CSC343H1: Assignment 3

Junming Zhang
Yuchen Fan

Due: August 5th, 2019 before 10 p.m.

Database Design and SQL DDL

1. Consider the relation $R(A, B, C, D, E, F)$. Let the set of FDs for R be $\{A \to B, CD \to A, CB \to D, CE \to D, AE \to F\}$.

   (a) What are all of the keys for R?

Solution.
CE is the only key for relation R.

Each key consists of attribute C and E since no RHS of an FD contains C and E. Also, since $CE^+ = ABCDEF$, CE is the minimal superkey, thus the key. And also, since any combination of another attribute and CE has more attributes than simply CE, then there is no other keys.

   (b) Do the given FDs form a minimal basis? Prove or disprove.

Solution.
The given FDs form a minimal basis.

Proof. To prove the given set is a minimal basis, run the algorithm Minimal_basis($\mathcal{S}$) which has an FDs set as the input and output the minimal basis form of $\mathcal{S}$. If the output is the same as the input, then $\mathcal{S}$ is a minimal basis.
Thus execute this algorithm and take the set $\mathcal{S}$ formed by the given FDs.

Step 1: split RHS for each FD. Denote the new set as S_1.
No RHS of any FD can be split since each of them is singleton.

Step 2: Reduce the LHS of each FD if applicable. Denote the new set as S_2.

(a) $A \to B$: not reducible since the LHS is singleton.
(b) $CD \to A$: $C^+ = C$ & $D^+ = D$, which means, in fact, no singleton of this LHS yields anything. Thus the LHS of this FD is not reducible.
(c) $CB \to D$: $B^+ = B$, which means no singleton of this LHS yields anything. Thus the LHS of this FD is not reducible.
(d) $CE \to D$: $E^+ = E$, which means no singleton of this LHS yields anything. Thus the LHS of this FD is not reducible.
(e) $AE \to F$: $AE^+ = ABE$, which means AE does not yield F, thus not reducible.

Step 3: Try to eliminate each FD. Denote the new set as S_3.

(a) ${A^+}_{S_2-(a)} = A$. We need this FD.
(b) ${CD^+}_{S_2-(b)} = CD$. We need this FD.
(c) ${CB^+}_{S_2-(c)} = CB$. We need this FD.
(d) ${CE^+}_{S_2-(d)} = CE$. We need this FD.
(e) ${AE^+}_{S_2-(e)} = ABE$. We need this FD.

Through the whole algorithm, no FD in $\mathcal{S}$ is ever reduced, hence $S_3 = \mathcal{S}$. Since S_3 is generated by the algorithm Minimal_basis($\mathcal{S}$), S_3 is a minimal basis for R, and therefore $\mathcal{S}$ is a minimal basis for R.

■

   (c) Provide a decomposition of R into 3NF-satisfying relations.

Solution.
Let $\mathcal{S}$ be the set of FDs given in the problem and $L = \{A, B, C, D, E, F\}$, a set of attributes in R. Run the algorithm 3NF_synthesis($\mathcal{S}, L$) to decompose R into 3NF-satisfying relations.

Step 1: Verify if $\mathcal{S}$ is a minimal basis (The fact that $\mathcal{S}$ is a minimal basis has been proved in 1(b)).

Step 2: Union X, Y for each $X \to Y \in \mathcal{S}$ to define a new relation.
New relations derived are listed below:

$$R1(A, B)$$
$$R2(A, C, D)$$
$$R3(B, C, D)$$
$$R4(C, D, E)$$
$$R5(A, E, F)$$

Step 3: Check if there is any new relation a superkey for relation R, if not, add a relation with a schema as a key for the relation R.
CE is a key by of R by 1(a), thus R4 is a superkey for R. There is no need to add a new relation whose schema is the key for R.

