CSC343 A3

Haider Sajjad, Muskan Patpatia

August 4, 2019

Database Design and SQL DDL

1. a) The only key in R is CE.

   b) Lets assume the given FD's do not form a minimal basis. Make a set S to hold the FD's in minimal cover.

   The set of FD's are: $\{A \to B, CD \to A, CB \to D, CE \to D, AE \to F\}$

   To find minimal basis we first start with reducing left hand side of every FD X $\to$ Y where $|X| \geq 2$ in the set. Let's make a set L to hold these FD's:

   CD $\to$ A
   $C^+ = C$, and $D^+ = D$
   So L = {CD $\to$ A}

   CB $\to$ D
   $C^+ = C$, and $B^+ = B$
   So L = {CD $\to$ A, CB $\to$ D}

   CE $\to$ D
   $C^+ = C$, and $E^+ = E$
   So L = {CD $\to$ A, CB $\to$ D, CE $\to$ D}

   AE $\to$ F
   $A^+ = AB$, and $E^+ = E$
   So L = {CD $\to$ A, CB $\to$ D, CE $\to$ D, AE $\to$ F}

   Next Let's remove any redundant FD's, first lets look at the FD's in L, if they are not redundant store them in S, as they are needed:

   CD $\to$ A, pretend this doesn't exist
   Then, $CD^+ = CD$, so we need it
   So S = {CD $\to$ A}

   CB $\to$ D, pretend this doesn't exist
   Then, $CB^+ = CB$, so we need it
   So S = {CD $\to$ A, CB $\to$ D}

   CE $\to$ D, pretend this doesn't exist
   Then, $CE^+ = CE$, so we need it

So S = {CD → A, CB → D, CE → D}

CE → D, pretend this doesn't exist
Then, $CE^+ = CE$, so we need it
So S = {CD → A, CB → D, CE → D}

AE → F, pretend this doesn't exist
Then, $AE^+ = ABE$, so we need it
So S = {CD → A, CB → D, CE → D, AE → F}

Next look at the rest of the FD's:

A → B, pretend this doesn't exist
Then, $A^+ = A$, so we need it
So S = {A → B, CD → A, CB → D, CE → D, AE → F}

Now S is the set of FD's in minimal basis, but S = the set of given FD's.
Contradiction, the given FD's form a minimal basis.

c)
Minimal basis of the FD's are: $\{A \to B, CD \to A, CB \to D, CE \to D, AE \to F\}$

Using these FD's we get relations: $R_1(\underline{\mathbf{A}}B), R_2(\underline{\mathbf{CD}}A), R_3(\underline{\mathbf{CB}}D), R_4(\underline{\mathbf{CE}}D), R_5(\underline{\mathbf{AE}}F)$
Where the underlined letters are the keys.
The key for this schema is R_4 because it has the key for R.

d)
No, all the relations in part c) are in BCNF because every FD in a relation has its left hand side be the superkey for that relation.

2. a)

This is true, here is a direct proof.

Assume R is a relation decomposed into relations R_1 and R_2. And assume $R_1 \cap R_2 = A$, so they share a single attribute A, and lets assume A is a key for either R_1 or R_2.

We want to prove that $R_1 \bowtie R_2 = R$, proving the decomposition is lossless.

First, notice that $R \subseteq R_1 \bowtie R_2$. This is because R_1 and R_2 are decompositions of R, joining them over a shared attribute A will give a cross product of the two relations and every attribute of R will be in $R_1 \bowtie R_2$ because a cross product gives us every combination.

Next, we will prove that $R_1 \bowtie R_2 \subseteq R$.

First recall $R_1 \cap R_2 = A$, where A a key in either R_1 or R_2. This means that when natural joining R_1 and R_2, the relation with A as its key, it's tuples will appear once for every appearance of A in the join. So $R_1 \bowtie R_2 \subseteq R$ because joining over a key means no extra tuples get formed, and $|R_1 \bowtie R_2| = |R|$.

Since $R \subseteq R_1 \bowtie R_2$ and $R_1 \bowtie R_2 \subseteq R$ then $R_1 \bowtie R_2 = R$. So the decomposition is lossless.

b)
Yes, if a relation is in BCNF it is guaranteed to be in 3NF. This is because since R is in BCNF the left hand side of every FD in R will be a superkey, satisfying the 3NF requirement.

However a relation being in 3NF is not guaranteed to be in BCNF, because a relation can still be in 3NF and not satisfy the requirement that every FD's left hand side is a superkey.

3. a) This is false.
Take this relation for example:

A	**B**	**C**
a	b	c
a1	b	c

The functional dependency $A \rightarrow B$ holds because, $a \rightarrow b$ and $a1 \rightarrow b$ both hold.
But, $b \rightarrow c$ and $b \rightarrow c1$ means the functional dependency $B \rightarrow C$ does not hold.

So, $A \rightarrow B$ does not imply $B \rightarrow C$.

b) This is false.
Take this relation for example:

A	**B**	**C**
a	b	c
a	b1	c1
a1	b	c2

The functional dependency $AB \rightarrow C$ holds because $(a, b \rightarrow c)$ and $(a, b1 \rightarrow c1)$ and $(a1, b \rightarrow c2)$ all hold.

