csc343 winter 2021
assignment #3: database (re)design
sample solutions

goals

This assignment aims to help you learn to:

- design a good schema
- understand violations of functional dependencies
- create a minimal basis for a set of functional dependencies
- project a set of functional dependencies onto a set of attributes
- find all the keys for a set of functional dependencies
- re-factor relation(s) into BCNF
- re-factor relations into 3NF

Your assignment must be typed to produce a PDF document **a3.pdf**, and a plain text document **reservation.ddl** (hand-written submissions are not acceptable). You may work on the assignment in groups of 1 or 2, and submit a single assignment for the entire group on MarkUs. You must establish your group well before the due date by submitting an incomplete, or even empty, submission (of course, we only grade the last submission before the due date/time).

exercises

1. Relation *Reservation* is meant to keep track of which skipper reserves which craft on which date, but its design has some redundancy:

   $$Reservation(sID, age, length, sName, day, cName, rating, cID)$$

   ... where sID identifies the skipper, $sName$ is the skipper's name, whereas $rating$ and age record the skipper's skill (a number between 0 and 5, inclusive) and age (a number greater than 0). The reserved craft is identified by cID, its name is $cName$, $length$ is in feet, and the date and time the craft is reserved for by the skipper is day. The following dependencies hold:[1]

   $$S = \{sID \to sName, rating, age; \;\; cID \to cName, length\}$$

[1]Since multiple attributes may be separated by commas, we use semicolons to separate FDs.

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

(b) Design a schema in DDL called `reservation.ddl` that represents the same information as *Reservation*, using exactly the same attribute names, but has the following goals, in descending order of importance:

   i. has as few redundancies as possible;

   ii. allows as few NULL or DEFAULT values as possible;

   iii. enforces as many constraints from the description above as possible, without using triggers or assertions.

   Your schema should import into psql without error using the command:

```
\i reservation.ddl
```

   While you are developing your schema you may want to ensure that your previous version is removed before you read in a new one:

```
drop schema if exists reservation cascade;
create schema reservation;
set search_path to reservation;
```

   Use comments at the beginning of `reservation.ddl` to explain which constraints were not enforced (if any) and which redundancies are still allowed (if any). As the designer you have freedom to choose datatypes for the various attributes.

sample solution:

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

Using the provided functional dependencies and relation, the following two rows could exist:

sID	age	length	sName	day	cN ame	rating	cID
1	20	5	Charlie	13-09-2021 21:00:00			
1	20	5	Charlie	14-09-2021 06:00:00			

It would be possible to have two rows with every thing in common except the day attribute.

(b)

```
drop schema if exists reserve cascade;
create schema reserve;
set search_path to reserve;

-- decompose relation Reserve to create two new entities,
-- one for Skipper, another for Craft.
-- Reserve can refer to these with a foreign key for each.

create table Skipper (
    -- serial autoincrements an integer key
        sID serial  PRIMARY KEY,
        sName text NOT NULL,
        rating integer CONSTRAINT skipper_rating
```

```
         check (rating in (0,1,2,3,4,5)),
        age DATE NOT NULL
);

create table Craft (
    -- serial autoincrements an integer key
        cID serial PRIMARY KEY,
        length float NOT NULL,
        cName text NOT NULL
);

create table Reserve (
        sID integer REFERENCES Skipper(sID),
        cID integer REFERENCES Craft(cID),
        day DATE NOT NULL,
        -- naming our key is not necessary but good practice
        -- all three attributes combined as a key will avoid double booking
        CONSTRAINT reserve_key PRIMARY KEY  (sID, cID, day)

);
```

Note: Students may decide that avoiding double-booking is so wide-spread as to be common sense, so they can decide that cID,day is a key or that sID,day is a key, but they should make sure to state this in their comments.

