A document with a signature

Description automatically generated

Question 1  
A) F = {A->BC, B->D, D->E, C->A}  
 1) Augmentation: {B->D} -> {BC->CD}  
 Augmentation: {D->E} -> {CD->CE}  
 Augmentation: {C->A} -> {CE->AE}  
 Transitivity: BC->CD->CE->AE  
 BC->AE  
 2) Decomposition: {A->BC} -> {A->B}  
 Transitivity: {B->D} -> {A->D}  
 Transitivity: {D->E} -> {A->E}  
 Union: {A->D} + {A->E} -> {A->ED}  
 A->ED  
 3) Unable to be logically implied. The only FD with C on the right-hand side requires A, and the only FD with A on the right-hand side requires C. We require either A or C in the left-hand side of the provided FD.

B) F = {A->BC, CD->E, E->F, B->D}  
 1) Candidate Key  
 Augmentation: {A->BC} -> {AC->BC}

Trivial: {A->A} -> {AC->ABC}

Augmentation: {B->D} -> {AC->ABCD}  
 Augmentation: {D->E} -> {AC->ABCDE}  
 Augmentation: {E->F} -> {AC->ABCDEF}  
 2) Candidate Key  
 Augmentation: {A->BC} -> {AD->BCD}

Trivial: {A->A} -> {AC->ABCD}  
 Augmentation: {D->E} -> {AC->ABCDE}  
 Augmentation: {E->F} -> {AC->ABCDEF}

3) Candidate Key  
 Augmentation: {A->BC} -> {A->BC}

Trivial: {A->A} -> {A->ABC}  
 Augmentation: {B->D} -> {A->ABCD}  
 Augmentation: {D->E} -> {A->ABCDE}  
 Augmentation: {E->F} -> {A->ABCDEF}

C) F = {A->B, AB->C, CD->E, E->FG}  
 1) A->C  
 Start: A->B  
 Augmentation: {A->A} -> {A->AB}  
 Transitivity: {A->AB} + {AB->C} -> {A->C}

2) CD->FG  
 Transitivity: {CD->E} + {E->FG} -> {CD->FG}

3) AD->E  
 Start: A->B  
 Augmentation: {A->A} -> {A->AB}  
 Transitivity: {A->AB} + {AB->C} -> {A->C}  
 Augmentation: {A->C} -> {AD->CD}  
 Transitivity: {AD->CD} + {CD->E} -> {AD->E}  
 4) AD->FG  
 Transitivity: {AD->E} + {E->FG} -> {AD->FG}

5) AD->ABCDEFG  
 Augmentation: {A->B} -> {AD->ABD}  
 Augmentation: {AB->C} -> {AD->ABCD}  
 Augmentation: {CD->E} -> {AD->ABCDE}  
 Augmentation: {E->FG} -> {AD->ABCDEFG}

Question 2  
A) R(W, X, Y, Z); F = {X->Y, YZ->W, W->Z}; G = {X->YZ, Y->W, W->Z, Z->Y}  
 Attribute closure for F.  
 {}+={}

{W}+={*W*,*Z*}

{X}+={*X*,*Y*}

{Y}+={*Y*}

{Z}+={*Z*}

{W,X}+={*W*,*X*,*Y*,*Z*}

{W,Y}+={*W*,*Y*,*Z*}

{W,Z}+={*W*,*Z*}

{X,Y}+={*X*,*Y*}

{X,Z}+={*X*,*Z*,*Y*,*W*}

{Y,Z}+={*Y*,*Z*,*W*}

{W,X,Y}+={*W*,*X*,*Y*,*Z*}

{W,X,Z}+={*W*,*X*,*Z*,*Y*}

{W,Y,Z}+={*W*,*Y*,*Z*}

{X,Y,Z}+={*X*,*Y*,*Z*,*W*}

{W,X,Y,Z}+={*W*,*X*,*Y*,*Z*}

Attribute closure for G.

{}+={}

{W}+={*W*,*Z*,*Y*}

{X}+={*X*,*Y*,*Z*,*W*}

{Y}+={*Y*,*W*,*Z*}

{Z}+={*Z*,*Y*,*W*}

{W,X}+={*W*,*X*,*Y*,*Z*}

{W,Y}+={*W*,*Y*,*Z*}

{W,Z}+={*W*,*Z*,*Y*}

{X,Y}+={*X*,*Y*,*Z*,*W*}

{X,Z}+={*X*,*Z*,*Y*,*W*}

{Y,Z}+={*Y*,*Z*,*W*}

{W,X,Y}+={*W*,*X*,*Y*,*Z*}

{W,X,Z}+={*W*,*X*,*Z*,*Y*}

{W,Y,Z}+={*W*,*Y*,*Z*}

{X,Y,Z}+={*X*,*Y*,*Z*,*W*}

{W,X,Y,Z}+={*W*,*X*,*Y*,*Z*}

1. The attribute closures for functional dependencies F and G are shown above, and the set of functional dependencies for G is larger than that for F. As an example, X is a candidate key for G, but the first candidate keys for F require two attributes (either {W, X} or {X, Z}).
2. To adjust F+ to be equivalent to G+ we can modify the FD YZ->W to just be Y->W, and add an additional FD Z->Y. In this configuration, the attributes W, Y, and Z form circular dependencies, where as long as we have one of the attributes, we can always reach the others. In order to match G+, we also need to have X connect into this dependency cycle. We could use any FD in the form X->attribute, but F already has X->Y, so there was no need to change it.