Step 4: Return the final schema produced by the 3NF algorithm.
The final schema returned by this algorithm is:

$$R1(A, B)$$
$$R2(A, C, D)$$
$$R3(B, C, D)$$
$$R4(C, D, E)$$
$$R5(A, E, F)$$

(d) Are any of the relations that you made in part (c) not in BCNF?

Solution.
Every new relation is in BCNF.

Let $\mathcal{S}$ be the set of FDs given in the problem and $L = \{A, B, C, D, E, F\}$, a set of attributes in R. First of all, project FDs in $\mathcal{S}$ on to all new relations. Also, let L_1, L_2, L_3, L_4, L_5 be attribute sets of new relations $R1, R2, R3, R4, R5$ respectively. Execute:

```
1: procedure DO_PROJECTION($\mathcal{S}$, $[L_1...L_5]$)
2:     $i \leftarrow 1$
3:     $NewFDSetArray \leftarrow Array[1...length([L_1...L_5])]$    ▷ contains new FD sets for new relations generated in 1(c)
4:     while $i \leq length([L_1...L_5])$ do
5:         $S_i \leftarrow Project(\mathcal{S}, L_i)$    ▷ project FDs in $\mathcal{S}$ on to each $L_i$
6:         $NewFDASetArray[i] = S_i$
7:         $i \leftarrow i + 1$
8:     end while
9:     return $NewFDSetArray$    ▷ return new FD sets generated
10: end procedure
```

And tables of projection are listed below.

I. $R1(A, B)$
$L_1 = \{A, B\}$

A	B	closure	FDs
✓		$A^+ = AB$	$A \to B$
	✓	$B^+ = B$	*Nothing*

Thus $S_1 = \{A \to B\}$, and A is the superkey of $R1$. Thus $R1$ is in BCNF.

II. $R2(A, C, D)$
$L_2 = \{A, C, D\}$

A	C	D	closure	FDs
✓			$A^+ = AB$	*Nothing*
	✓		$C^+ = C$	*Nothing*
		✓	$D^+ = D$	*Nothing*
✓	✓		$AC^+ = AC$	*Nothing*
✓		✓	$AD^+ = AD$	*Nothing*
	✓	✓	$CD^+ = ACD$	$CD \to A$

Thus $S_2 = \{CD \to A\}$, and CD is the superkey of $R2$. Thus $R2$ is in BCNF.

III. $R3(B, C, D)$
$L_3 = \{B, C, D\}$

B	C	D	closure	FDs
✓			$B^+ = B$	*Nothing*
	✓		$C^+ = C$	*Nothing*
		✓	$D^+ = D$	*Nothing*
✓	✓		$BC^+ = BCD$	$BC \to D$
✓		✓	$BD^+ = BD$	*Nothing*
	✓	✓	$CD^+ = ACD$	*Nothing*

Thus $S_3 = \{BC \to D\}$, and BC is the superkey of $R3$. Thus $R3$ is in BCNF.

IV. $R4(C, D, E)$
$L_4 = \{C, D, E\}$

C	D	E	closure	FDs
✓			$C^+ = C$	*Nothing*
	✓		$D^+ = D$	*Nothing*
		✓	$E^+ = E$	*Nothing*
✓	✓		$CD^+ = ACD$	*Nothing*
✓		✓	$CE^+ = CDE$	$CE \to D$
	✓	✓	$DE^+ = DE$	*Nothing*

Thus $S_4 = \{CE \to D\}$, and BC is the superkey of $R4$. Thus $R4$ is in BCNF.

V. $R5(A, E, F)$
$L_5 = \{A, E, F\}$

A	E	F	closure	FDs
✓			$A^+ = AB$	*Nothing*
	✓		$E^+ = E$	*Nothing*
		✓	$F^+ = F$	*Nothing*
✓	✓		$AE^+ = AEF$	$AE \to F$
✓		✓	$AF^+ = AF$	*Nothing*
	✓	✓	$EF^+ = EF$	*Nothing*

Thus $S_5 = \{AE \to F\}$, and AE is the superkey of $R5$. Thus $R5$ is in BCNF.