2. Relation F has attributes $KLMNOPQRS$ and functional dependencies G:

$$G = \{KOQ \to PS, L \to KN, KQ \to RS\}$$

   (a) Which FDs in G violate BCNF? List them.
   (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).
   (c) Does your final schema preserve dependencies? Explain why you answer yes or no.
   (d) BCNF guarantees a lossless join. However demonstrate this to a possibly-skeptical observer using the Chase Test.

Show us the steps in your work. This allows us to assign part marks, if needed, and to be sure that you have not taken any unwarranted short cuts.

sample solution:

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

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

- KOQ$^+$ = KOPQRS, so KOQ is not a superkey and KOQ $\to$ PS violates BCNF.

- L$^+$ $= KLN$, so L is not a superkey and L $\rightarrow$ KN violates BCNF.
- KQ$^+$ $= KQRS$, so KQ is not a superkey and KQ $\rightarrow$ RS violates BCNF.

(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 R using FD KOQ $\rightarrow$ PS. KOQ$^+$ = KOPQRS, so this yields two relations $R_1(KOPQRS)$ and $R_2(KLMNOQ)$

First, project FDs onto $R_1(KOPQRS)$.

K	O	P	Q	R	S	closure	FDs
✓						$K^+ = K$	nothing
	✓					$O^+ = O$	nothing
		✓				$P^+ = P$	nothing
			✓			$Q^+ = Q$	nothing
				✓		$R^+ = R$	nothing
					✓	$S^+ = S$	nothing
✓	✓					$KO^+ = KO$	nothing
✓		✓				$KP^+ = KP$	nothing
✓			✓			$KQ^+ = KQRS$	$KQ \rightarrow RS$: violates BCNF; abort the projection.

We must decompose R_1 further.

Decompose R_1 using FD KQ $\rightarrow$ RS. $KQ^+ = KQRS$, so this yields two relations $R_3(KQRS)$ and $R_4(KOPQ)$.

K	Q	R	S	closure	FDs
✓				$K^+ = K$	nothing
	✓			$Q^+ = Q$	nothing
		✓		$R^+ = R$	nothing
			✓	$S^+ = S$	nothing
✓	✓			$KQ^+ = KQRS$	$KQ \rightarrow RS$, KQ is a superkey of R_3
supersets of KQ				irrelevant	can only generate weaker FDs than what we already have
✓		✓		$KR^+ = KR$	nothing
✓			✓	$KS^+ = KS$	nothing
	✓	✓		$QR^+ = QR$	nothing
	✓		✓	$QS^+ = QS$	nothing
		✓	✓	$RS^+ = RS$	nothing
✓		✓	✓	$KRS^+ = KRS$	nothing
	✓	✓	✓	$QRS^+ = QRS$	nothing

Relation R_3 satisfies BCNF.

Project FDs onto $R_4(KOPQ)$. Relation R_4 satisfies BCNF.

K	O	P	Q	closure	FDs
✓				$K^+ = K$	nothing
	✓			$O^+ = O$	nothing
		✓		$P^+ = P$	nothing
			✓	$Q^+ = Q$	nothing
✓	✓			$KO^+ = KO$	nothing
✓		✓		$KP^+ = KP$	nothing
✓			✓	$KQ^+ = KQRS$	nothing
	✓	✓		$OP^+ = OP$	nothing
	✓		✓	$OQ^+ = OQ$	nothing
		✓	✓	$PQ^+ = PQ$	nothing
✓	✓	✓		$KOP^+ = KOP$	nothing
✓	✓		✓	$KOQ^+ = KOQRSP$	$KOQ \rightarrow P$, KOQ is a superkey of R_4
✓		✓	✓	$KPQ^+ = KPQ$	nothing
	✓	✓	✓	$OPQ^+ = OPQ$	nothing

Now we return and project FDs onto $R_2(KLMNOQ)$:

K	L	M	N	O	Q	closure	FDs
✓						$K^+ = K$	nothing
	✓					$L^+ = KLN$	$L \rightarrow KN$: violates BCNF; abort projection.