B) R(V, W, X, Y, Z); F = {W->V, X->Y, XW->Z, YZ->XW}  
 1)

YZ:

Start with YZ->XW

Augment with W->V  
 YZ->VWX

XY:

Nowhere to go from here

W:

Start with W->V

Nowhere to go from here

2) YZ is a candidate key. Adding the reflexivity rule to the work above, we get YZ->VWXYZ.

C) R(A, B, C, D, E, F); F = {AB->C, CD->E, E->F, B->D}

{}+={}

{A}+={*A*}

A->A

{B}+={*B*,*D*}

B->B, B->D

{C}+={*C*}

C->C

{D}+={*D*}

D->D

{E}+={*E*,*F*}

E->E, E->F

{F}+={*F*}

F->F

{A,B}+={*A*,*B*,*C*,*D*,*E*,*F*}

AB->AB, B->D, AB->C, CD->E, E->F

{A,C}+={*A*,*C*}

AC->AC

{A,D}+={*A*,*D*}

AD->AD

{A,E}+={*A*,*E*,*F*}

AE->AE, E->F

{A,F}+={*A*,*F*}

AF->AF

{B,C}+={*B*,*C*,*D*,*E*,*F*}

BC->BC, B->D, CD->E, E->F

{B,D}+={*B*,*D*}

BD->BD

{B,E}+={*B*,*E*,*F*,*D*}

BE-BE, B->D, E->F

{B,F}+={*B*,*F*,*D*}

BF->BF, B->D

{C,D}+={*C*,*D*,*E*,*F*}

CD->CD, CD->E, E->F

{C,E}+={*C*,*E*,*F*}

CE->CE, E->F

{C,F}+={*C*,*F*}

CF->CF

{D,E}+={*D*,*E*,*F*}

DE->DE, E->F

{D,F}+={*D*,*F*}

DF->DF

{E,F}+={*E*,*F*}

EF->EF

{A,B,C}+={*A*,*B*,*C*,*D*,*E*,*F*}

See {AB}+

{A,B,D}+={*A*,*B*,*D*,*C*,*E*,*F*}

See {AB}+

{A,B,E}+={*A*,*B*,*E*,*C*,*F*,*D*}

See {AB}+

{A,B,F}+={*A*,*B*,*F*,*C*,*D*,*E*}

See {AB}+

{A,C,D}+={*A*,*C*,*D*,*E*,*F*}

ACD->ACD, CD->E, E->F

{A,C,E}+={*A*,*C*,*E*,*F*}

ACE->ACE, E->F

{A,C,F}+={*A*,*C*,*F*}

ACF->ACF

{A,D,E}+={*A*,*D*,*E*,*F*}

ADE->ADE, E->F

{A,D,F}+={*A*,*D*,*F*}

ADF->ADF

{A,E,F}+={*A*,*E*,*F*}

AEF->AEF

{B,C,D}+={*B*,*C*,*D*,*E*,*F*}

See {BC}+

{B,C,E}+={*B*,*C*,*E*,*F*,*D*}

See {BC}+

{B,C,F}+={*B*,*C*,*F*,*D*,*E*}

See {BC}+

{B,D,E}+={*B*,*D*,*E*,*F*}

BDE->BDE, E->F

{B,D,F}+={*B*,*D*,*F*}

BDF->BDF

{B,E,F}+={*B*,*E*,*F*,*D*}

BEF->BEF, B->D

{C,D,E}+={*C*,*D*,*E*,*F*}

CDE->CDE, E->F

{C,D,F}+={*C*,*D*,*F*,*E*}

CDF->CDF, CD->E

{C,E,F}+={*C*,*E*,*F*}

CEF->CEF

{D,E,F}+={*D*,*E*,*F*}

DEF->DEF

{A,B,C,D}+={*A*,*B*,*C*,*D*,*E*,*F*}

See {AB}+

{A,B,C,E}+={*A*,*B*,*C*,*E*,*F*,*D*}

See {AB}+

{A,B,C,F}+={*A*,*B*,*C*,*F*,*D*,*E*}

See {AB}+

{A,B,D,E}+={*A*,*B*,*D*,*E*,*C*,*F*}

See {AB}+

{A,B,D,F}+={*A*,*B*,*D*,*F*,*C*,*E*}

See {AB}+

{A,B,E,F}+={*A*,*B*,*E*,*F*,*C*,*D*}

See {AB}+

{A,C,D,E}+={*A*,*C*,*D*,*E*,*F*}

ACDE->ACDE, E->F

{A,C,D,F}+={*A*,*C*,*D*,*F*,*E*}

ACDF->ACDF, CD->E

{A,C,E,F}+={*A*,*C*,*E*,*F*}

ACEF->ACEF

{A,D,E,F}+={*A*,*D*,*E*,*F*}

ADEF->ADEF

{B,C,D,E}+={*B*,*C*,*D*,*E*,*F*}

BCDE->BCDE, E->F

{B,C,D,F}+={*B*,*C*,*D*,*F*,*E*}

BCDF->BCDF, CD->E

{B,C,E,F}+={*B*,*C*,*E*,*F*,*D*}

BCEF->BCEF, B->D

{B,D,E,F}+={*B*,*D*,*E*,*F*}

BDEF->BDEF

{C,D,E,F}+={*C*,*D*,*E*,*F*}

CDEF->CDEF

{A,B,C,D,E}+={*A*,*B*,*C*,*D*,*E*,*F*}

See {AB}+

{A,B,C,D,F}+={*A*,*B*,*C*,*D*,*F*,*E*}

See {AB}+

{A,B,C,E,F}+={*A*,*B*,*C*,*E*,*F*,*D*}

See {AB}+

{A,B,D,E,F}+={*A*,*B*,*D*,*E*,*F*,*C*}

See {AB}+

{A,C,D,E,F}+={*A*,*C*,*D*,*E*,*F*}

ACDEF->ACDEF

{B,C,D,E,F}+={*B*,*C*,*D*,*E*,*F*}

BCDEF->BCDEF

{A,B,C,D,E,F}+={*A*,*B*,*C*,*D*,*E*,*F*}