However, since $(a \rightarrow c)$ and $(a1 \rightarrow c)$, $A \rightarrow C$ does not hold.
Also, since $(b \rightarrow c)$ and $(b \rightarrow c2)$, $B \rightarrow C$ does not hold.

So, $AB \rightarrow C$ does not imply $A \rightarrow C$ and $B \rightarrow C$.

4. In this question, bname = "beverage name" and bsize = "beverage size". Also denote tid to be "transaction id" and cid is "customer id".

a) R(city, phone, manager, bname, instock, calories, date ,price, loyalty, bsize, tid, numTransactions, home_store, cid)

b)
The functional dependencies are as follows:

city $\rightarrow$ phone, manager
city, bname, bsize $\rightarrow$ instock
bname, bsize $\rightarrow$ calories
tid $\rightarrow$ date, loyalty, cid, bname, bsize, city
bname, bsize, loyalty $\rightarrow$ price
loyalty, cid $\rightarrow$ numTransactions, home_store
R[home_store] $\subseteq$ R[city]

c)

city	phone	manager	bname	instock	calories	date	price	loyalty	bsize	tid	numTransactions	home_store	cid
Toronto	403	Tom	apple	2	200	2019-05-05	4.0	111	r	1	4	Toronto	10044
Toronto	403	Tom	apple	1	200	2019-05-06	4.0	112	r	1	14	Toronto	10045
Chicago	905	Henry	apple	8	400	2019-05-05	4.0	111	l	1	4	Toronto	10044

The redundancy anomaly here would be that there are multiple tuples with duplicate information with city Toronto. For every transaction in the same city, we will get back data we already know and don't need.

The deletion anamoly would be that if i delete manager Henry for Chicago then the complete information about the store chicago will be deleted as well.

The update anamoly would be that if i update manager Tom for Toronto to another manager name then i will have to update other tuples for Toronto as well.

d) First, to find the key of R, we will find the closure of every subset of attributes in R:

$city^+$ = city, phone, manager
$city, bname, bsize^+$ = city, bname, bsize, phone, manager, instock
$bname, bsize^+$=calories
tid^+=date, loyalty, cid, bname, bsize, city, phone, manager, price, calories, instock, numTransactions, home_store
$bname, bsize, loyalty^+ = bname, bsize, loyalty, price, calories$
$loyalty, cid^+ = loyalty, cid, numTransactions, home_store$

Therefore key is tid since all the attributes of relation R are in its closure.

BCNF process:

city → phone, manager
city is not the key therefore we need to decompose into two relations.
S1(city,phone,manager)
S2(city, bname, instock, calories, date, price, loyalty, bsize, tid, numTransactions, home_store, cid)
For S1 city is the key and therefore S1 is in bcnf.
This FD:
city, bname, bsize → instock
Is in S2, and tid is the key in S2. This FD fails to contain it in it's left hand side. So S2 is not in BCNF.
Now decompose S2 into:

S3(city, bname, bsize, instock)
S4(city, bname, bsize, calories, date, price, loyalty, tid, numTransactions, home_store,cid)
For S3 (city, bname, bsize) is the key and the projected fd is city,bname,bsize → instock hence S3 is in bcnf.
For S4 key is tid and the following projected fd:
bname,bsize → calories
violates bcnf because it doesn't contain tid. Therefore we need to decompose S4 further.

S5(bname,bsize,calories)
S6(city, bname, bsize, date, price, loyalty, tid, numTransactions, home_store, cid)
In S5 the key is bname, bsize and only projected fd bname, bsize $\rightarrow$ calories satisfies bcnf. Hence S5 is in bcnf.
Now for S6 we have key is tid and the following prjected fds:

bname, bsize, loyalty $\rightarrow$ price
This FD violates bcnf. Hence lets decompose S6 further.

S7(bname, bsize, loyalty,price)
S8(city, bname, bsize, date, loyalty, tid, numTransactions, home_store, cid)
Now for S7 key is bname, bsize, loyalty and the only projected fd bname ,bsize, loyalty $\rightarrow$ price satisfies bcnf. hence S7 is in bcnf.
Now for S8 key is tid and the following projected fd
loyalty,cid $\rightarrow$ numTransactions,home_store
violates bcnf property. Hence S8 needs to be decomposed further.

S9(loyalty, cid, numTransactions, home_store)
S10(city, bname, bsize ,date, loyalty, tid, cid)
For S9 key is loyalty, cid and the only projected fd is loyalty, cid $\rightarrow$ numTransactions ,home_store and this fd satisfies bcnf property. hence S9 is in bcnf.
For S10 key is tid and only one FD fits in it.
tid $\rightarrow$ date,loyalty,cid,bname,bsize,city
Since the key is tid and the projected fd satisfies bcnf property. Therefore S10 is is in bcnf.

All relations after renaming to appropriate names:

juice_stock(<u>city, bname, bsize</u>, instock)
store(<u>city</u>,phone,manager)
beverages(<u>bname, bsize</u>, calories)
beverage_price(<u>bname, bsize, loyalty</u>, price)
loyalty_card(<u>loyalty, cid</u>, numTransactions, home_store)
transactions(<u>tid</u>, city, bname, bsize, date, loyalty, cid)

e)