We must decompose R_2 further. Decompose R_2 using FD $L \rightarrow KN$. $L^+ = KLN$, so this yields two relations: $R_5(KLN)$ and $R_6(LMOQ)$.

Project FDs onto $R_5(KLN)$:

K	L	N	closure	FDs
✓			$K^+ = K$	nothing
	✓		$L^+ = LKN$	$L \rightarrow KN$: L is a superkey of R_5.
supersets of L			irrelevant	can only generate weaker FDs than what we already have
		✓	$N^+ = N$	nothing
✓		✓	$KN^+ = KN$	nothing

R_5 satisfies BCNF.

Project FDs onto $R_6(LMOQ)$:

L	M	O	Q	closure	FDs
✓				$L^+ = LKN$	nothing
	✓			$M^+ = M$	nothing
		✓		$O^+ = O$	nothing
			✓	$Q^+ = Q$	nothing
✓	✓			$LM^+ = LMKN$	nothing
✓		✓		$LO^+ = LKNO$	nothing
✓			✓	$LQ^+ = LQKNRS$	nothing
	✓	✓		$MO^+ = MO$	nothing
	✓		✓	$MQ^+ = MQ$	nothing
		✓	✓	$OQ^+ = OQ$	nothing
✓	✓	✓		$LMO^+ = LMNOK$	nothing
✓	✓		✓	$LMQ^+ = LMQKNRS$	nothing
✓		✓	✓	$LOQ^+ = LKNOQRS$	nothing
	✓	✓	✓	$MOQ^+ = MOQ$	nothing

Relation R_6 satisfies BCNF.

Final decomposition:

- $R_3(KQRS)$, $KQ \rightarrow RS$.
- $R_4(KOPQ)$, $KOQ \rightarrow P$.
- $R_5(KLN)$, $L \rightarrow KN$.
- $R_6(LMOQ)$, no FDs.

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

FDs $KQ \rightarrow RS$ and $L \rightarrow KN$ are preserved. $KOQ \rightarrow S$ is given by a stronger $KQ \rightarrow S$ in R3. Decomposition is dependency preserving.

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

We say if a decomposition is lossless if the original relation can be recovered completely by natural joining the decomposed relations. Given decomposition of R into $R_3(KQRS)$, $R_4(KOPQ)$, $R_5(KLN)$, and $R_6(LMOQ)$ and FDs $KQ \rightarrow RS$, $KOQ \rightarrow P$ and $L \rightarrow KN$.

Note: we indicate the unknown values with a subscripted variable.

relation	K	L	M	N	O	P	Q	R	S
R5(KLN)	k	l	m_1	n	o_1	p_1	q_1	r_1	s_1
R4($KOPQ$)	k	l_2	m_2	n_2	o	p	q	r_2	s_2
R3($KQRS$)	k	l_3	m_3	n_3	o_3	p_3	q	r	s_3
R6($LMOQ$)	k_4	l	m	n_4	o	p_4	q	r_4	s_4

Use $L \rightarrow KN$:

relation	K	L	M	N	O	P	Q	R	S
$R5(KLN)$	k	l	m_1	n	o_1	p_1	q_1	r_1	s_1
$R4(KOPQ)$	k	l_2	m_2	n_2	o	p	q	r_2	s_2
$R3(KQRS)$	k	l_3	m_3	n_3	o_3	p_3	q	r	s
$R6(LMOQ)$	**k**	l	m	**n**	o	p_4	q	r_4	s_4

Use $KOQ \to PS$:

relation	K	L	M	N	O	P	Q	R	S
$R5(KLN)$	k	l	m_1	n	o_1	p_1	q_1	r_1	s_1
$R4(KOPQ)$	k	l_2	m_2	n_2	o	p	q	r_2	**s**
$R3(KQRS)$	k	l_3	m_3	n_3	o_3	p_3	q	r	s
$R6(LMOQ)$	k	l	m	n	o	**p**	q	r_4	**s**

Use $KQ \to RS$:

relation	K	L	M	N	O	P	Q	R	S
$R5(KLN)$	k	l	m_1	n	o_1	p_1	q_1	r_1	s_1
$R4(KOPQ)$	k	l_2	m_2	n_2	o	p	q	**r**	s
$R3(KQRS)$	k	l_3	m_3	n_3	o_3	p_3	q	r	s
$R6(LMOQ)$	k	l	m	n	o	p	q	**r**	s

We have a complete row (no subscripted values) demonstrating that BCNF guarantees a lossless join.

