WORKSHEET 6

Student Name: Vivek Kumar **UID:** 21BCS8129
DOMAIN CAMP: 16-01-2023 to 28-01-2023 **Section/Group:** DWWC-77
Subject Name: Database Management System

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

Fc = { A → BC

CD → E

B → D

E → A }

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

Ans. 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 Fc and

Armstrong's axioms to determine if a given functional dependency is in F +

(A, B, C, D, E) is not in BCNF because B → D is not a trivial dependency and it is not a super key for (A, B, C, D, E):

A → BC given

A → B, A → C decomposition

B → D, so A → D given, transitive

A → CD union

CD → E, so A → E transitive

A → ABCDE union of above steps

E → A, so E → ABCDE given, transitive

CD → E, so CD → ABCDE transitive

B → D, so BC → CD augmentation

BC → ABCDE transitive

Since BC is a candidate key, B cannot be a super key. As soon as we find one functional

dependency that does not meet the criteria for BCNF, the schema is not in BCNF.

Ques 2. ***Given the relation schema R = (A, B, C, D, E) and functional dependencies***

Fc = { A → BC

CD → E

B → D

E → A } determine that (B, D) is in BCNF.

Ans. We determine that (B, D) is in BCNF because the nontrivial functional dependency B → D is given, so B is a superkey for schema (B, D).

Ques 3. ***Suppose you are given the following functional dependencies:***
fd1: name → address, gender fd2: address → rank fd3: rank, gender
→ salary
Give a primary key of the relation r(name, address, gender, rank, salary). Prove your answer formally using Armstrong's Axioms

Ans. 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 super key. As the definition of a candidate key is a minimal super key and {name} is a super key, and the empty set is not a super key, '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.

Ques 4. ***Normalize the relation r(name, address, gender, rank, salary) to 3rd normal form, ensuring that the resulting relations are dependency-preserving and lossless-join decomposition. Specify the primary keys in the normalized relations by underlining them.***

r 1(name, address, gender) r

2(address, rank) r 3(rank,

gender, salary)

(or r 3 '(address, gender, salary) is also possible)

Ans. 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 = Π R1 (r) ⋈ Π R2 (r) ⋈ Π 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.

***Ques 5.* show that two relations r 1 and r 2 form a lossless join**

Ans. To show that two relations r 1 and r 2 form a lossless join, we must show one

of the following:

r 1 ∩ r 2 → r 1 r

1 ∩ r 2 → r2

In our case, the intersection of r 1 and r 2 is address, so we must determine

if either:

address → name, address, gender or address

→ address, rank holds.

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

***Ques 6.* Given the relation schema R = (A, B, C, D, E) and the set of functional dependencies:**
F = { E → AB
BC → D
D → E
AB → BC
BC → E }
Compute the canonical cover Fc. Show your steps clearly!

Ans. Use the union rule to replace

BC → D

BC → E

With

BC → DE

The left side of each functional dependency in F is now unique, so there are no more functional dependencies to replace using the union rule.

The attribute B in BC of AB → BC is extraneous because from the algorithm

from page 209 of the text, AB → C logically implies AB →

BC, so replace AB → BC with AB → C.

The attribute E in BC → DE is extraneous because E ∈ DE and

(F - {BC → DE}) U {BC → (DE - E)} logically implies F. This

is true because BC → D is one of the given functional

dependencies, so replace BC → DE with BC → D.

There are no more extraneous attributes, since none of the attributes

on the left side or right side of any remaining functional dependency is

extraneous. Therefore, the canonical cover is:

F c = { E → AB

BC → D

D → E

AB → C }

Ques 7. ***Consider the following functional dependencies over the attribute set ABCDEFGH:***

A -> E, BE -> D, AD -> BE, BDH -> E, AC -> E,

F-> A, E -> B, D -> H, BG -> F, CD -> A

Find a minimal cover for this set of functional dependencies.

Ans. First, we change every functional dependency (FD) into the form X->A (with only one attribute on the right-hand side):

A->E

BE->D

AD->B

AD->E

BDH->E

AC-> E

F->A E->B

D->H

BG->F

CD->A

Now we check the new set of FDs to see if any of them is redundant (i.e. they can be inferred from the others). An FD X>A is redundant if the closure of X contains A after removing the FD X->A.

