CSC343 A3 Winter 2021

Shadman Kaif - 1005303137
Abdurrafay Khan - 1005215517

April 2, 2021

Exercises

1. **a) Give one example of a redundancy that relation *Reservation*, combined with FDs S, allow.**

Relation: *Reservation (sID, age, length, sName, day, cName, rating, cID)*
S = {sID → sName, rating, age; cID → cName, length}

An example of a redundancy that is allowed is when the skipper is consistent, but the craft differs. This doesn't violate the FDs S as the skipper functionally determines their name, rating and age, but the skipper can reserve multiple crafts. An example of this is illustrated below.

sID	age	length	sName	day	cName	rating	cID
1	20	3	John Doe	2021-03-21 4:05:00 PM	Mona	2	14
1	20	4	John Doe	2021-03-21 5:05:00 PM	Lisa	5	16

As we can see, the relation *Reservation,* is redundant since the same skipper reserves two different crafts, which also happens to be on the same day just an hour after according to the day.

2. **a) Which FDs in G violate BCNF? List them.**

BCNF requires that the LHS of an FD be a superkey.

- × KOQ^+ = KOQPSR (but not LMN), so KOQ is not a superkey and KOQ → PS violates BCNF.
- × L^+ = LKN (but not MOPQRS), so L is not a superkey and L → KN violates BCNF.
- × KQ^+ = KQRS (but not LMNOP), so KQ is not a superkey and KQ → RS violates BCNF.

Therefore, all of the functional dependencies in G violate BCNF.

2. b) Use the BCNF decomposition method to derive a redundancy-preventing, lossless, decomposition of F into a new schema consisting of relations that are in BCNF. Be sure to project the FDs from G onto the relations in your final schema. There may be more than one correct answer possible, since there are choices possible at steps in the decomposition. List your final relations alphabetically, and order the attributes within each relation alphabetically (this avoids combinatorial explosion of the number of alternatives we have to check).

• Decompose G using functional dependency KOQ → PS. KOQ^+ = KOQPSR, so this yields two relations: R1 = KOQPSR and R2 = KOQLMN.

• Project the FDs onto R1 = KOQPSR.

K	O	Q	P	S	R	Closure	FDs
✓						$K^+ = K$	None.
	✓					$O^+ = O$	None.
		✓				$Q^+ = Q$	None.
			✓			$P^+ = P$	None.
				✓		$S^+ = S$	None.
					✓	$R^+ = R$	None.
✓	✓					$KO^+ = KO$	None.
✓		✓				$KQ^+ = KQRS$	KQ → RS: violates BCNF; abort projection.

We must decompose R1 further.

• Decompose R1 using functional dependency KQ → RS. This yields two relations: R3 = KQRS and R4 = KQOP.

• Project the FDs onto R3 = KQRS.

K	Q	R	S	Closure	FDs
✓				$K^+ = K$	None.
	✓			$Q^+ = Q$	None.
		✓		$R^+ = R$	None.
			✓	$S^+ = S$	None.
✓	✓			$KQ^+ = KQRS$	$KQ \rightarrow RS$: KQ is a superkey of R3.
✓		✓		$KR^+ = KR$	None.
✓			✓	$KS^+ = KS$	None.
	✓	✓		$QR^+ = QR$	None.
	✓		✓	$QS^+ = QS$	None.
		✓	✓	$RS^+ = RS$	None.
✓	✓	✓		$KQR^+ = KQRS$	Generates weaker FD.
✓	✓		✓	$KQS^+ = KQRS$	Generates weaker FD.
✓		✓	✓	$KRS^+ = KRS$	None.
	✓	✓	✓	$QRS^+ = QRS$	None.

This relation satisfies BCNF.

• Project the FDs onto R4 = KQOP.

K	Q	O	P	Closure	FDs
✓				$K^+ = K$	None.
	✓			$Q^+ = Q$	None.
		✓		$O^+ = O$	None.
			✓	$P^+ = P$	None.
✓	✓			$KQ^+ = KQRS$	None.
✓		✓		$KO^+ = KO$	None.
✓			✓	$KP^+ = KP$	None.
	✓	✓		$QO^+ = QO$	None.
	✓		✓	$QP^+ = QP$	None.
		✓	✓	$OP^+ = OP$	None.
✓	✓	✓		$KOQ^+ = KOQPSR$	$KOQ \rightarrow P$: KOQ is a superkey of R4.
✓	✓		✓	$KQP^+ = KQPRS$	None.
✓		✓	✓	$KOP^+ = KOP$	None.
	✓	✓	✓	$QOP^+ = QOP$	None.