According to the projection result of all new relations, there is no new relation not in BCNF.

2. Answer the following questions.

   (a) Prove or disprove the following:
   Suppose a relation R is decomposed into R1 and R2 with one common attribute between the two new relations.
   If the common attribute between R1 and R2 forms a key for at least one of R1 or R2, then the decomposition is lossless.

Solution.
Yes, according to definition of lossless decomposition, we need to prove that at least one of the following functional dependencies are in F^+(where F^+ stands for the closure for every attribute or attribute sets in F):

1) $R1 \cap R2 \rightarrow R1$
2) $R1 \cap R2 \rightarrow R2$

Assume the common attribute in R1 and R2 is A, and A is key for at least one of R1 or R2. These imply that $R1 \cap R2 = A$. Also A is a superkey for at least one of R1 and R2, meaning we have a functional dependency $\{A \rightarrow \textit{other\_attributes\_in\_one\_relation}\}$. In other words, we have proven that $R1 \cap R2 \rightarrow R1$ or $R1 \cap R2 \rightarrow R2$ that $\in F^+$.

(b) Can a relation and a set of FDs be in both BCNF and 3NF at the same time? If so, explain what conditions must be met. If not, explain what is preventing this from being possible.

Solution.
Yes. If a relation R is in both BCNF and 3NF, then for every nontrivial FD $X \rightarrow Y$ which held in R, X is a superkey.
If a relation R is in BCNF, then for all nontrivial FDs in R, the LHS of FD is a superkey for R. Based on the property of R explained above, R satisfies this condition, thus R is in BCNF.
If a relation R is in 3NF, then for each FD $X \rightarrow A$, X is a superkey or A is prime. Since each FD of R has LHS as the superkey for R based on the properties of R, R satisfies the 3NF property. Therefore R is in 3NF.
Thus R is in both BCNF and 3NF.

3. Prove or disprove that:

(a) If $A \rightarrow B$ then $B \rightarrow C$

Solution.
No, this can be proved by a counterexample.

Proof. Build an instance of a relation $R(A, B, C)$ which has FD set $\{A \rightarrow B\}$ as below.

A	B	C
1	2	3
1	2	9

This instance satisfies all the FDs in the relation $R(A, B, C)$, and no tuples violate any FD. However, seen from the instance, C is not determined by B. Thus, $B \nrightarrow C$ even if $A \rightarrow B$ in this instance. Therefore, if only the FD $A \rightarrow B$ is known, it is not enough to state $B \rightarrow C$.

■

(b) If $AB \rightarrow C$ then $A \rightarrow C$ and $B \rightarrow C$

Solution.
No, this can be proved by a counterexample.

Proof. Build an instance of a relation $R(A, B, C, D)$ which has FD set $\{AB \rightarrow C\}$ as below.

A	B	C	D
1	2	3	4
1	2	3	5
1	3	2	6
2	2	7	8

This instance satisfies all the FDs in the relation $R(A, B, C, D)$, and no tuples violate any FD in this instance. However, seen from the instance, C is not determined by A from tuple 1 and 3, and C is not determined by B from tuple 1 and 4. Thus, $A \nrightarrow C$ and $B \nrightarrow C$ even if $AB \rightarrow C$ in this instance. Therefore, if only the FD $AB \rightarrow C$ is known, it is not enough to state $A \rightarrow C$ and $B \rightarrow C$.

■

4. **Design and DDL** Consider the following domain: You are running a Fresh Juice business with multiple stores around the country, and you want to keep the information for these stores in a relational database. The following is a list of that information:

- Stores: Each store has a city, phone number, and manager. There is only one store per city.
- Beverages: Each juice beverage has a name (e.g. Kiwi Lime), and a number of calories for a regular and large size. A large size always has 200 more calories than the regular size. Every store should keep track of the number of inventory of each beverage (how much it has left in stock).
- Transactions: When a customer makes an order, that order should have a date, price, and an indication of which loyalty card was used, if applicable. You can assume one beverage is ordered per transaction, and we should know what that beverage was.
- Loyalty card: Customers can have a loyalty card if they like to go to your stores a lot. There needs to be information on how many transactions a customer made with the card, and their home store (the one they go to most frequently).