- (A) +K-{A->E} ={A,B} so the FD A->E is NOT redundant.
- (E) +K-{E->D} ={E,B} so the FD E->D is NOT redundant.
- (A) +K-{A->B} ={A,E,B} so the FD A->B is redundant and will be removed from K.
- (BD) +K-{BD->E} ={B,D,H} so the FD BD->E is NOT redundant.
- (F) +K-{F->A} ={F} so the FD F->A is NOT redundant.
- (E) +K-{E->B} ={E,D,H} so the FD E->B is NOT redundant.
- (D) +K-{D->H} ={D} so the FD D->H is NOT redundant.
- (BG) +K-{BG->F} ={B,G} so the FD BG->F is NOT redundant.
- (CD) +K-{CD->A} ={C,D,H} so the FD CD->A is NOT redundant.

So the minimal cover is as follows:

A->E

E->D

BD->E

F->A

E->B

D->H

BG->F

CD->A

Ques 8. ***Consider the following functional dependencies over the attribute set ABCDEFGH:***

A -> E, BE -> D, AD -> BE, BDH -> E, AC -> E,

F-> A, E -> B, D -> H, BG -> F, CD -> A

Decompose the relation ABCDEFGH into a lossless 3NF schema.

Ans. A possible 3NF decomposition:

R1 (A,E) A->E

R2 (B,D,E) E->D, BD->E, E->B

R3 (A,F) F->A

R4 (D,H) D->H

R5 (B,F,G) BG->F

R6 (A,C,D) CD->A

R7 (B,C,G) no functional dependency

Ques 9. ***Given the attribute set R = ABCDEFGH and the functional dependency set F = {BC → GH, AD → E, A → H, E → BCF, G → H}, decompose R into BCNF by decomposing in the order of the given functional dependencies.***

Ans. ADE, BCEF, GH, BCG

BC -> GH violates BCNF, decompose.

Relations: ABCDEF BCGH

AD -> E does not violate BCNF because AD is a super key, skip.

A -> H No relation contains AH, skip.

E -> BCF violates BCNF, decompose.

Relations: ADE EBCF BCGH

G -> H violates BCNF, decompose.

ADE EBCF BCG GH

<u>Ques 10.</u> Given the attribute set R = ABCDEF and the functional dependency set

F = {B → D, E → F, D → E, D → B, F → BD}.

a. Is the decomposition ABDE, BCDF lossless?

<u>Ans.</u> No, it is lossy.

ABDE ∩ BCDF = BD

BD -> BDEF, which is not equivalent to either ABDE or BCDF

b. If not, what functional dependency could you add to make it lossless?

<u>Ans.</u> Any or all of BDEF -> Any or all of AC

For example some valid answers were: B -> C, B -> A, BEFD -> AC, etc.

To be lossless, we want BD -> ABDE or BD -> BCDF. We know that BD -> BDEF, so either

want BD -> ABDEF or BD -> CBDEF (or both!). This will be satisfied if

BDEF -> A and/or C.

<u>Ques 11.</u> In below table, find this relation is in 1st Normal form or Not. If below table is not in 1st NF then convert it into 1NF?

Table 1: Student

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY
1	RAM	9716271721, 9871717178	HARYANA	INDIA
2	RAM	9898297281	PUNJAB	INDIA
3	SURESH		PUNJAB	INDIA

Ans. STUDENT table 1 is not in 1NF because of multi-valued attribute STUD_PHONE. To convert it into 1st NF, we need to apply decomposition. Now, after applying decomposition table is in 1st NF.

Table 2: Student

STUD_NO	STUD_NAME	STUD_PHONE	STUD_STATE	STUD_COUNTRY
1	RAM	9716271721	HARYANA	
1	RAM	9871717178	HARYANA	INDIA
2	RAM	9898297281	PUNJAB	INDIA
3	SURESH		PUNJAB	INDIA

Ques 12. In below table, find this relation is in 1st Normal form or Not. If below table is not in 1st NF then convert it into 1NF?