This relation satisfies BCNF.

• Return to R2 = KOQLMN and project the FDs onto it.

K	O	Q	L	M	N	Closure	FDs
✓						$K^+ = K$	None.
	✓					$O^+ = O$	None.
		✓				$Q^+ = Q$	None.
			✓			$L^+ = LKN$	L → KN: violates BCNF; abort projection.

We must decompose R2 further.

• Decompose R2 using functional dependency L → KN. This yields two relations: R5 = LKN and R6 = LMOQ.

• Project the FDs onto R5 = LKN.

L	K	N	Closure	FDs
✓			$L^+ = LKN$	L → KN: L is a superkey of R5
	✓		$K^+ = K$	None.
		✓	$N^+ = N$	None.
✓	✓		$LK^+ = LKN$	Generates weaker FD.
✓		✓	$LN^+ = LKN$	Generates weaker FD.
	✓	✓	$KN^+ = KN$	None.

This relation satisfies BCNF.

• Project the FDs onto R6 = LMOQ.

L	M	O	Q	Closure	FDs
✓				$L^+ = LKN$	None.
	✓			$M^+ = M$	None.
		✓		$O^+ = O$	None.
			✓	$Q^+ = Q$	None.
✓	✓			$LM^+ = LMKN$	None.
✓		✓		$LO^+ = LOKN$	None.
✓			✓	$LQ^+ = LQKN$	None.
	✓	✓		$MO^+ = MO$	None.
	✓		✓	$MQ^+ = MQ$	None.
		✓	✓	$OQ^+ = OQ$	None.
✓	✓	✓		$LMO^+ = LMOKN$	None.
✓	✓		✓	$LMQ^+ = LMQKN$	None.
✓		✓	✓	$LOQ^+ = LOQKNPSR$	None.
	✓	✓	✓	$MOQ^+ = MOQ$	None.

This relation satisfies BCNF.

• Final decomposition:
(a) R3 = KQRS with FD KQ → RS,
(b) R4 = KOPQ with FD KOQ → P,
(c) R5 = KLN with FD L → KN,
(d) R6 = LMOQ with no FDs

2. c) Does your final schema preserve dependencies? Explain why you answer yes or no.

Yes, our final schema preserves dependencies. For the original FDs, KOQ → P, KQ → RS, and L → KN in G, there is a relation that includes all of the FD's attributes. This ensures that they are preserved. The only FD that is not accounted for is KOQ → S. We can use an example to verify whether this FD is preserved.

R3 = KQRS with FD KQ → RS

K	Q	R	S
x	a	3	2
y	a	4	2

R4 = KOPQ with FD KOQ → P

K	O	P	Q
x	z	1	a
y	z	1	a

R5 = KLN with FD L → KN

L	K	N
8	x	c
9	y	c

R6 = LMOQ with no FDs

L	M	O	Q
8	n	z	a
9	n	z	a

The natural join of these tables is:

K	L	M	N	O	P	Q	R	S
x	8	n	c	z	1	a	3	2
y	9	n	c	z	1	a	4	2

As seen from the table above, all the original FDs including KOQ → S hold. Therefore, our final schema preserves all original dependencies. This makes sense as KQ → S is preserved in R3 so it makes sense to have KOQ → S preserved as well since KOQ is a superset of KQ.

2. d) BCNF guarantees a lossless join. However demonstrate this to a possibly-skeptical observer using the Chase Test.

Restating our relations and FDs established in Question 2b:

(a) R3 = KQRS with FD KQ → RS,
(b) R4 = KOPQ with FD KOQ → P,
(c) R5 = KLN with FD L → KN,
(d) R6 = LMOQ with no FDs

We can make a table to demonstrate the Chase Test:

	K	L	M	N	O	P	Q	R	S
R3	**k**	l_3	m_3	n_3	o_3	p_3	**q**	**r**	**s**
R4	**k**	l_4	m_4	n_4	**o**	**p**	**q**	$\cancel{r_4}$ **r**	$\cancel{s_4}$ **s**
R5	**k**	**l**	m_5	**n**	o_5	p_5	q_5	r_5	s_5
R6	$\cancel{k_6}$ **k**	**l**	**m**	$\cancel{n_6}$ **n**	**o**	$\cancel{p_6}$ **p**	**q**	$\cancel{r_6}$ **r**	$\cancel{s_6}$ **s**

As we see the R6 row has the tuple <**k, l, m, n, o, p, q, r, s**>, we can conclude that the chase test demonstrates that it is a lossless-join decomposition as BCNF guarantees.