(a) Define a **single** relation for this domain that manages to store all of the required information (just write it out *R(...)*, no need for SQL definitions yet). There is not necessarily one correct answer for this relation, but the information should be stored in a practically useful way. It is ok to add attributes that aren't explicitly listed in the domain as long as they are useful.

Solution.
$R(Store\_id, City, phone\_number, manager, transaction\_id, tran\_date, tran\_price,$
$loyalty\_card\_id, card\_home\_store, number\_of\_tran\_in\_card,$
$beverage\_id, name\_of\_beverage\_sold, beverage\_regular\_calories,$
$beverage\_large\_calories, amount)$

(b) Write all of the functional dependencies for your relation that would be inferred by the description of this domain. Do not include trivial or redundant FDs (find a minimal basis).

Solution.
Store_id → City, Store_id → phone_number, Store_id → manager,
transaction_id → tran_date, transaction_id → tran_price, transaction_id → loyalty_card_id,
transaction_id → name_of_beverage_sold
loyalty_card_id → number_of_tran_in_card, loyalty_card_id → card_home_store,
beverage_id → name_of_beverage_sold, beverage_id → beverage_regular_calories,
beverage_id → beverage_large_calories,
store_id, beverage_id → amount

(c) Provide a useful instance of your relation that shows all three types of anomalies. Describe the anomalies you have presented as they appear in your particular instance (give an example for each of the three anomalies in your relation).

Solution.

Store_id	City	phone_number	manager	transaction_id	tran_date	tran_price
1	Toronto	111-111-1111	Peter	1	01/11	10
2	Vancouver	222-222-2222	Daniel	2	01/11	20
3	Ottawa	333-333-3333	James	3	01/11	15
1	Toronto	111-111-111	Peter	4	01/11	5
1	Toronto	111-111-111	Peter	5	01/11	18

continue with

loyalty_card_id	card_home_store	number_of_tran_in_card	beverage_id	name_of_beverage_sold
123456	1	4	123	orange
234567	1	5	456	apple
123456	1	4	789	banana
345678	3	6	101112	grape
456789	2	10	121314	watermelon

continue with

beverage_regular_calories	beverage_large_calories	amount
100	300	50
100	300	40
300	500	30
200	400	20
150	350	40

Redundancy: Lots of duplicate information for store with Store_id 1 in Toronto tuple.
Update: Change number_of_tran_in_card for loyalty_card with loyalty_card_id 123456 in one tuple requires updating all tuples with loyalty_card_id 123456.
Deletion: Delete 4 as a transaction_id can remove the tuple with loyalty_card_id 345678 entirely.

(d) Your relation will likely (read certainly) have some redundancy. Decompose your relation into a set of relations without any BCNF violations. Write all of your steps in full and clearly show why your relations do not violate BCNF.

Solution.
Let R be the original relation and F be the set of all FDs. In order to decompose this relation to all new relations that does not have any BCNF violations, execute the algorithm BCNF_decomp(R, F).

Step 0. run BCNF_decom(R, F).
Detail: find one LHS of FD in F, `(store_id, beverage_id)`, is not a superkey, thus the FD with `(store_id, beverage_id)`, is not in BCNF.
Let

$$R_0 = (store\_id, beverage\_id)^+ = (store\_id, beverage\_id, amount)$$
$$R_{0.5} = R - ((store\_id, beverage\_id)^+ - (store\_id, beverage\_id)) = (store\_id, city, phone\_number, manager, transaction\_id, tran\_date, tran\_price, loyality\_card\_id, beverage\_id, name\_of\_beverage\_sold, number\_of\_tran\_in\_card, card\_home\_store, beverage\_regular\_calories, beverage\_large\_calories)$$

Then project FDs onto R_0 and $R_{0.5}$. R_0 is in BCNF but $R_{0.5}$ not in BCNF.

