1) Examine the following relation/table. Identify all the functional dependencies in this relation (we will assume that this is the only possible relation instance!).

|  |  |  |
| --- | --- | --- |
| A table with unknown FDs | | |
| **A** | **B** | **C** |
| 1 | a | f |
| 2 | b | f |
| 1 | a | f |
| 3 | c | g |
| 2 | b | f |
| 2 | d | f |
| 2 | e | f |

2) First let's consider a BCNF decomposition. Assume that we have the following relation schema.

movie(movieName, whenMade, starName, age, address)

Further assume that we have the following functional dependencies.

movieName --> whenMade

starName --> age

starName --> address

starName, age --> age

movieName, starName --> whenMade, age

Try to answer the following.

1) Are there any trivial FDs?

**Solution**: Yes,   
    starName, age --> age   
is trivial.

2) Can we transitively determine any FDs?

**Solution**: No, not directly.

3) What is the closure of the lefthand side of each FD?

**Solution**:  
  movieName+ = {movieName, whenMade}  
  starName+ = {age, address, starName}  
  movieName,starName+ = {movieName, starName, age, address, whenMade}

4) Are there any BCNF violating FDs?

**Solution**:  Yes,

   movieName --> whenMade     violates BCNF since it is non-trivial and the lefthand side is not a key  
   starName --> age                     violates BCNF since it is non-trivial and the lefthand side is not a key  
   starName --> address               violates BCNF since it is non-trivial and the lefthand side is not a key  
   starName, age --> age              is trivial, so it does not violate BCNF  
   movieName, starName --> whenMade, age     The lefthand side is a key, so it does not violate BCNF.

5) What is the BCNF decomposition for this relation?

**Solution**:    
First let's decompose using  
   movieName --> whenMade  
We project out the attributes involved in the violating FD to obtain the relations:  
  Movie(movieName, whenMade)  
  Rest(movieName, starName, age, address)  
  
Now we work on the other FDs with the Rest relation.  
  starName --> age, address  
  
The left-hand side of this FD is still not a key, so we decompose Rest further to obtain.  
  Star(starName, age, address)  
  RestmovieName, starName)  
  
We have no more violating FDs so we are done! The decomposed schema, with the Rest relation suitably renamed, is   
  Movie(movieName, whenMade)  
  Star(starName, age, address)  
  StarsIn(movieName, starName)

3) For the same relation schema given above, assume instead that we have the following FDs.

movieName --> whenMade

starName --> movieName

starName, address, age --> movieName

Try to answer the following.

1) Are there any trivial FDs?

**Solution**: No.

2) Can we transitively determine any FDs?

**Solution**: Yes,   
   starName --> whenMade

3) What is the closure of the lefthand side of each FD?

**Solution**:   
  movieName+ = {movieName, whenMade}  
  starName+ = {starName, movieName, whenMade}  
  starName,address,age+ = {starName, address, age, movieName, whenMade}

4) Are there any BCNF violating FDs?

**Solution**: Yes,   
   movieName --> whenMade                      violates BCNF since it is non-trivial and the lefthand side is not a key  
   starName --> movieName                         violates BCNF since is is non-trivial and the lefthand side is not a key  
   starName, address, age --> movieName   does not violate BCNF since the lefthand side is a key

5) What is the BCNF decomposition for this relation?

**Solution**:    
First let's decompose using  
   movieName --> whenMade  
We project out the attributes involved in the violating FD to obtain the relations:  
  Movie(movieName, whenMade)  
  Rest(movieName, starName, age, address)  
  
Now we work on the other FD with the Rest relation.  
  starName --> movieName  
  
The left-hand side of this FD is still not a key, so we decompose Rest further to obtain.  
  StarsIn(starName, movieName)  
  Rest(starName, age, address)  
  
We have no more violating FDs so we are done! The decomposed schema, with the Rest relation suitably renamed, is   
  Movie(movieName, whenMade)  
  Star(starName, age, address)  
  StarsIn(starName, movieName)

4) A relation *R* has attributes *A, B, C, D, E, F, G, H, I, J*, and satisfies the following FDs:

*ABD -> E*  
*AB  -> G*  
*B   -> F*  
*C   -> J*  
*CJ  -> I*  
*G   -> H*

What are the candidate keys?

(This is Problem 9.15 from Introduction to Database Systems by C. J. Date, Sixth Edition)

**Solution**: The candidate keys must be a subset of *(A,B,C,D,G,J)* since these appear on the left hand side of the FDs above and determine all of the remaining attributes.

For these together to be a candidate key, the attributes must be **minimal** but they are not since we can remove *J* which is determined by *C* and also remove *G* which is determined by *AB* giving us a candidate key *(A, B, C, D)* which is a minimal set.

5) What is the difference between 3NF and BCNF?

**Solution** The 3NF assumes that all attributes not part of the candidate keys depend on the candidate keys but does not deal with dependencies within the keys. BCNF deals with such dependencies.

A relation R is said to be in BCNF if whenever X -> A holds in R, and A is not in X, then X is a candidate key for R.

It should be noted that most relations that are in 3NF are also in BCNF. Infrequently, a 3NF relation is not in BCNF and this happens only if

* the candidate keys in the relation are composite keys (that is, they are not single attributes),
* there is more than one candidate key in the relation, and
* the keys are not disjoint, that is, some attributes in the keys are common.

The BCNF differs from the 3NF only when there are more than one candidate keys and the keys are composite and overlapping.

6) Is every binary relation in BCNF? Prove it.

**Solution**: If both attributes form the primary key, the relation is in BCNF. If one of the attributes is a primary key, the other must be determined by it and thus the relation is in BCNF.

Another way of proving the above statement is that only two FDs are possible, either *A -> B* or *B -> A*. If no FDs exist, both attributes together are the key. Otherwise either *A* is a candidate key or *B* is or both are. None of these cases violates BCNF requirements.

7) Give an example of a relation that is in 3NF but not in BCNF.

**Solution** A relation that is in 3NF but not in BCNF is given below if we assume that sname and cname are unique and therefore the relation has a number of candidate keys viz. (sno, cno), (sno, cname), (sname, cno), (sname, cname).

enrol [sno, sname, cno, cname, date-enrolled]

8) A relation *NADDR* is defined with attributes *NAME* (unique), *STREET*, *CITY*, *STATE* and *POSTCODE*. For each postcode, there is only one city and state. Also, for any given street, city and state, there is only one postcode. Is this relation in BCNF? 3NF? Can you think of a better design?

(This is Problem 10.7 from Introduction to Database Systems by C. J. Date, Sixth Edition)

**Solution**: The only candidate key is *NAME*. The FDs are

*POSTCODE -> CITY,STATE*

*STREET,CITY,STATE -> POSTCODE*

*NAME -> STREET, CITY,STATE*

The relation is in 2NF since all non-key attributes (i.e. *STREET, CITY, STATE, POSTCODE*) are fully determined by the key (*NAME*). The dependence of *POSTCODE* on *NAME* is transitive but that is acceptable in 2NF. The relation however is not in 3NF because of the transitive dependence and would need to be decomposed perhaps in the following two relations:   
POSTCODE [POSTCODE, CITY, STATE]  
LOCATED [NAME, STREET,  POSTCODE]  
but such decomposition may not be desirable in practice since it may be necessary to keep all the address information together