CS 1555 - Database Management Systems

Assignment 6: Database Design - Normalization

Gordon Lu, PSID: 4191042

Question 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:
First, let's find the canonical form:

- Transform all functional dependencies to canonical form (one attribute on the right):

$$AB \rightarrow E$$

$$B \rightarrow E$$

$$B \rightarrow D$$

$$E \rightarrow D$$

$$DF \rightarrow A$$

$$C \rightarrow F$$

$$DC \rightarrow A$$

When we transform to canonical form, we do not consider the functional dependency B → ED, since there are two attributes on the right. So instead, we decompose this functional dependency into B → E and B → D.

- Now, we drop any extraneous attributes:

A in AB→ E is extraneous, since we already have B → E. So, the set of functional dependencies becomes:

B → E

B → D

E → D

DF → A

C → F

DC → A

- Now, we drop redundant functional dependencies:

Through the transitive rule from Armstrong's Axioms, B → E and E → D implies B → D, so we drop B → D.

Additionally, through the pseudotransitivity and composition rule, C → F and DF → A implies DC → A, so we drop DC → A.

The set of functional dependencies then becomes:

B → E

E → D

DF → A

C → F

Now, let's find the keys of R:

Observations:

- Note that B and C do not appear on the right-hand side of any functional dependencies, so they have to appear in all keys of R.
- BC+: BC → BCE (since B → E) → BCED (since E → D) → BCEDA (since DF → A) → BCEDAF (since C → A). In any case, we do not need to consider any other combination, since any other combination containing BC is a super key and not minimal.

With this key in mind, we can now use the Synthesis Method to construct a set of 3NF relations.

1) Starting from the canonical cover, we can start with grouping functional dependencies with the same determinant:

- In this case, none of the functional dependencies have the same determinant, so we can leave the functional dependencies as is.

2) Now, we construct a relation for each group:
R1($\underline{B}$, E)

R2($\underline{E}$, D)

R3($\underline{C}$, F)

R4($\underline{D}$,$\underline{F}$,A)

3) Now, check if any of the relations contain the key of the original relation, and add if not present.

- Since the key, **BC**, is not present in any of the relations, we add another relation with B, C:

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

Now, the relations R1, R2, R3, R4, and R5 are in 3NF.

Question 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:

First, let's find the canonical form:

- Transform all functional dependencies to canonical form (one attribute on the right):

$$A \rightarrow B$$

$$B \rightarrow C$$

$$B \rightarrow D$$

$$A \rightarrow D$$

$$B \rightarrow C$$

$$AB \rightarrow C$$

$$AB \rightarrow D$$

When we transform to canonical form, we do not consider the functional dependency B → CD, since there are two attributes on the right. So instead, we decompose this functional dependency into B → C and B → D. The same applies to the functional dependency AB → CD since there are two attributes on the right. So, we decompose this functional dependency into AB → C and AB → D.

- Now, we drop any extraneous attributes:

Both A and B in AB→ D are extraneous, since we already have A → D and B → D. Additionally, note that we have two instances of B → C, so can remove one of the instances. Also, note that A in AB → C is extraneous, since we already have B → C. So, the set of functional dependencies becomes:

$$A \rightarrow B$$

$$B \rightarrow C$$

$$B \rightarrow D$$

$$A \rightarrow D$$

- Now, we drop redundant functional dependencies:

Through the transitive rule from Armstrong's Axioms, A → B and B → D implies A → D, so we drop A → D.

The set of functional dependencies then becomes:

$$A \rightarrow B$$

$$B \rightarrow C$$

$$B \rightarrow D$$

Now, let's find the keys of R:

Observations:

- Note that A does not appear on the right-hand side of any functional dependencies, so they have to appear in all keys of R.
- A+: A → AB (since A → B) → ABC (since B → C) → ABCD (since B → D). In any case, we do not need to consider any other combination, since any other combination containing A is a super key and not minimal.

Apply the Decomposition Method on R:

1) Using our canonical cover, and key, let's use A → B to decompose R, to get:
R1(<u>A</u>,C,D) is in BCNF form

R2(<u>A</u>,B) is in BCNF form

However, R1 and R2 do not preserve the functional dependency B → D. So, we must start over with the functional dependency B → C instead.

- Keep B in here, since we are going to be using it later in the decomposition

2) Using B → C to decompose R, we get:

R11(<u>A</u>,B,D) is in 2NF

- R11 is in 2NF since we have no partial dependencies, and each value is atomic.

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

3) Using B → D to decompose R11, we get:
R111(<u>A</u>,B) is in BCNF form

R112(<u>B</u>,D) is in BCNF form

- There are no transitive dependencies and satisfies 2NF.

Group the relations with the same key:
R1(A,B)

R2(B,C,D)

R1, R2 are in BCNF form.

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

R1: (ProductID, Length, Width, Height, Weight, OrderID, OrderDate, CustomerID, TotalPrice)

R2: (CustomerID, Address, City, State, ZipCode, PhoneNumber)

R3: (ProductID, OrderID, 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.