Step 1. run BCNF_decom($R_{0.5}$, $F_{0.5}$).
Detail: find one LHS of FD in $F_{0.5}$, `store_id`, is not a super key, thus the FD with `store_id`, is not in BCNF.
Let

$$R_1 = store\_id^+ = (store\_id, city, phone\_number, manager)$$
$$R_2 = R - (store\_id^+ - store\_id) = (store\_id, transaction\_id, tran\_date, tran\_price, loyality\_card\_id, beverage\_id, name\_of\_beverage\_sold, number\_of\_tran\_in\_card, card\_home\_store, beverage\_regular\_calories, beverage\_large\_calories)$$

Then project FDs onto R_1 and R_2. R_1 is in BCNF but R_2 not in BCNF.

Step 2. run BCNF_decom(R_2, F_2), F_2 is the set of FDs projected on R_2 from F.
Detail: find one LHS of FD in F_2, `transaction_id`, is not a super key, thus the FD with `transaction_id`, is not in BCNF.
Let

$$R_3 = transaction\_id^+ = (transaction\_id, tran\_date, tran\_price, loyality\_card\_id, beverage\_id$$
$$name\_of\_beverage\_sold, number\_of\_tran\_in\_card, card\_home\_store,$$
$$beverage\_regular\_calories, beverage\_large\_calories)$$
$$R_4 = R_2 - (transaction\_id^+ - transaction\_id) = (store\_id, transaction\_id)$$

Then project FDs onto R_3 and R_4. R_4 is in BCNF but R_3 not in BCNF.

Step 3. run BCNF_decom(R_3, F_3), F_3 is the set of FDs projected on R_3 from F_2.
Detail: find one LHS of FD in F_3, `loyality_card_id`, is not a super key, thus the FD with `loyality_card_id`, is not in BCNF.
Let

$$R_5 = loyality\_card\_id^+ = (number\_of\_tran\_in\_card, card\_home\_store, loyality\_card\_id)$$
$$R_6 = R_3 - (loyality\_card\_id^+ - loyality\_card\_id) = (transaction\_id, tran\_date, tran\_price,$$
$$loyality\_card\_id, beverage\_id, name\_of\_beverage\_sold, beverage\_regular\_calories,$$
$$beverage\_large\_calories)$$

Then project FDs onto R_5 and R_6. R_5 is in BCNF but R_6 not in BCNF.

Step 4. run BCNF_decom(R_6, F_6), F_6 is the set of FDs projected on R_6 from F_3.
Detail: find one LHS of FD in F_6, `beverage_id`, is not a superkey, thus the FD with `beverage_id`, is not in BCNF.
Let

$$R_7 = beverage\_id^+ = (beverage\_id^+, name\_of\_beverage\_sold,$$
$$beverage\_regular\_calories, beverage\_large\_calories)$$
$$R_8 = R_6 - (beverage\_id^+ - beverage\_id) = (transaction\_id,$$
$$tran\_date, tran\_price, loyality\_card\_id, beverage\_id)$$

Then project FDs onto R_7 and R_8. R_7 and R_8 are both in BCNF.

Step 4. run BCNF_decom(R_6, F_6), F_6 is the set of FDs projected on R_6 from F_3.
Detail: find one LHS of FD in F_6, `beverage_id`, is not a superkey, thus the FD with `beverage_id`, is not in BCNF.
Let

Finally, the new relations are

$$R_0(store\_id, beverage\_id, amount)$$
$$R_1(city, phone\_number, manager, store\_id)$$
$$R_4(transaction\_id, store\_id)$$
$$R_5(card\_home\_store, number\_of\_tran\_in\_card, loyality\_card\_id)$$
$$R_7(beverage\_id, name\_of\_beverage\_sold, beverage\_regular\_calories, beverage\_large\_calories)$$
$$R_8(transaction\_id, tran\_date, tran\_price, loyality\_card\_id, name\_of\_beverage\_sold)$$

(e) Explain how your new relations prevent the anomalies you pointed out in part (c).