3. Relation R has attributes $ABCDEFGH$ and functional dependencies S:

$$S = \{ACDE \to B, B \to CF, CD \to AF, BCF \to AD, ABF \to H\}$$

   (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.

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

   (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.

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

sample solution:

(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.

To find a **minimal basis**, we'll first simplify FDs to singleton right-hand sides. We'll also number the resulting FDs for easy reference, and call this set S_1:

1. $ACDE \to B$
2. $B \to C$
3. $B \to F$

4. $CD \rightarrow A$
5. $CD \rightarrow F$
6. $BCF \rightarrow A$
7. $BCF \rightarrow D$
8. $ABF \rightarrow H$

We next try to **reduce** the LHS of FDs with multiple attributes on the LHS. For these closures, we will close over the full set S_1, including even the FD being considered for simplification; remember that we are not considering removing the FD, just strengthening it. Note: The order in which we consider attributes to reduce may affect the results we get, but we will always get *a* minimal basis.

1. $ACDE \rightarrow B$
   - $A^+ = A$, so we can't reduce the LHS to A.
   - $C^+ = C$, so we can't reduce the LHS to C.
   - $D^+ = D$, so we can't reduce the LHS to D.
   - $E^+ = E$, so we can't reduce the LHS to E.
   - $AC^+ = AC$, so we can't reduce the LHS to AC.
   - $AD^+ = AD$, so we can't reduce the LHS to AD.
   - $AE^+ = AE$, so we can't reduce the LHS to AE.
   - $CD^+ = ACDF$, so we can't reduce the LHS to CD.
   - $CE^+ = CE$, so we can't reduce the LHS to CE.
   - $DE^+ = DE$, so we can't reduce the LHS to DE.
   - $ACD^+ = ACDF$, so we can't reduce the LHS to ACD.
   - $ACE^+ = ACE$, so we can't reduce the LHS to ACE.
   - $ADE^+ = ADE$, so we can't reduce the LHS to ADE.
   - $CDE^+ = ACDEF$, so we can't reduce the LHS to CDE.
2. $B \rightarrow C$, nothing to reduce.
3. $B \rightarrow F$, nothing to reduce.
4. $CD \rightarrow A$
   - $C^+ = C$, so we can't reduce the LHS to C.
   - $D^+ = D$, so we can't reduce the LHS to D.
5. $CD \rightarrow F$
   - $C^+ = C$, so we can't reduce the LHS to C.
   - $D^+ = D$, so we can't reduce the LHS to D.
6. $BCF \rightarrow A$
   - $B^+ = \mathbf{A}BCDFH$, so we can reduce the LHS: $B \rightarrow A$ ✓
7. $BCF \rightarrow D$
   - $B^+ = ABC\mathbf{D}FH$, so we can reduce the LHS: $B \rightarrow D$ ✓
8. $ABF \rightarrow H$
   - $A^+ = A$
   - $B^+ = ABCDF\mathbf{H}$, so we can reduce the LHS: $B \rightarrow H$ ✓

We call the revised set of FDs after reducing LHS S_2:

1. $ACDE \rightarrow B$
2. $B \rightarrow C$
3. $B \rightarrow F$
4. $CD \rightarrow A$
5. $CD \rightarrow F$
6. $B \rightarrow A$
7. $B \rightarrow D$
8. $B \rightarrow H$

Now we look for redundant FDs to eliminate. Each row in the table below indicates which of the 8 FDs we still have on hand as we consider removing the next one. Of course, as we do the closure test to see whether we can remove $X \rightarrow Y$, we can't use $X \rightarrow Y$ itself, so an FD is never included in its own row. Again, note that there may be more than one correct result depending on the order of elimination.

