Database Design – Practice Questions Solution

1. Given the relation schema R = (A, B, C, D, E) and the canonical cover of its set of functional dependencies

   F_c = { $A \rightarrow BC$
   $CD \rightarrow E$
   $B \rightarrow D$
   $E \rightarrow A$ }

   Compute a lossless join decomposition in Boyce-Codd Normal Form for R. Show your steps clearly to get full marks!

   Using the algorithm to decompose a relation to BCNF from figure 7.6 in text:

1. result = {(A, B, C, D, E)}; done = false;

2. Note that we are given the canonical cover F_c in the question. This means that we can avoid computing the closure of F and just use F_c and Armstrong's axioms to determine if a given functional dependency is in F^+.

3. (A, B, C, D, E) is not in BCNF because $B \rightarrow D$ is not a trivial dependency and it is not a superkey for (A, B, C, D, E):

$A \rightarrow BC$	given
$A \rightarrow B$, $A \rightarrow C$	decomposition
$B \rightarrow D$, so $A \rightarrow D$	given, transitive
$A \rightarrow CD$	union
$CD \rightarrow E$, so $A \rightarrow E$	transitive
$A \rightarrow ABCDE$	union of above steps
$E \rightarrow A$, so $E \rightarrow ABCDE$	given, transitive
$CD \rightarrow E$, so $CD \rightarrow ABCDE$	transitive
$B \rightarrow D$, so $BC \rightarrow CD$	augmentation
$BC \rightarrow ABCDE$	transitive

   Since BC is a candidate key, B cannot be a superkey. As soon as we find one functional dependency that does not meet the criteria for BCNF, the schema is not in BCNF.

4. $B \rightarrow D$ holds on (A, B, C, D, E),
   $B \rightarrow ABCDE$ is not in F^+ (*i.e.* can't be computed using Armstrong's Axioms from the canonical cover F_c) and
   $B \cap D$ = empty set, so:

   result = {(A, B, C, D, E) – (A, B, C, D, E)} ∪ {(A, B, C, D, E) – D} ∪ (B, D)
   result = {empty set} ∪ (A, B, C, E) ∪ (B, D)
   result = {(A, B, C, E), (B, D)}

5. We determine that (B, D) is in BCNF because the nontrivial functional dependency $B \rightarrow D$ is given, so B is a superkey for schema (B, D).

6. We determine that (A, B, C, E) is in BCNF because for the nontrivial functional dependencies given, $A \rightarrow BC$ and $E \rightarrow A$, both A and E are superkeys for the schema (A, B, C, E), since $A \rightarrow ABCDE$ and $E \rightarrow ABCDE$ from step 3.

2. Suppose you are given the following functional dependencies:

   fd1: name → address, gender
   fd2: address → rank
   fd3: rank, gender → salary

a) Give a primary key of the relation r(name, address, gender, rank, salary). Prove your answer formally using Armstrong's Axioms.

   name is the primary key. To prove this, first prove that name is a superkey, then prove it is a candidate key and finally a primary key.

   To prove that name is a superkey:

1.	name → address	decomposition on fd1
2.	name → gender	decomposition on fd1
3.	name → rank	transitivity on 1. and fd2
4.	name, gender → salary	3. & pseudo-transitivity on fd2
5.	name → name, gender	augmentation of 2.
6.	name → salary	transitivity on 4. and 5.
7.	name → name, address, rank, gender, salary	union of 1., 2., 3., 6. & augmentation w. name

From the given FD's, it is not possible to have an FD that only has 'name' on its right-hand side, therefore, the attribute name must be part of any superkey. As the definition of a candidate key is a minimal superkey and {name} is a superkey, and the empty set is not a superkey, 'name' must be a candidate key.

Since name is a candidate key and it is the only candidate key, it is also the primary key.

b) (5 marks) Normalize the relation r(name, address, gender, rank, salary) to 3rd normal form, ensuring that the resulting relations are dependency-preserving and lossless-join decompositions. Specify the primary keys in the normalized relations by underlining them.

Use the algorithm on page 230 of the text to work through the decomposition. Note that the fd's given form a canonical cover F_c, so one possible decomposition is as follows:

r_1(name, address, gender)
r_2(address, rank)
r_3(rank, gender, salary)
(or r_3'(address, gender, salary) is also possible)

Mention MUST be made about why the decomposition is a lossless join and dependency preserving. Note the text mentions that the decomposition results in a lossless join, dependency-preserving decomposition, so this would be sufficient.

Otherwise, a decomposition is a lossless join if, for all relations r on schema R that are legal under the given set of functional dependency constraints,

$r = \Pi_{R1}(r) \bowtie \Pi_{R2}(r) \bowtie \Pi_{R3}(r)$

If we were required to prove this, note that the universal relation r is first decomposed into two smaller relations r_1 and r_2. The relation r_2 is then further decomposed to r_{21} and r_{22}. If we can show that r_{21} and r_{22} is a lossless-join, we can recover the relation r_2. Then if we can show that r_1 and r_2 also form a lossless-join, then we can recover the universal relation r and the entire decomposition is a lossless join.

To show that two relations r_1 and r_2 form a lossless join, we must show one of the following:

$r_1 \cap r_2 \rightarrow r_1$
$r_1 \cap r_2 \rightarrow r_2$

In our case, the intersection of r_1 and r_2 is address, so we must determine if either:
address $\rightarrow$ name, address, gender or
address $\rightarrow$ address, rank holds.

We can get the second FD by augmenting the given FD address $\rightarrow$ rank with address, so this is a lossless-join and we can recover (name, address, gender, rank).

The intersection of (name, address, gender, rank) and (rank, gender, salary) is (rank, gender). We must determine if either
rank, gender $\rightarrow$ name, address, rank, gender or
rank, gender $\rightarrow$ rank, gender, salary holds.

We can get the second FD by augmenting the given FD
rank, gender $\rightarrow$ salary with rank and gender, so this decomposition is also a lossless-join and therefore the entire decomposition is a lossless-join.

A decomposition is dependency preserving if $F - F' = \{\varnothing\}$, or $\{F_1 \cup F_2 \cup F_3\}^+ = F^+$. This can be proven by calculating the closures as indicated. An easier way, based on page 223 of the text, is to show that all FD's in F_c can be tested in a single relation in the decomposition. So,

name $\rightarrow$ address, gender can be tested in r_1

address, $\rightarrow$ rank can be tested in r_2

rank, gender $\rightarrow$ salary can be tested in r_3