ID	Name	Courses
1	**A**	**c1, c2**
2	**E**	**c3**
3	**M**	**c2, c3**

Ans. In above table, for ID1, courses have multivalued dependency which violates the rules of 1NF. Hence, above table is not in 1NF. To convert this table into 1NF we need to apply decomposition.

ID	Name	Courses
1	A	c1
1	A	c2
2	E	c3
3	M	c2
3	M	c3

Ques 13. Consider below table and find it is in 2nd NF or not? If it is not in 2nd NF then convert it into 2 NF.

STUD_NO	*COURSE_NO*	*COURSE_FEE*

1	***C1***	***1000***
2	***C2***	***1500***
1	***C4***	***2000***
4	***C3***	***1000***
4	***C1***	***1000***
2	***C5***	***2000***

Ans. {Note that, there are many courses having the same course fee.}

Here,
COURSE_FEE cannot alone decide the value of COURSE_NO or STUD_NO;
COURSE_FEE together with STUD_NO cannot decide the value of COURSE_NO;
COURSE_FEE together with COURSE_NO cannot decide the value of STUD_NO; Hence,
COURSE_FEE would be a non-prime attribute, as it does not belong to the one only candidate key {STUD_NO, COURSE_NO} ;
But, COURSE_NO -> COURSE_FEE , i.e., COURSE_FEE is dependent on COURSE_NO, which is a proper subset of the candidate key. Non-prime attribute COURSE_FEE is dependent on a proper subset of the candidate key, which is a partial dependency and so this relation is not in 2NF. To convert the above relation to 2NF, we need to split the table into two tables such as :

Table 1: STUD_NO, COURSE_NO

STUD_NO	COURSE_NO
1	C1
2	C2
1	C4
4	C3
4	C1
2	C5

Table 2: COURSE_NO, COURSE_FEE

COURSE_NO	COURSE_FEE
C1	1000
C2	1500
C3	1000
C4	2000
C5	2000

NOTE: 2NF tries to reduce the redundant data getting stored in memory. For instance, if there are 100 students taking C1 course, we don't need to store its Fee as 1000 for all the 100 records, instead once we can store it in the second table as the course fee for C1 is 1000.

<u>Ques 14.</u> Consider a Relation R has ten attributes ABCDEFGHIJ. Fields of R contain only atomic values. R = {AB→C, AD->GH BD-> EF A-> I H->J}. Find the following relation is in which normal form. If it is in 1^st^ NF then convert it into 2^nd^ NF.

<u>Ans.</u> Holds partial dependency.

As, Candidate Key **is ABD**$^+$ and the relation AB→ C which holds partial dependency, hence it holds 1st NF. Now convert it into 2nd NF.

Relation R1: AB-> C

R2: A->I;

R3: AD-> GH & H-> J

R4: BD->EF

R5: (ABD).

Now it is in 2nd NF.

Ques 15. Consider the relation Email (Sender, Receiver, Date, Time, Subject) with the functional dependencies.

FD1: {Receiver, Date} ->Time

FD2: {Sender, Date} ->{Receiver, Subject}

FD3: Receiver ->Subject

What is the highest normal form for the given relation?

Ans. Relation schema is in 2nd NF.

Ques 16. Consider the following relational schema:

Suppliers(sid:integer, sname:string, city:string, street:string)

Parts(pid:integer, pname:string, color:string)

Catalog(sid:integer, pid:integer, cost:real)

Assume that, in the suppliers relation above, each supplier and each street within a city has a unique name, and (sname, city) forms a candidate key. No other functional dependencies are implied other than those implied by primary and candidate keys. Find normal form?

Ans. A relation is in BCNF if for every one of its dependencies X → Y, at least one of the following conditions hold:

X → Y is a trivial functional dependency (Y   X)

X is a super key for schema R

Since (sname, city) forms a candidate key, there is no non-trivial dependency X → Y where X is not a super key. Hence, the above said schema is in BCNF.

Ques 17. Consider the following relational schemes for a library database: Book (Title, Author, Catalog_no, Publisher, Year, Price) Collection (Title, Author, Catalog_no) with in the following functional dependencies:

Title Author --> Catalog_no

Catalog_no --> Title, Author, Publisher, Year

Publisher Title Year --> Price

Assume {Author, Title} is the key for both schemes. Find the normal form of Book and collection.

