----COMP9311 Assignment 3 z3492782 Li Yu ----

1. Consider a relation R(A,B,C,D,E,F). For each of the following sets of functional dependencies (i.e, i. to iv.), assuming that those are the only dependencies that hold for R, do the following:

a. List all of the candidate keys for R.
b. What are the BCNF violations, if any?
c. Decompose the relation, as necessary, into collections of BCNF relations?

i. AD → B, C → D, BC → A, B → D

Answers for i:

a. Candidate keys: ACEF, BCEF

A candidate key is any set X, such that $X^+ = R$ and there is no Y subset of X such that $Y^+ = R$.
Appear only in Left: C
Appear only in Right: None
Appear in Left & Right: A,B,D
No Appear: E,F
So, the Candidate key must have C, might have EF and might combine with A ,B, D.

1) C->{C,D}
2) CEF->{C,D,E,F}
3) ACEF->{ACDEF}->{ABCDEF}, So ACEF is one candidate key
4) BCEF->{ABCEF}->{ABCDEF}, So BCEF is one candidate key
5) CDEF->{CDEF}

b. BCNF violations: Not BCNF.

Not BCNF. Because none of the left hand sides (AD, C, BC, B) contains a key.
E.g.
in AD→B, AD does not contain a key on the left hand side.
in C → D, C does not contain a key on the left hand side.
in BC → A, BC does not contain a key on the left hand side.
in B → D, B does not contain a key on the left hand side.

c. Decompose the relation:
Final schema (with keys boldened and underlined): **B**D, **AB**, **C**D, **ACEF**

-Start from a schema ABCDEF, with key ACEF.
-The FD AD → B violates BCNF (FD with non key on LHS).
-To fix, we need to decompose into table: ADB and ACDEF.
-FDs for ADB are { AD→B， B→D }, with key AD and AB.

-The FD B→D violates BCNF (FD with non key on LHS).
- To fix, we need to decompose ADB into table: BD and AB.
-FDs for BD is { B→D }, therefore key is B, therefore BCNF.
-FDs for AB is { }, therefore key is AB, therefore BCNF.
-Continue with ACDEF.
-FDs for ACDEF is { C→D}.
-Key for ACDEF is ACEF, and FD C → D violates BCNF (FD with non key on LHS).
-To fix, we need to decompose into tables: CD and ACEF.
-FDs for CD is { C → D }, therefore key is C, therefore BCNF.
-FDs for ACEF is {}, so key is ACEF and table is BCNF.
-Final schema(with keys boldened and underlined): **<u>B</u>**D, **<u>AB</u>**, **<u>C</u>**D, **<u>ACEF</u>**

ii. BC → E, C → AB, AF → CD

Answers for ii:

a. Candidate keys: AF, CF

A candidate key is any set X, such that $X^+ = R$ and there is no Y subset of X such that $Y^+ = R$.
Appear only in Left: F
Appear only in Right: D,E
Appear in Left & Right: A,B,C
No Appear: None
So, the Candidate key must have F, might combine with A, B, C.

1) F->{F}
2) AF->{A,C,D,F}->{A,B,C,D,F}->{A,B,C,D,E,F} So AF is one candidate key
3) BF->{B,F}
4) CF->{ABCF}->{ABCEF}->{ABCDEF } So CF is one candidate key

b. BCNF violations: Not BCNF.

Not BCNF. Because the left hand sides (BC, C) do not contains a key.
E.g.
in BC→E, BC does not contain a key on the left hand side.
in C → AB, C does not contain a key on the left hand side.

c. Decompose the relation:
Final schema (with keys boldened and underline): **<u>BC</u>**E, **<u>C</u>**AB, **<u>CDF</u>**

-Start from a schema ABCDEF, with key AF.
-The FD BC → E violates BCNF (FD with non key on LHS).
-To fix, we need to decompose into table: BCE and ABCDF.
-FDs for BCE is { BC → E }, therefore key is BC, therefore BCNF.
-FDs for ABCDF are { C → AB, AF → CD }.
-Key for ABCDF is AF, and FD C → AB violates BCNF (FD with non key on LHS).

-To fix, we need to decompose ABCDF into tables: CAB and CDF.
-FDs for CAB is { C → AB }, therefore key is C, therefore BCNF.
-FDs for CDF is {}, so key is CDF and table is BCNF.
-Final schema (with keys boldened and underline): **<u>BC</u>**E, **<u>C</u>**AB, **<u>CDF</u>**

iii. ABF → D, CD → E, BD → A

Answers for iii:

a. Candidate keys: ABCF, BCDF

A candidate key is any set X, such that $X^+ = R$ and there is no Y subset of X such that $Y^+ = R$.
Appear only in Left: B,C,F
Appear only in Right: E
Appear in Left & Right: A,D
No Appear: None
So, the Candidate key must have BCF, might combine with A ,D.

1) BCF->{BCF}
2) ABCF->{A,B,C,D,F}->{A,B,C,D,E,F} So ABCF is one candidate key
3) BCDF->{ABCDF}->{A,B,C,D,E,F} So BCDF is one candidate key

b. BCNF violations: Not BCNF.