Solution.
Redundancy: City is suerkey in R1 and one store per city, therefore, it will no duplicate information for Store_id 1 in Toronto tuple.
Update: loyality_card_id is superkey in R5, therefore, there is only one tuple whose card is 123456, only change number_of_tran_card for 123456 once.

Deletion: When we delete 4 as transaction_id, it will delete one tuple form R4 and R8 respectively and we will preserve loyalty_card_id 345678 in R5.

(f) **SQL DDL:**
Using your new relations from part (d), create a schema using the SQL DDL language. They should have proper relation names, attribute names and types, and constraints (including keys, foreign key, unique, not null, etc.).
You should add **comments** above each table and attribute describing what it represents. You can add comments using a double dash `--`. You should also insert some useful data into each of the relations (directly in the ddl file, not through csv).
Use the DDL files from the lectures and A2 as examples to help you make them. Unlike A2, you do not need to write any SQL queries for this part.

Solution.
Our schema is in the file `fruits.ddl` with some insertions for our demo,
and the demo is in the file `fruits-demo.txt`.

Description on all constraints:

- Stores
  - `not null:` a store should have a city, phone_number and manager, thus we think they should not be null.
  - `oneCityOnly:` each store is in only one city, thus check for uniqueness of city.
- Loyality_Cards
  - `not null:` card_home_store and number_of_tran_in_card are not null since for each loyality_card, the store which they go most often to get the card and how many transactions (how often) this card is used must be recorded to get the card.
  - `storeOfCard:` reference the home store to the info of the store.
- Beverages
  - `not null:` name_of_beverage_sold, beverage_regular_calories, beverage_large_calories, beverage_regular_stock, beverage_large_stock are not null, since these are basic data of a beverage type and used to tell the name of a beverage and distinguish regular and large beverage.
  - `twoHundredMore:` the calories of the same type large size beverage is 200 more than regular one.
  - `oneName:` each beverage involves one name.
- Inventory
  - `not null:` beverage_id and amount should not be null since if there is a store, some beverages should be stocked in some amount.
  - `oneInventoryOneStore:` one store has one store_id in this table.
  - `oneStockOnebeverage:` one beverage has one number.
- TransactionInfo
  - `not null:` beverage_id should not be null because there is always some beverage sold per transaction.
  - `cardUsed:` the loyality card used in transaction if applicable, reference it to the info of a loyality_card.
  - `beverageInTransaction:` reference the beverage sold in this transaction to the info of a beverage.
- TransactionsOfStore
  - `not null:` the foreign key reference store_id is not null, because one transaction must happen in a store.
  - `validTransaction:` reference a valid transaction to the transactionInfo table to find info about the transaction.

- `validStore:` reference the store of the transaction to the stores table to find info about the store of this transaction.

What to hand in for this part: In your A3.pdf file, describe the decisions for any constraints you put in your DDL file. Hand in a file called `fruits.ddl` containing your schema, as well as a plain text file called `fruits-demo.txt` that shows you starting postgreSQL, successfully importing fruits.ddl, and exiting posgreSQL. This is similar to what you did in the preps. You must hand in this demo and the file must be a plain text file or you get **zero** for this part of the assignment.

Submission instructions

Your assignment must be typed; handwritten assignments will not be marked. You may use any word- processing software you like.
For this assignment, hand in a file A3.pdf that contains your answers to the questions above. Also hand in `fruits.ddl` and `fruits-demo.txt`.
You must declare your team and hand in your work electronically using the MarkUs online system. Well before the due date, you should declare your team and try submitting with MarkUs.