Ans. Book (Title, Author, Catalog_no, Publisher, Year, Price)

Collection (Title, Author, Catalog_no)

within the following functional dependencies:

- Title, Author --> Catalog_no
- Catalog_no --> Title, Author, Publisher, Year
- Publisher, Title, Year --> Price

Assume {Author, Title} is the key for both schemes

- The table "Collection" is in BCNF as there is only one functional dependency "Title Author –> Catalog_no" and {Author, Title} is key for collection.
- Book is not in BCNF because Catalog_no is not a key and there is a functional dependency "Catalog_no –> Title Author Publisher Year".
- Book is not in 3NF because non-prime attributes (Publisher Year) are transitively dependent on key [Title, Author].
- Book is in 2NF because every non-prime attribute of the table is either dependent on the whole of a candidate key [Title, Author], or on another nonprime attribute.
  In table book, candidate keys are {Title, Author} and {Catalog_no}. In table Book, non-prime attributes (attributes that do not occur in any candidate key) are Publisher, Year and Prince

Hence, Book is in 2NF and Collection is in 3NF.

Ques 18. Let consider Schema *R* = ABCD, *F* = {A -> B, B -> C, AC -> D}. Find Normal Form and convert it into 3rd NF.

Ans. Candidate key: {A}
Step 1: nothing
Step 2: Minimal *F'* = {A -> B, B -> C, A -> D} Step
3: create relations:
For A->B, create a relation R1(A,B)
For B->C, create a relation R2(B,C)
For A->D, create a relation R3(A,D)
Step 4: do nothing

Step 5: do nothing, since candidate key A is in A->B

Result: R1(A,B), R2(B,C), R3(A,D)

<u>Ques 19.</u> Consider a relation R (A,B,C,D,E,F,G,H) with functional dependencies F = {A->B, ABCD->E, EF->G, EF->H, ACDF->EG}. Find Normal Form and convert it into 3rd NF.

<u>Ans.</u> After step 1: F1 = {A->B, ABCD->E, **EF -> GH**, ACDF -> EG} In step 2:

Remove attribute **B** from LHS of A**B**CD->E
Remove **E** from RHS of ACDF->EG
Remove ACDF ->G

Result: F2 = {A-> B, ACD -> E, EF -> GH} Candidate key: {ACDF}
Step 3: create relations:

A->B: create a relation R1(A, B)
ACD->E: create a relation R2(A, C, D, E)
EF->GH: create a relation R3(E, F, G, H)

Step 4: do nothing
Step 5: ACDF is a candidate key, so create a relation R4(A,C,D,F) Result:
R1(A,B), R2(A,C,D,E), R3(E,F,G,H), R4(A,C,D,F)

<u>Ques 20.</u>There is a company wherein employees work in more than one department. They store the data like this:

emp_id	emp_nationalit	emp_dept	dept_typ	dept_no_of_em y e p
1001	**Austrian**	**Production and planning**	**D001**	**200**
1001	**Austrian**	**Stores**	**D001**	**250**
1002	**American**	**design and technical support**	**D134**	**100**
1002	**American**	**Purchasing department**	**D134**	**600**

Functional dependencies in the table above:
emp_id -> emp_nationality emp_dept -> {dept_type, dept_no_of_emp}

Candidate key: {emp_id, emp_dept}

Find Highest NF and Convert it into BCNF.

Ans. The table is not in BCNF as neither emp_id nor emp_dept alone are keys. To make the table comply with BCNF we can break the table in three tables like this:

emp_nationality table:

emp_id	emp_nationality
1001	Austrian
1002	American

emp_dept table:

emp_dept	dept_type	dept_no_of_emp
Production and planning	D001	200
stores	D001	250
design and technical support	D134	100
Purchasing department	D134	600

emp_dept_mapping table:

emp_id	emp_dept
1001	Production and planning
1001	Stores
1002	design and technical support
1003	Purchasing department

Functional dependencies: emp_id -> emp_nationality emp_dept -> {dept_type, dept_no_of_emp}

Candidate keys:
For first table: emp_id
For second table: emp_dept
For third table: {emp_id, emp_dept}

This is now in BCNF as in both the functional dependencies left side part is a key.