Not BCNF. Because none of the left hand sides(ABF, CD, BD) contains a key.
E.g.
in ABF→D, ABF does not contain a key on the left hand side.
in CD→E, CD does not contain a key on the left hand side
in BD→A, BD does not contain a key on the left hand side.

c. Decompose the relation:
Final schema (with keys boldened and underlined): **<u>BD</u>**A, **<u>BDF</u>**, **<u>ABCEF</u>**

-Start from a schema ABCDEF, with key ABCF.
-The FD ABF → D violates BCNF (FD with non key on LHS).
-To fix, we need to decompose into table: ABFD and ABCEF.
-FDs for ABFD are { ABF → D, BD → A }, with key ABF.
-Key for ABFD is ABF, and FD BD → A violates BCNF (FD with non key on LHS).
-To fix, we need to decompose ABFD into tables: BDA and BDF.
-FDs for BDA are { BD→A }, therefore key is BD, therefore BCNF.
-FDs for BDF are { }, therefore key is BDF, therefore BCNF.
-Continue with ABCEF.
-FDs for ABCEF is { }.
-Key for ABCEF is ABCEF, therefore BCNF.
-Final schema (with keys boldened and underlined): **<u>BD</u>**A, **<u>BDF</u>**, **<u>ABCEF</u>**

iv. AB → D, BCD → EF, B → C

a. Candidate keys: AB

A candidate key is any set X, such that $X^+ = R$ and there is no Y subset of X such that $Y^+ = R$.

Appear only in Left: A,B

Appear only in Right: E,F

Appear in Left & Right: C,D

No Appear: None

So, the Candidate key must have AB, and might combine with C, D.

1) AB->{A,B,D}->{A,B,C,D}->{A,B,C,D,E,F} As, AB only appear in the left. So, AB is the only candidate key.

b. BCNF violations: Not BCNF

Not BCNF. Because none of the left hand sides(BCD, B) contains a key.

E.g.

in BCD → EF, BCD does not contain a key

in B → C, B does not contain a key

c. Decompose the relation:

Final schema(with keys boldened and underlined): **<u>AB</u>**D, **<u>B</u>**C, **<u>BDEF</u>**

-Start from a schema ABCDEF, with key AB.

-The FD BCD→EF violates BCNF (FD with non key on LHS).

-To fix, we need to decompose into table: BCDEF and ABCD .

-FDs for BCDEF is { BCD→EF, B → C }.

-Key for BCDEF is BCD, and B → C violates BCNF (FD with non key on LHS).

-To fix, we need to decompose BCDEF into tables: BC and BDEF.

- FDs for BC is { B → C }, therefore key is B, therefore BCNF.

- FDs for BDEF is { } with key BDEF, therefore BCNF.

-Continue with ABCD.

-FDs for ABCD are { AB→D， B→C }.

-Key for ABCD is AB, and B→C violates BCNF (FD with non key on LHS).

-To fix, we need to decompose ABCD into tables: BC and ABD.

- FDs for BC is { B → C }, therefore key is B, therefore BCNF.

- FDs for ABD is { AB → D }, therefore key is AB, therefore BCNF.

-Final schema(with keys boldened and underlined): **<u>AB</u>**D, **<u>B</u>**C, **<u>BDEF</u>**

2. Assuming the schema from assignment 2 (i.e., the ASX database), give the following queries in relational algebra.

i. List all the company names that are in the sector of "Technology".

Answer:

SectorCode=Proj[code](Sel[sector='Technology'] (Category))

Answer=Proj[name](SectorCode join Company)

ii. List all the company codes that have more than five executive members on record (i.e., at least six).

Answer:
R1=GroupBy[code, count(person)] (Executive)
R2=Rename[1->code, 2->n](R1)
Answer=Proj[code](Sel[n>5](R2))

iii. Output the person names of the executives that are affiliated with more than one company.

Answer:
R1= GroupBy [person,count(code)](Executive)
R2=Rename[1->person, 2->n](R1)
Answer=Proj[person](Sel[n>1](R2))

iv. List all the companies (by their Code) that are the only one in their Industry (i.e., no competitors). Same as Assignment 2, please include both Code and Industry in the output.

Answer:
R1=GroupBy [industry, count[code]] (Category)
R2=Rename[1->industry, 2->n](R1)
R3=Proj[industry](Sel[n=1](R2))
Answer=Proj[code, R3.industry](Category Join[industry] R3)

3.Suppose relations R, S and T have r tuples, s tuples and t tuples, respectively. Derive the minimum and maximum numbers of tuples that the results of the following expressions can have.

i.R UNION (S INTERSECT T).

