MATB41H3 - Assignment 6

Ronald John Tutor
1009272001

Andrii Matsevytyi
1010798433

December 3, 2023

Problem 1

Consider a relation R with attributes ABCDEFG and with functional dependencies S:

$$S = \{AC \to D, \quad BG \to E, \quad D \to CFG, \quad DG \to B, \quad G \to F\}$$

(a) State which of the given FDs violate BCNF.

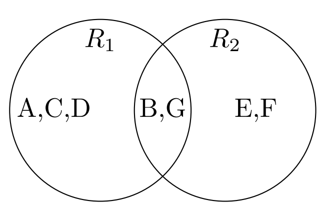
To check the FDs that violate BCNF, it is enough for us to do the closure on each FD and ensure that the left-hand side is a superkey everytime.

$$AC^+ = ACDFGBE \qquad \checkmark \tag{1}$$
$$BG^+ = BGEF \qquad \times \tag{2}$$
$$D^+ = DCFGBE \qquad \times \tag{3}$$
$$DG^+ = DCFGBE \qquad \times \tag{4}$$
$$G^+ = GF \qquad \times \tag{5}$$

Notice that (1) has a checkmark because it is does not violate BCNF since doing the cloure on AC yields all the attributes in R. However, $(2), (3), (4)$, and (5) are marked with $\times$ since it does not yield all the attributes in R.

(b) Employ the BCNF decomposition algorithm to obtain a lossless and redundancy-preventing decomposition of relation R into a collection of relations that are in BCNF. Make sure it is clear which relations are in the final decomposition, and don't forget to project the dependencies onto each relation in that final decomposition. Because there are choice points in the algorithm, there may be more than one correct answer. List the final relations in alphabetical order (order the attributes alphabetically within a relation, and order the relations alphabetically).

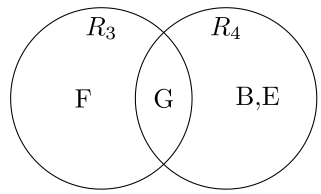
We will employ the BCNF decomposition algorithm on the table. Specifically, we begin with the FD $BG \to E$ since it violated BCNF first.



Consider $\boldsymbol{R_2 : BEFG}$, we get the table

Left Side	**Closure**	**FD**
B	B^+=B	NA
E	E^+=E	NA
F	F^+=F	NA
G	G^+=GF	$G \to F$

Then, we have to split $R2$ again using $G \to F$.



Consider $\boldsymbol{R_3 : FG}$, we know that this must NOT violate BCNF since it is a table of two attributes.
We also get the table,

Left Side	**Closure**	**FD**
F	F^+=F	NA
G	G^+=GF	$G \to F$

Consider $\boldsymbol{R_4 : BEG}$, we get the table

Left Side	**Closure**	**FD**
B	B^+=B	NA
E	E^+=E	NA
G	G^+=GF	NA
BE	BE^+=BE	NA
BG	BG^+=BGE	$BG \to E$
EG	EG^+=EGF	NA
~~BGE~~		

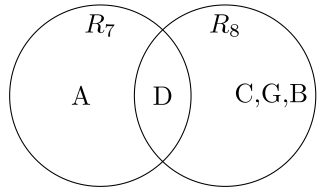
Note that the last row is omitted since we already know that BG is a key so BGE must be a superkey. We also know that know that R_4 does not violate BCNF.

We finished decomposing R_2. Now we have to go back to R_1.

Consider $\boldsymbol{R_1 : ABCDG}$, we get the table

Left Side	**Closure**	**FD**
A	A^+=A	NA
B	B^+=B	NA
C	C^+=C	NA
D	D^+=DCFGBE	$D \to BCG$

Then, we have to split R_1 again using $D \to BCG$ since it violates BCNF considering it does not get A.



Consider $\boldsymbol{R_7 : AD}$, we know that this must NOT violate BCNF since it is a table of two attributes. We also get the table,

Left Side	**Closure**	**FD**
A	A^+=A	NA
D	D^+=DCFGB	NA

Consider $\boldsymbol{R_8 : BCDG}$, we get the table

Left Side	**Closure**	**FD**
B	B^+=B	NA
C	C^+=C	NA
D	D^+=DCFGB	$D \rightarrow BCG$
G	G^+=GF	NA
BC	BC^+=BC	NA
~~BD~~		
BG	BG^+=BGE	NA
~~CD~~		
CG	CG^+=CG	NA
~~DG~~		
~~BCD~~		
BCG	BCG^+=BCGE	NA
~~BDG~~		
~~CDG~~		
~~BCDG~~		

After the row where left side is D, all other rows that contains D is ommitted from consideration since it is guaranteed that it will be a superkey since D itself is a key. Then, from the table above, we see that every possible combination was considered and no FD violates BCNF. We can conclude that R_8 does not violate BCNF.