FD	Exclude from S_2	Closure	Decision
1	1	$ACDE^+_{S_1-1} = ACDEF$	keep
2	2	$B^+_{S_2-2} = ABDFH$	keep
3	3	$B^+_{S_2-3} = ABCDHF$	FD $B \rightarrow F$ can be removed.
4	3,4	$CD^+_{S_2-3-4} = CDF$	keep
5	3,5	$CD^+_{S_2-3-5} = ACD$	keep
6	3,6	$B^+_{S_2-3-6} = ABCDH$	FD $B \rightarrow A$ can be removed.
7	3,6,7	$B^+_{S_2-3-6-7} = BCH$	keep
8	3,6,8	$B^+_{S_2-3-6-8} = ABCDF$	keep

No further simplifications are possible, so the following set S_3 is a minimal basis (ordered alphabetically):

1. $ACDE \rightarrow B$
2. $B \rightarrow C$
7. $B \rightarrow D$
8. $B \rightarrow H$
4. $CD \rightarrow A$
5. $CD \rightarrow F$

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

Attribute G is not found in the Minimal Basis FDs. That means that it has to be in every key - there is no other way to get it. In fact, even if an attribute appears in an FD, if it never appears in a RHS, it will have to be in every key. E is an example of this. If, on the other hand, an attribute appears only on the RHS of the FDs, never the left, it is of no help to us in computing closures. It can not be in any key, because we would always remove it and still get the same closure. H is an example of this.

This means we only have to consider all combinations of A, B, C, and D: A, B, C, D, AB, AC, AD, BC, BD, CD, ABC, ABD, CBD. For each, we must add in E and G since they must be in every key. So, the list of possible keys to consider is AEG, BEG, CEG, DEG, ABEG, ACEG, ADEG, BCEG, BDEG, CDEG, ABCEG, ABDEG, CBDEG.

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

- $AEG^+ = AEG$, this is not a key.
- $BEG^+ = ABCDEFGH$, BEG is a key, and ABEG, BCEG, BDEG, ABCEG, ABDEG, CBDEG are superkeys.
- $CEG^+ = CEG$, this is not a key.
- $DEG^+ = DEG$, this is not a key.
- $ACEG^+ = ACEG$, this is not a key.
- $ADEG^+ = ADEG$, this is not a key.
- $CDEG^+ = ABCDEFGH$, CDEG is a key.

In conclusion, there are two keys for R: BEG and CDEG.

(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.

Following the 3NF synthesis algorithm, we would get one relation for each FD of the minimal basis from the previous step. However, we should merge the right-hand sides before doing so. This will 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.

Lets call the revised FDs S_4:
$ACDE \rightarrow B$
$B \rightarrow CDH$
$CD \rightarrow AF$

The resulting set of relations consists of:

$R_1(A, B, C, D, E)$
$R_2(B, C, D, H)$
$R_3(A, C, D, F)$

None of them contains the key of the relation, so we add $R_4(B, E, G)$.

Final decomposition:
$R_1(A, B, C, D, E)$
$R_2(B, C, D, H)$
$R_3(A, C, D, F)$
$R_4(B, E, G)$

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

$CD \rightarrow AF$ can be projected onto R_1 and $CD^+ = CDAF$ meaning CD is not a superkey for R_1 therefore the schema allows redundancy.

submissions

Submit **a3.pdf** and **reservation.ddl** on MarkUs. One submission per group, whether a group is one or two people. You declare a group by submitting an empty, or partial, file, and this should be done **well before** the due date. You may always replace such a file with a better version, until the due date.

Double check that you have submitted the correct version of your file by downloading it from MarkUs.