Step 1:
Tmp1 = S INTERSECT T
Assumptions: S and T are union-compatible.
|s∩t|<= min(|s|+|t|)
Min_Tmp1=0 (when S∩T=∅)
Max_Tmp1=t (when S∩T!=∅ and T⊂S)
or
Max_Tmp1=s (when S∩T!=∅ and S⊂T)

Step 2:
Tmp2= R UNION Tmp1
Assumptions: R and Tmp1 are union-compatible.
|r∪Tmp1|<=|r|+|Tmp1|
Min_Tmp2=|r∪Tmp1|=|r∪Min_Tmp1|=r (when S∩T=∅)
Max_Tmp2=|r∪Tmp1|=|r|+|Max_Tmp1|=r+t (when S∩T!=∅ and T⊂S)
or
Max_Tmp2=|r∪Tmp1|=|r|+| Max_Tmp1|=r+s (when S∩T!=∅ and S⊂T)

Answer:
R UNION (S INTERSECT T)

Assumptions: S and T are union-compatible, R and (S INTERSECT T) are union-compatible.
Min=|r∪Tmp1|=|r∪Min_Tmp1|=r (when S∩T=∅)
Max=|r∪Tmp1|=|r|+| Max_Tmp1|=r+t (when S∩T!=∅ and T⊂S)
or
Max =|r∪Tmp1|=|r|+| Max_Tmp1|=r+s (when S∩T!=∅ and S⊂T)

ii .SEL[c](R × S), for some condition c.

Step1:
Tmp1 =R × S
|r×s|=|r|*|s|
Min_Tmp1=r*s
Max_Tmp1= r*s
Step2:
Tmp2=SEL[c](Tmp1), for some condition c
Min_Tmp2=0 (when no match on the condition c.)
Max_Tmp2= r*s (when all match on the condition c)

Answer:
SEL[c](R × S), for some condition c.
Min=0 (when no match on the condition c)
Max=r*s (when all match on the condition c)

iii. R - PROJ[a](R JOIN S), for some list of attributes a.

Step1:
Tmp1 = R JOIN S
R JOIN S=Proj[R ∪ S](Sel[match](R × S))
|r⋈s|≪|r × s|,
Min_Tmp1=r*s
Max_Tmp2=r*s

Step2:
Tmp2=PROJ[a](Tmp1)= PROJ[a](R JOIN S)
Min_Tmp2= 0 (when (R join S) have no attribute a)
Max_Tmp2= r*s (when r*s>0)

Step3:
Tmp3=R – Tmp2=R - PROJ[a](R JOIN S)
Assumptions: R and PROJ[a](R JOIN S) are union-compatible.
Min_Tmp3 =r - Max_Tmp2 =r-r*s=0 (When (R join S) all match attribute a and PROJ[a](R JOIN S)∩R=R)
Max_Tmp3=r - Min_Tmp2=r-0=r (when (R join S) all match attribute a and (R JOIN S)= ∅)

Answer:

R - PROJ[a](R JOIN S), for some list of attributes a.

Assumptions: R and PROJ[a](R JOIN S) are union-compatible.

Min= r - Max_Tmp2 =r-r*s=0 (When (R join S) all match attribute a and PROJ[a](R JOIN S) ∩ R=R)

Max= r- Min_Tmp2=r-0= r (when (R join S) all match attribute a and (R JOIN S)= ∅)

4.

i. For the following execution schedule, construct its precedence graph. Is this schedule serialisable? Explain your answer.

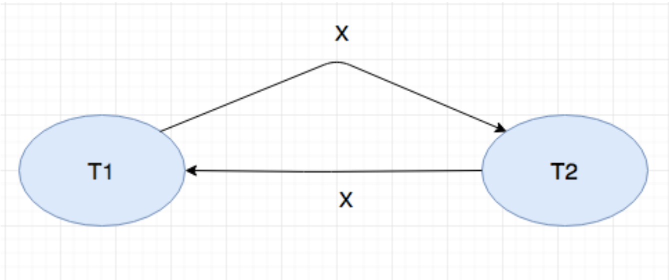
```
T1:R(X) T2:R(X) T1:W(X) T2:W(X) T2:R(Y) T1:R(Y) T1:W(Y) T2:W(X)
```

T1	R(X)		W(X)			R(Y)	W(Y)	
T2		R(X)		W(X)	R(Y)			W(X)

T2: R(X), T1: W(X) conflict gives T2-->T1

T1: W(X), T2: W(X) conflict gives T1-->T2

Graph has a cycle ==> not serialisable



ii. For the following execution schedule, construct its precedence graph. Is this schedule serialisable? Explain your answer.

```
T3:R(X) T4:W(Y) T4:W(Z) T1:W(Y) T2:R(Y) T3:R(D) T2:W(X) T1:R(X)
```

T1				W(Y)				R(X)
T2					R(Y)		W(X)	
T3	R(X)					R(D)		
T4		W(Y)	W(Z)					

T3: R(X), T2: W(X) conflict gives T3-->T2

T4: W(Y), T1: W(Y) conflict gives T4-->T1

T1: W(Y), T2:R(Y) conflict gives T1-->T2

T2: W(X), T1: R(X) conflict gives T2-->T1

Graph has a cycle ==> not serialisable