A, CD→F, CDE→G}** which is a minimal basis of T.

b)

We first construct a summary table of all attributes in P for their appearance in the minimal basis T’ based on the following intuitions:

If some attributes do not appear in all FDs, it must be in a key; otherwise there is no other way to get it.

If some attributes never appear in LHS, no point to us to conduct closure tests.

If some attributes only appear in LHS, it will also need to be in a key because no FDs will depend to it.

|  |  |  |  |
| --- | --- | --- | --- |
| **Attributes** | **Appears on** | | **Conclusion** |
| **LHS** | **RHS** |
| A, B, C, D | **√** | **√** | Need closure check |
| F, G | **-** | **√** | Not in any key |
| E | **√** | **-** | Must be in every key |
| H | **-** | **-** | Must be in every key |

Then for the following minimal combination of attributes that needs closure check:

Does not functionally depend on all attributes, not a key

Functionally determines all attributes, is a key

Does not functionally depend on all attributes, not a key

Does not functionally depend on all attributes, not a key

In conclusion the only key found from this minimal basis is BEH.

c)

Given the minimal basis

**T’ = {ACDE→B, B→A, B→C, B→D, CD→A, CD→F, CDE→G}**

We first combine all FDs with same LHS (specifically combining all RHS for B→\* ) and construct a relation for each FD that includes all attributes in both LHS and RHS:

**R1(ABCDE), R2(ABCD), R3(ACDF), R4(CDEG)**

Since none of the above relations is a superkey as H is not included in any of them, we need to add another relation whose schema is the key: **R5(BEH).**

Thus, our final 3NF synthesis algorithm produces the collection of relations R1(ABCDE), R2(ABCD), R3(ACDF), R4(CDEG), R5(BEH) in 3rd NF.

d)

Since the above schema is produced through 3rd NF synthesis algorithm, it does not guarantee no redundancies because we are not decomposing the relation as opposed to BCNF decomposition (In BCNF, LHS of FDs are enforced to be superkey and thus we need to keep splitting the relation until all attributes will be determined, whereas in 3rd NF we build the set from scratch with non-superkey allowed in LHS, which implies redundancy may occur).