3. a) Find a minimal basis for S. Your final answer must put the FDs in ascending alphabetical order, and the attributes within the LHS and RHS of each FD into alphabetical order.

S = {ACDE → B; B → CF; CD → AF; BCF → AD; ABF → H}

Step 1: Split the RHSs to get our initial set of FDs, S1:

(a) ACDE → B
(b) B → C
(c) B → F
(d) CD → A
(e) CD → F
(f) BCF → A
(g) BCF → D
(h) ABF → H

Step 2: For each FD, try to reduce the LHS:

(a)

A	C	D	E	Closure	Reduction
✓				$A^+ = A$	None.
	✓			$C^+ = C$	None.
		✓		$D^+ = D$	None.
			✓	$E^+ = E$	None.
✓	✓			$AC^+ = AC$	None.
✓		✓		$AD^+ = AD$	None.
✓			✓	$AE^+ = AE$	None.
	✓	✓		$CD^+ = CDAF$	None.
	✓		✓	$CE^+ = CE$	None.
		✓	✓	$DE^+ = DE$	None.
✓	✓	✓		$ACD^+ = ACDF$	None.
✓	✓		✓	$ACE^+ = ACE$	None.
✓		✓	✓	$ADE^+ = ADE$	None.
	✓	✓	✓	$CDE^+ = CDEAFB$	CDE → B

Thus, we can reduce this FD to: CDE → B.

(b) B^+ = BCFADH. We cannot reduce this FD as it is already a singleton. FD remains B ⟶ C.

(c) B^+ = BCFADH. We cannot reduce this FD as it is already a singleton. FD remains B ⟶ F.

(d)

C	D	Closure	Reduction
✓		$C^+ = C$	None.
	✓	$D^+ = D$	None.

We cannot reduce this FD. FD remains CD ⟶ A.

(e) Similarly to one above, we cannot reduce this FD as singleton LHS don't yield anything. FD remains CD ⟶ F.

(f)

B	C	F	Closure	Reduction
✓			B^+ = BCFADH	B ⟶ A
	✓		$C^+ = C$	None.
		✓	$F^+ = F$	None.

Thus, we can reduce this FD to: B ⟶ A.

(g)

B	C	F	Closure	Reduction
✓			B^+ = BCFADH	B ⟶ D
	✓		$C^+ = C$	None.
		✓	$F^+ = F$	None.

Thus, we can reduce this FD to: B ⟶ D.

(h)

A	B	F	Closure	Reduction
✓			$A^+ = A$	None.
	✓		$B^+ = BCFADH$	$B \rightarrow H$
		✓	$F^+ = F$	None.

Thus, we can reduce this FD to: $B \rightarrow H$.

Therefore, the new set of FDs, S2 is:
(a) $CDE \rightarrow B$
(b) $B \rightarrow C$
(c) $B \rightarrow F$
(d) $CD \rightarrow A$
(e) $CD \rightarrow F$
(f) $B \rightarrow A$
(g) $B \rightarrow D$
(h) $B \rightarrow H$

Step 3: Try to eliminate each FD.
(a) $CDE^+_{S2 - \{(a)\}}$ = CDEAF. We need this FD as it doesn't have B.
(b) $B^+_{S2 - \{(b)\}}$ = BFADH. We need this FD as it doesn't have C.
(c) $B^+_{S2 - \{(c)\}}$ = BCADHA**F**. We can remove this FD as it has F.
(d) $CD^+_{S2 - \{(c), (d)\}}$ = CDF. We need this FD as it doesn't have A.
(e) $CD^+_{S2 - \{(c), (e)\}}$ = CDA. We need this FD as it doesn't have F.
(f) $B^+_{S2 - \{(c), (f)\}}$ = BCDH**A**F. We can remove this FD as it has A.
(g) $B^+_{S2 - \{(c), (g)\}}$ = BCAH. We need this FD as it doesn't have D.
(h) $B^+_{S2 - \{(c), (h)\}}$ = BCADF. We need this FD as it doesn't have H.