For ease, let's abbreviate all of the attributes like so:
ProductID → PID

Length → L

Width → W

Height → H

Weight → WGT

OrderID → OID

OrderDate → ODate

CustomerID → CID

TotalPrice → TP

Address → A

City → C

State → S

ZipCode → ZC

PhoneNumber → PH

ProductQuantity → PQ

First, let's construct an mxn table:

	PID	L	W	H	WGT	OID	ODate	CID	TP	A	C	S	ZC	PH	PQ
R1(PID, L, H, W, OID,ODate, CID,TP)															
R2(CID, A, C, S,ZC , PH)															
R3(PID, OID, PQ)															

Now, let's label as **Known** and **Unknown**

	PID	L	W	H	WGT	OID	ODate	CID	TP	A	C	S	ZC	PH	PQ
R1(PID, L, H, W, WGT, OID, ODate, CID,TP)	K	K	K	K	K	K	K	K	K	U	U	U	U	U	U
R2(CID, A, C, S,ZC , PH)	U	U	U	U	U	U	U	K	U	K	K	K	K	K	U
R3(PID, OID, PQ)	K	U	U	U	U	K	U	U	U	U	U	U	U	U	K

Now, let's loop over the functional dependencies:

1) Starting with FD1: ProductID → Length, Width, Height, Weight, we need to update any unknowns to known, if the ProductID is known. This functional dependency tells us if we have ProductID, then we know Length, Width, Height, and Weight, so let's go ahead and update the values in the table:

	PID	L	W	H	WGT	OID	ODate	CID	TP	A	C	S	ZC	PH	PQ
R1(PID, L, H, W, WGT, OID, ODate, CID,TP)	K	K	K	K	K	K	K	K	K	U	U	U	U	U	U
R2(CID, A, C, S,ZC , PH)	U	U	U	U	U	U	U	K	U	K	K	K	K	K	U
R3(PID, OID, PQ)	K	K	K	K	K	K	U	U	U	U	U	U	U	U	K

2) Now, looking at FD2: OrderID → OrderDate, CustomerID, TotalPrice, we need to update any unknowns to known, if the OrderID is known. This functional dependency tells us if we have OrderID, then we know OrderDate, CustomerID, and TotalPrice, so let's go ahead and update the values in the table:

	PID	L	W	H	WGT	OID	ODate	CID	TP	A	C	S	ZC	PH	PQ
R1(PID, L, H, W, WGT, OID, ODate, CID,TP)	K	K	K	K	K	K	K	K	K	U	U	U	U	U	U
R2(CID, A, C, S,ZC , PH)	U	U	U	U	U	U	U	K	U	K	K	K	K	K	U
R3(PID, OID, PQ)	K	K	K	K	K	K	K	K	K	U	U	U	U	U	K

3) Now, looking at FD3: CustomerID → Address, City, State, ZipCode, PhoneNumber, we need to update any unknowns to known, if the CustomerID is known. This functional dependency tells us if we have CustomerID, then we know Address, City, State, ZipCode, and PhoneNumber, so let's go ahead and update the values in the table:

	PID	L	W	H	WGT	OID	ODate	CID	TP	A	C	S	ZC	PH	PQ
R1(PID, L, H, W, WGT, OID, ODate, CID,TP)	K	K	K	K	K	K	K	K	K	K	K	K	K	K	U
R2(CID, A, C, S,ZC , PH)	U	U	U	U	U	U	U	K	U	K	K	K	K	K	U
R3(PID, OID, PQ)	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K

4) Now, looking at FD4: ProductID, OrderID → ProductQuantity, we need to update any unknowns to known, if ProductID and OrderID are known. This functional dependency tells us if we have ProductID and OrderID, then we know ProductQuantity, so let's go ahead and update the values in the table:

	PID	L	W	H	WGT	OID	ODate	CID	TP	A	C	S	ZC	PH	PQ
R1(PID, L, H, W, WGT, OID, ODate, CID,TP)	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K
R2(CID, A, C, S,ZC , PH)	U	U	U	U	U	U	U	K	U	K	K	K	K	K	U
R3(PID, OID, PQ)	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K

The decomposition is lossless. Looking at the above table, we can see that there are two rows that are comprised entirely out of "K" symbols, meaning it is lossless. Now, we must check if the decomposition preserves all of its functional dependencies.

First, let's look at FD1, R1,R2, and R3:

1) Since all of the attributes in FD1 are known in R1, it is not necessary to perform a join to get FD1, therefore, FD1 is preserved

Now, let's look at FD2, R1, R2, and R3:

1) Since all of the attributes in FD2 are known in R1, it is not necessary to perform a join to get FD2, therefore, FD2 is preserved

Now, let's look at FD3, R1, R2, and R3:

1) Since all of the attributes in FD3 are known in R2, it is not necessary to perform a join to get FD3, therefore, FD3 is preserved

Finally, let's look at FD4, R1, R2, and R3:

1) Since all of the attributes in FD4 are known in R3, it is not necessary to perform a join to get FD4, therefore, FD4 is preserved

So, all of the functional dependencies are preserved through this decomposition.

Putting it all together, since the decomposition is lossless and all of the functional dependencies are preserved, we have a **good decomposition**.