csc343 winter 2021
assignment #3: database (re)design
due April 2nd, 4 p.m.

Eric Zhu and Kristin Huang

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]

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

[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.

sID	age	length	sName	day	cName	rating	cID
1	21	150	Bob Ross	2020/1/1 12:00	HMS Liz	5	1
1	21	155	Bob Ross	2020/1/1 13:00	HMS James	5	2

Here we see that we have redundant sID, sName, rating, and age.

(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.

Completed in `Reservation.ddl` as required. Please refer to that file for the solution.

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.
Check if this non-trivial FD violate BCNF: KOQ -> PS. We check if KOQ is a superkey.

$$KOQ^+ = \{K, O, Q, P, S, R\}$$

Here we see that KOQ is not a superkey because the closure doesn't contain all attributes. The FD is also non-trivial, hence it violates BCNF.
Check if this non-trivial FD violate BCNF: L -> KN. We check if L is a superkey.

$$L^+ = \{L, K, N\}$$

Here we see that L is not a superkey because the closure doesn't contain all attributes. The FD is also non-trivial, hence it violates BCNF.
Check if this non-trivial FD violate BCNF: KQ -> RS. We check if KQ is a superkey.

$$KQ^+ = \{K, Q, R, S\}$$

Here we see that KQ is not a superkey because the closure doesn't contain all attributes. The FD is also non-trivial, hence it violates BCNF.

Answer: The FDs in G that violate BCNF are all the functional dependencies: KOQ -> PS, L -> KN, KQ -> RS.

(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).

We pick KOQ -> PS to start the BCNF decomposition since this FD violates BCNF.

$$KOQ^+ = \{K, O, Q, P, S, R\}$$

Our initial relation F has attributes $KLMNOPQRS$.

R1 = $KOQ^+ = KOQPSR$

R2 = F - KOQ^+ + KOQ = KLMNOPQRS - KOQPSR + KOQ = KLMNOQ.

Now inspect the FDs to see if there are BCNF violations.

Part 1)

For R1 = KOQPSR, we project the functional dependencies: KOQ -> PS, KQ -> RS

BCNF violation testing for functional dependency KOQ -> PS:

$$KOQ^+ = \{K, O, Q, P, S, R\}$$

BCNF violation testing for functional dependency KQ -> RS:

$$KQ^+ = \{K, Q, R, S\}$$

So since KQ violates the BCNF condition, we will break R1 down further into Ra and Rb using KQ -> RS.

Part 1A)

Let Ra = KQ^+ = KQRS

Part 1B)

Then let Rb = R1 - Ra^+ + Ra = KOQPSR - KQRS + KQ = KOQP

Then for Ra = KQRS, we project the functional dependency of KQ -> RS.

We test if the functional dependency KQ -> RS violates BCNF for Ra:

$$KQ^+ = \{K, Q, R, S\}$$

So we see KQ -> RS does not violate BCNF. Hence we can stop recursing at Ra.

Then for Rb = KOQP, we have the functional dependency KOQ -> P. This KOQ is a superkey so we do not violate BCNF. Hence we can stop recursing.

Part 2)

Now we check to see if R2 need to be further decomposed. Our R2 = KLMNOQ. We can project the functional dependencies: L -> KN onto R2.

Now we test to see if L -> KN violates BCNF with R2.

$$L^+ = \{L, K, N\}$$

Here L is not a superkey and L -> KN is non-trivial so this violates BCNF.

Then we can decompose R2 into Rc and Rd.

Part 2A) Let Rc = $L^+ = LKN$

Part 2B) Let Rd = $R2 - L^+ + L$ = LMOQ.

We project the functional dependency L -> KN onto Rc and test if it violates BCNF.

$$L^+ = \{L, K, N\}$$

L is a superkey for Rc so this functional dependency does not violate BCNF. Hence we can stop recursing at Rc.

We cannot project any functional dependency onto Rd so we can stop recursing at Rd.

Final Answer: Organized alphabetically, we have:

Ra: attributes KLN, functional dependencies: L -> KN.

Rb: attributes KOPQ, functional dependencies: KOQ -> P

Rc: attributes KQRS, functional dependencies: KQ -> RS.

Rd: attributes LMOQ, no functional dependencies.

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

Yes, our final schema does preserve dependencies. Our final schema still have the FDs: L -> KN, KQ -> RS. And we can also derive the FD KOQ -> PS.

Since we have the FD: KQ -> RS, then we can add an extra attribute on the left hand side so that KOQ -> RS.

Our final schema also have the FD KOQ -> P, so combining these, we get: KOQ -> PRS. Hence we can derive KOQ -> PS.

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

Assume a tuple $t = (k, l, m, n, o, p, q, r, s)$.

We use these relations to create rows in the table:

Ra: attributes KLN, functional dependencies: L -> KN.

Rb: attributes KOPQ, functional dependencies: KOQ -> P.

Rc: attributes KQRS, functional dependencies: KQ -> RS.

Rd: attributes LMOQ, no functional dependencies.

K	L	M	N	O	P	Q	R	S
k	l	m1	n	o1	p1	q1	r1	s1
k	l2	m2	n2	o	p	q	r2	s2
k	l3	m3	n3	o3	p3	q	r	s
k4	l	m	n4	o	p4	q	r4	s4

Then we use the functional dependencies to modify the table. Using L -> KN, the last row is changed to the version below.

