CS 1555 / 2055 – Database Management Systems (Fall 2020)
Dept. of Computer Science, University of Pittsburgh

Assignment #6: Database Design - Normalization (Solution)

Release: Nov. 3, 2020 Due: 8:00 PM, Nov. 12, 2020.

---

1. [30 points] Consider the relation R(A,B,C,D,E,F). Use **synthesis method** to construct a set of 3NF relations from the following functional dependencies. Indicate the primary key for each relation in the result.

   $AB \rightarrow E$

   $B \rightarrow ED$

   $E \rightarrow D$

   $DF \rightarrow A$

   $C \rightarrow F$

   $DC \rightarrow A$

   Solution:

   (a) Find the canonical cover of F:

      i. Transform all FDs to canonical form (i.e., one attributes on the right):
         $AB \rightarrow E$
         $B \rightarrow E$
         $B \rightarrow D$
         $E \rightarrow D$
         $DF \rightarrow A$
         $C \rightarrow F$
         $DC \rightarrow A$

      ii. Drop extraneous attributes: A in $AB \rightarrow E$ is extraneous, since we have $B \rightarrow E$.
         The complete proof is as follows:
         Considering the minimality of LHS of $AB \rightarrow E$
         First we need to compute: A+ using minimal cover
         A+ = A (Initialization)
         = A
         Done.
         => E is not subset of A+. So attr B is necessary.
         Then, we need to compute: B+ using minimal cover
         B+ = B (Initialization)
         = BD ($B \rightarrow D$)
         = BDE ($B \rightarrow E$)
         ...
         Done.
         => E is a subset of (AB - A)+. So A is NOT necessary

The set of FDs becomes:

$B \rightarrow E$
$B \rightarrow D$
$E \rightarrow D$
$DF \rightarrow A$
$C \rightarrow F$
$DC \rightarrow A$

iii. Drop (transitive) redundant FDs:
$B \rightarrow E$ and $E \rightarrow D$ implies $B \rightarrow D$, so we drop $B \rightarrow D$.
$C \rightarrow F$ implies $CD \rightarrow FD$. We also have $DF \rightarrow A$, so we drop $DC \rightarrow A$.
The set of FDs becomes:

$B \rightarrow E$
$E \rightarrow D$
$DF \rightarrow A$
$C \rightarrow F$

which is the canonical cover of F.

(b) Find the primary key of R: Observations:

i. B and C do not appear on the right hand side of any FD, so they have to appear in all keys of R.

ii. BC+ : $BC \rightarrow BCE$ (since $B \rightarrow E$) $\rightarrow BCED$ (since $E \rightarrow D$) $\rightarrow BCEDF$ (since $C \rightarrow F$) $\rightarrow BCEDFA$ (since $DF \rightarrow A$). So BC is a key of R.
In this case, we do not need to consider any other combination, because any other combination containing BC (e.g., BCD) is a super key and not minimal.

(c) We do not need to group the FDs in the canonical cover because all the determinants on the left side are unique.

(d) Construct a relation for each group:

R1 ($\underline{B}$, E)
R2 ($\underline{E}$, D)
R3 ($\underline{D, F}$, A)
R4 ($\underline{C}$, F)

(e) If none of the relations contain the key for the original relation, add a relation with the key:

R5 ($\underline{B, C}$)

R1, R2, R3, R4, and R5 are in 3NF and in BCNF.

2. [30 points] Consider the relation R(A,B,C,D) and the following set of functional dependencies F. Apply the **decomposition method** on R to end up with BCNF relations and dependency preserving. Indicate the primary key for each relation in the result.

$A \rightarrow B$

$B \rightarrow CD$

$A \rightarrow D$

$B \rightarrow C$

$AB \rightarrow CD$

Solution:

(a) Find the canonical cover of F:

A → B

B → D

B → C

(b) Apply the decomposition method on R to end up with BCNF relations:

i. Using A → B to decompose R, we can get:

R1′ (<u>A</u>, C, D) in BCNF

R2′ (<u>A</u>, B) in BCNF

Note that this decomposition does not preserve dependencies B → D and B → C, so we choose another dependency and try again from the start.

ii. Using B → D to decompose R, we can get:

R1 (<u>A</u>, B, C) in 2NF

R2 (<u>B</u>, D) in BCNF

iii. Using B → C to decompose R1, we can get:

R11 (<u>A</u>, B) in BCNF

R12 (<u>B</u>, C) in BCNF

A correct decomposition would be R2, R11 and R12. R2, R11 and R12 are in BCNF and dependency preserving. An efficient one that eliminates an unnecessary join is :

T1 (<u>A</u>, B)

T2 (<u>B</u>, C, D)

where we group R2 and R12 since they share the same primary key. T1 and T2 are in BCNF and dependency preserving.

3. [40 points] Using the table method, check if the following decomposition is good, bad or ugly. Show all steps.

R1: (<u>ProductID</u>, Length, Width, Height, Weight, <u>OrderID</u>, OrderDate, CustomerID, Total-Price)
R2: (<u>CustomerID</u>, Address, City, State, ZipCode, PhoneNumber)
R3: (<u>ProductID</u>, <u>OrderID</u>, ProductQuantity)

Assume the functional dependency set to be:
FD1: ProductID → Length, Width, Height, Weight
FD2: OrderID → OrderDate, CustomerID, TotalPrice
FD3: CustomerID → Address, City, State, ZipCode, PhoneNumber
FD4: ProductID, OrderID → ProductQuantity

Hint: bad decomposition is a lossy one, while ugly decomposition is lossless but does not preserve some dependencies.

Solution:

Let the attributes be sorted in the following order:
(1) ProductID, (2) Length, (3) Width, (4) Height, (5) Weight, (6) OrderID, (7) OrderDate, (8) CustomerID, (9) TotalPrice, (10) Address, (11) City, (12) State, (13) ZipCode, (14) PhoneNumber, (15) ProductQuantity

Initially the table looks like the following. Note that the table uses simplified marks. "k" means known cell, and empty cell means "U".

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R1	k	k	k	k	k	k	k	k	k						
R2								k		k	k	k	k	k	
R3	k					k									k

Using FD1, we can add more "k" marks in the table. New marks are in italic uppercase.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R1	k	k	k	k	k	k	k	k	k						
R2								k		k	k	k	k	k	
R3	k	*K*	*K*	*K*	*K*	k									k

Then we use FD2 to update the table. New marks are in italic uppercase.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R1	k	k	k	k	k	k	k	k	k						
R2								k		k	k	k	k	k	
R3	k	k	k	k	k	k	*K*	*K*	*K*						k

Next, we use FD3 to update the table. New marks are in italic uppercase.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R1	k	k	k	k	k	k	k	k	k	*K*	*K*	*K*	*K*	*K*	
R2								k		k	k	k	k	k	
R3	k	k	k	k	k	k	k	k	k	*K*	*K*	*K*	*K*	*K*	k

Finally we use FD4 to update the table. New marks are in italic uppercase.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R1	k	k	k	k	k	k	k	k	k	k	k	k	k	k	*K*
R2								k		k	k	k	k	k	
R3	k	k	k	k	k	k	k	k	k	k	k	k	k	k	k

Now we have 2 rows filled with mark "k", so it is a lossless decomposition. Since it preserves all FDs, it is a good decomposition.