$\therefore$ Our final relations must be

$$R_7 : AD \text{ with no functional dependencies}$$
$$R_8 : BCDG \text{ with functional dependency } D \rightarrow BCG$$
$$R_4 : BEG \text{ with functional dependency} BG \rightarrow E$$
$$R_3 : FG \text{ with functional dependency } G \rightarrow F$$

(c) Does your schema preserve dependencies? Explain how you know that it does or does not.

No it does not because we lost the following functional dependencies:

$AC \rightarrow D$ since,

$$AC^+ = AC$$

All other functional dependencies are maintained so we have: $D \rightarrow CFG$ since,

$$D \rightarrow \text{DB}\mathbf{CGF}$$

$DG \rightarrow B$ since,

$$DG^{+} = \text{DG}\mathbf{B}\text{CF}$$

$G \rightarrow F$ since it still remains as a functional dependency on the smaller relation FG.

$BG \rightarrow E$ since, it still remains as a functional dependency on the smaller table BEG

(d) Use the Chase Test to show that your schema is a lossless-join decomposition. (This us guaranteed by the BCNF algorithm, but it's a good exercise.)

We will employ the Chase Test. Suppose our original relation R has a row with values ***abcdefg***. From (b), we know we have 4 relations in our schema. Then, projecting R into those 4 relations and joining them back up we get:

A	**B**	**C**	**D**	**E**	**F**	**G**
a			d			
	b	c	d			g
	b			e		g
					f	g

Note that the empty cells may or may not be empty, but it is an arbitrary value that is not written for brevity and cleanliness.

The first row on the table above comes from R_7, second row from R_8, third from R_4 and last row from R_3.

From the set of original FD's, S, using the FD $D \rightarrow CFG$ and the data in the second row, we get

A	**B**	**C**	**D**	**E**	**F**	**G**
a		**c**	d			**g**
	b	c	d			g
	b			e		g
					f	g

We also know from $DG \rightarrow B$ and the second row that

A	**B**	**C**	**D**	**E**	**F**	**G**
a	**b**	c	d			g
	b	c	d			g
	b			e		g
					f	g

$G \rightarrow F$ means we have:

A	**B**	**C**	**D**	**E**	**F**	**G**
a	b	c	d		**f**	g
	b	c	d		**f**	g
	b			e	**f**	g
					f	g

Finally, using $BG \rightarrow E$ and the data in the third row, we get:

A	**B**	**C**	**D**	**E**	**F**	**G**
a	b	c	d	**e**	f	g
	b	c	d		f	g
	b			e	f	g
					f	g

Since we get the row with values ***abcdefg*** on the first row, then by the Chase Test, we have shown that our new schema is a lossless-join decomposition.

Assignment 3

Task 2

a)

We have set A = {LMNOPQRS}.
And FD's set B = { N→M, NO→LR, NQR→MP, P→R, Q→NO }

Initial closures for each FD in B

N^+ → MN
NO^+ → LMNOR
NQR^+ → LMNOPQR
P^+ → PR
Q^+ → LMNOPQR

FD's in standart form, removing redundancy:
N → M
NO → L
NO → R
NQR → M (because N → M)
NQR → P (because Q→NO and NO→R)
P → R
Q → N
Q → O

Closures after updation:
N^+ → MN
NO^+ → LMNOR
NQR^+ → LMNOPQR
P^+ → PR
Q^+ → LMNOPQR

As we seem nothing changed, so minimal basis holds.

Minimal basis:
N → M
NO → L
NO → R
P → R
Q → N
Q → O
Q → P

b)

Searching for key in :
N → M (because Q→N and N→M) – now Q → M
NO → L (because Q→NO and NO→R) – now Q → L
NO → R (because Q→NO and NO→R) – now Q → R
NQR → M (because N → M)
NQR → P (because Q→NO and NO→R)
P → R (because Q→NO and NO→R and Q→P and P→R)
Q → N
Q → O

Key candidates:
Q^+ → LMNOPQR (gives full closure except S)
Q is on the LHS only so it always has to be in key
S doesn't appear on LHS or RHS so it also always has to be a key
Therefore, we have a key "QS" and adding other attributes is obviously unnecessary (since they all are already defined by Q)

Key:
QS

c) Recall FD's we have in minimal basis:
N → M
NO → LR
P → R
Q → NOP

Our resulting set of relations would have these attributes:
R1(N,M), R2(N,O,L,R), R3(P,R), R4(Q,N,O,P)

As we see, there is no repetitions in tables, none of them can be eliminated

Since we have a key QS, and there is no S in this relations, we should add R0(Q,S) for storing the key

Eventually, we get such set of relations:
R0(Q,S), R1(N,M), R2(N,O,L,R), R3(P,R), R4(Q,N,O,P)

d) It doesn't allow redundancy since the relations are formed from minimal basis FD's, none of the relations was removed during 3NF-decomposition and each FD's LHS is a superkey for relation it formed.