K	L	M	N	O	P	Q	R	S
k	l	m1	n	o1	p1	q1	r1	s1
k	l2	m2	n2	o	p	q	r2	s2
k	l3	m3	n3	o3	p3	q	r	s
k	l	m	n	o	p4	q	r4	s4

Then using KOQ -> PS, we change the second row, last row to the version below.

K	L	M	N	O	P	Q	R	S
k	l	m1	n	o1	p1	q1	r1	s1
k	l2	m2	n2	o	p	q	r2	s
k	l3	m3	n3	o3	p3	q	r	s
k	l	m	n	o	p	q	r4	s

Then using KQ -> RS, we change the second row, last row to the version below.

K	L	M	N	O	P	Q	R	S
k	l	m1	n	o1	p1	q1	r1	s1
k	l2	m2	n2	o	p	q	r	s
k	l3	m3	n3	o3	p3	q	r	s
k	l	m	n	o	p	q	r	s

Then we see the last row is $(k, l, m, n, o, p, q, r, s)$, so the tuple $t = (k, l, m, n, o, p, q, r, s)$. Therefore $t \in R$ so we have a lossless join.

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.

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.

Breaking the right hand sides of the set of functional dependencies, S, we get:

$$S = \{ACDE \to B, B \to C, B \to F, CD \to A, CD \to F, BCF \to A, BCF \to D, ABF \to H\}$$

Now, we must find the minimal basis for S. We do so by computing the closure of each attribute in the above left hand sides.

i. We get for A that $A^+ = \{A\}$
ii. We get for B that $B^+ = \{BCFADH\}$
iii. We get for C that $C^+ = \{C\}$
iv. We get for D that $D^+ = \{D\}$
v. We get for E that $E^+ = \{E\}$
vi. We get for F that $F^+ = \{F\}$

From the list above, we can reduce the LHS as follows, which we call $S2$:

i. $CDE \to B$ because the closure of AB^+ includes B.
ii. $B \to C$ because it's a singleton
iii. $B \to F$ because it's a singleton
iv. $CD \to A$ because no singleton LHS yields anything
v. $CD \to F$ because no singleton LHS yields anything
vi. $B \to A$ because A is in the closure of B
vii. $B \to D$ because D is in the closure of B
viii. $B \to H$ because H is in the closure of B

Then we try to eliminate each FD:

i. $CDE^+_{S2-i} = CDE$. We need this FD.
ii. $B^+_{S2-ii} = BFADH$. We need this FD.
iii. $B^+_{S2-iii} = BCADH\underline{F}$. We don't need this FD since we get F in the closure.
iv. $CD^+_{S2-\{iv,iii\}} = CDF$. We need this FD.
v. $CD^+_{S2-\{v,iii\}} = CDA$. We need this FD.
vi. $B^+_{S2-\{vi,iii\}} = BC\underline{A}DHF$. We don't need this FD since we get A in the closure.
vii. $B^+_{S2-\{vii,vi,iii\}} = BCAHF$. We need this FD.
viii. $B^+_{S2-\{viii,vi,iii\}} = BCADF$. We need this FD.

Our final set of FDs is:

i. $CDE \to B$
ii. $B \to CDH$
iii. $CD \to AF$

In other words, in set notation, we have:

$$S = \{CDE \to B, B \to CDH, CD \to AF\}$$

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

From lecture we learned that if an attribute doesn't appear anywhere in the FDs, we must have it in every key. The attribute that satisfies this rule is G. Then if an attribute appears in an FD but never on a RHS, it will also have to be in every key, so therefore, E satisfies this rule.

Next, the attributes that only appear on the RHS is H, so H is not a key. That leaves us with $\{A, B, C, D, F\}$ to consider.

So we check every possible combination of $\{A, B, C, D, F\}$ with E and G. Using the minimal basis we know that the closure of B includes more than itself, and the closure of CD includes more than itself. Thus we found the following two combinations.
$BEG^+ = ABCDEFGH$
$CDEG^+ = ABCDEFGH$
Since BEG and CDEG functionally determines all other attributes, meaning they are the keys of R.

We conclude that our key is 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.

We can use the minimal basis in the 3NF synthesis algorithm.
For each FD below, we get one relation:

i. $CDE \to B$
ii. $B \to CDH$
iii. $CD \to AF$

Thus our relations:
R1(B, C, D, E), R2(B, C, D, H), R3(A, C, D F).
We see that none of the relations contain the keys BEG or CDEG. Hence we add one of the keys as a relation.
Then we have (in alphabetical order):
R1(A, C, D F)
R2(B, C, D, E)
R3(B, C, D, H)
R4(B, E, G)

Functional Dependencies:

i. $CDE \to B$
ii. $B \to CDH$
iii. $CD \to AF$

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

We formed each relation ($R1$, $R2$, $R3$, $R4$) from their respective FDs, so the LHS of those FDs are superkeys for their respective relations as stated on the worksheet about 3NF. So we must project the FDs onto each relation to test for redundancy.
Similarly to the worksheet, we see that $B \to C$ and $B \to D$ hold in R, and so $B \to CD$. And that $B \to CD$ will project onto relation $R1$, but the closure of B, i.e., $B^+ = BCDH$, so B is **not** a superkey of this relation. Thus, this schema allows for redundancy.

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.

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.