CSC343 A3

Haider Sajjad, Muskan Patpatia

August 3, 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 $A \to B$ holds because, $a \to b$ and $a1 \to b$ both hold.
But, $b \to c$ and $b \to c1$ means the functional dependency $B \to C$ does not hold.

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

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

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

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

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

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

4.