Our new set of FDs, S3, is:
(a) $CDE \rightarrow B$
(b) $B \rightarrow C$
(c) $CD \rightarrow A$
(d) $CD \rightarrow F$
(f) $B \rightarrow D$
(g) $B \rightarrow H$

Step 4: We need to perform one final check to ensure that the FDs within S3 cannot be discarded.
(a) $CDE^{+}_{S3\ -\ \{(a)\}}$ = CDEAF. We need this FD as it doesn't have B.
(b) $B^{+}_{S3\ -\ \{(b)\}}$ = BDH. We need this FD as it doesn't have C.
(c) $CD^{+}_{S3\ -\ \{(c)\}}$ = CDF. We need this FD as it doesn't have A.
(d) $CD^{+}_{S3\ -\ \{(d)\}}$ = CDA. We need this FD as it doesn't have F.
(f) $B^{+}_{S3\ -\ \{(f)\}}$ = BCH. We need this FD as it doesn't have D.
(g) $B^{+}_{S3\ -\ \{(g)\}}$ = BCDAF. We need this FD as it doesn't have H.

Our final set of FDs, S4, is a minimal basis:
(a) B → C
(b) B → D
(c) B → H
(d) CD → A
(e) CD → F
(f) CDE → B

3. b) Find all the keys for R using your solution for a minimal basis.

The following table is based on our final set of FDs, S4, from part 3a.

Attribute	Appears on LHS	Appears on RHS	Conclusion
A	-	✓	is not in any key
E	✓	-	must be in every key
F	-	✓	is not in any key
G	-	-	must be in every key
H	-	✓	is not in any key
B, C, D	✓	✓	must check

This means that we only have to consider all combinations of B, C, and D. For each, we must add in E and G, since they are in every key.

B	C	D	E	G	Closure	Key
✓			✓	✓	BEG^+ = BEGCDHAF	Yes.
	✓		✓	✓	CEG^+ = CEG	No.
		✓	✓	✓	DEG^+ = DEG	No.
✓	✓		✓	✓	$BCEG^+$ = BCEGDHAF	Weaker than BEG.
✓		✓	✓	✓	$BDEG^+$ = BDEGCHAF	Weaker than BEG.
	✓	✓	✓	✓	$CDEG^+$ = CDEGBHAF	Yes.

Therefore, the minimum candidate keys of R are: BEG and CDEG.

3. c) Use the 3NF synthesis algorithm to find a lossless, dependency-preserving decomposition of relation R into a new schema consisting of relations that are in 3NF. Your final answer should combine FDs with the same LHS to create a single relation. If your schema has a relation that is a subset of another, keep only the larger relation.

• We can merge the RHS to yield a smaller set of relations and they will still form a lossless and dependency-preserving decomposition of relation R into a collection of relations that are in 3NF.

• We can call the new set S5:
(a) $B \rightarrow CDH$
(b) $CD \rightarrow AF$
(c) $CDE \rightarrow B$

• The set of relations that would result would have these attributes:
R1(B, C, D, H)
R2(A, C, D, F)
R3(B, C, D, E)

• From part 3b), we know the candidate keys are BEG and CDEG. Since attribute G is not included in R1, R2 or R3, we must add one of the two candidate keys as a new relation R4. We select BEG.

The final set of relations is:
R1(B, C, D, H)
R2(A, C, D, F)
R3(B, C, D, E)
R4(B, E, G)

3. (d) Does your solution allow redundancy? Explain how (with an example), or why not.

According to the worksheet done in class, "since we formed each relation from an FD, the LHS of those FDs are superkeys for their relations. However, there may be other FDs that violate BCNF and therefore allow redundancy. The only way to find out is to project the FDs onto each relation."

We can quite quickly find a relation that violates BCNF without doing all the full projections: We can try to project the FD, B $\rightarrow$ CDH, onto the relation R3(B, C, D, E). And B^+ = BCDHAF, so B is not a superkey of this relation as it doesn't contain E. Therefore, the schema allows redundancy.