See {AB}+

Question 3

Step 1

Fc = {AB->C, C->A, BC->D, ACD->B, D->EG, BE->C, CG->BD, CE->G}

Step 2

Check AB->C: neither A nor B can be removed

Check BC->D: neither B nor C can be removed

Check ACD->B: None can be removed

Check BE->C: neither B nor E can be removed

Check CG->BD: neither C nor G can be removed

Check CE->G: neither C nor E can be removed

Fc = {AB->C, C->A, BC->D, ACD->B, D->EG, BE->C, CG->BD, CE->G}

Step 3  
 Check AB->C: C cannot be removed

Check C->A: A cannot be removed

Check BC->D: D cannot be removed

**Check ACD->B: B can be removed**

Check D->EG: neither E nor G can be removed

Check BE->C: C cannot be removed

**Check CG->BD: D can be removed**

Check CE->G: cannot be removed

Fc = {AB->C, C->A, BC->D, ACD->∅, D->EG, BE->C, CG->B, CE->G}

Step 4: Remove ACD->∅

Fc = {AB->C, C->A, BC->D, D->EG, BE->C, CG->B, CE->G}

Step 5A:

Minimal Cover = {AB->C, C->A, BC->D, D->E, D->G, BE->C, CG->B, CE->G}

Question 4

A)  
(1) ? = XA < YA. That is, the lexicographical ordering is based on the alphabetical ordering of the attribute name  
(2) ? = XA == YA AND DA == DA. That is, two attributes are equal if and only if both the attribute names are equal, and the domains are the same.

B) ∀X = {B1, B2, …, Bm} ⊆ R : {B1 <B2 < … < Bm}  
Basically this requires that all attributes being considered must be listed in lexicographical order.

C) X <AS Y ↔ one of the following is true:  
 a) k < l (if X has fewer attributes than Y, it is considered less than Y)  
 b) k = l and Bi == Ci for all i < j and Bj < Cj for some j between 1-k (order by the first attribute that is different between the sets)

D)   
candidate\_keys = []

for subset1 in all possible attribute subsets:

Check if closure of the subset equals the relation R:

Is\_minimal = True

If yes:

For subset2 in all possible attribute subsets of subset1:

If subset2 != subset1 and closure(subset2) == R:

Is\_minimal = False

Break

If is\_minimal: candidate\_keys.append(subset1)

Return candidate\_keys.sort(key= <AS) # Sort candidate keys using <AS

E)

left\_side\_attributes = set()

right\_side\_attributes = set()

for (X, Y) in F:

left\_side\_attributes.update(X)

right\_side\_attributes.update(Y)

# Step 1

attributes\_neither\_side = set(R) - (left\_side\_attributes ∪ right\_side\_attributes)

# Step 2

attributes\_left\_only = left\_side\_attributes - right\_side\_attributes

# Step 3

attributes\_right\_only = right\_side\_attributes - left\_side\_attributes

# Step 4

step4\_attributes = attributes\_neither\_side ∪ attributes\_left\_only

# Step 5

if closure(step4\_attributes) == R:

return [step4\_attributes]

# Step 6

step6\_attributes = set(R) - (attributes\_right\_only ∪ step4\_attributes)

# Step 7

candidate\_keys = []

for comb in combinations of step6\_attributes:

if closure(step4\_attributes ∪ comb) == R:

candidate\_keys.append(combined\_attributes)

return candidate\_keys