20T3 COMP9311 Sample Solution

Q1:

1. **False**, the CREATE TYPE statement allows to create a new type.

2. **False**, because in SQL AS is only to rename the attribute.

3. **True**, 2NF requires that every *nonprime* attribute is fully dependent on every candidate key. BCNF requires that every attribute must be fully dependent on every key.

4. **False**, the *primary key* is an attribute or a set of attributes that uniquely identify a specific instance of an entity. Every entity in the data model must have a primary key whose values uniquely identify instances of the entity.

5. **False**, A lossless and dependency-preserving decomposition into 3NF is always possible.

6. **True**

7. **True**, SQL cannot control sequences of database operations

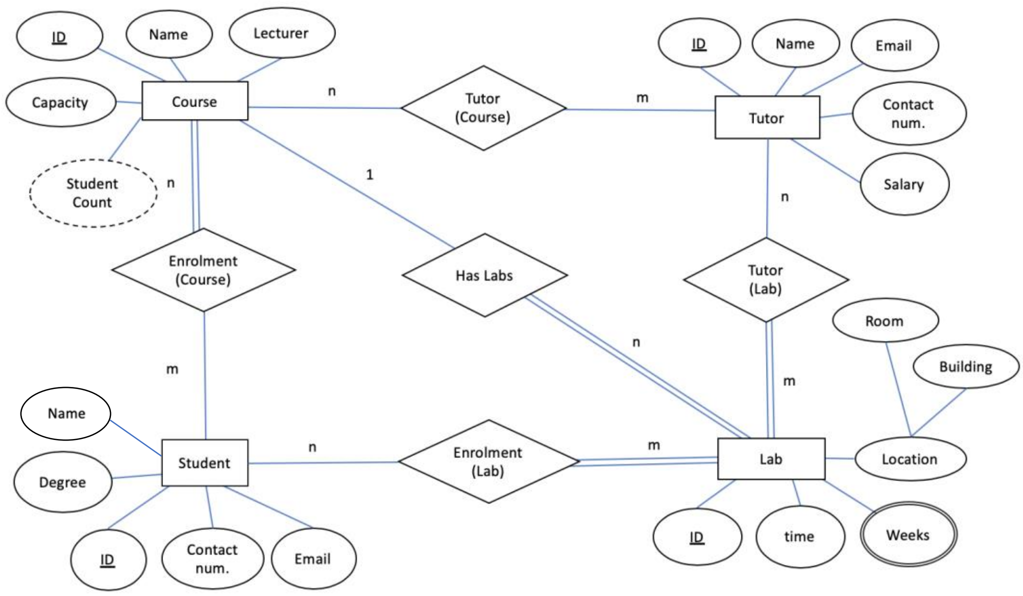
8. **False**, Hash index is not suitable for range check, it is suitable for specific value query

9. **False**, ISAM does not store data

10. **False**, optimistic control is a good option if there is not much interaction between transactions.

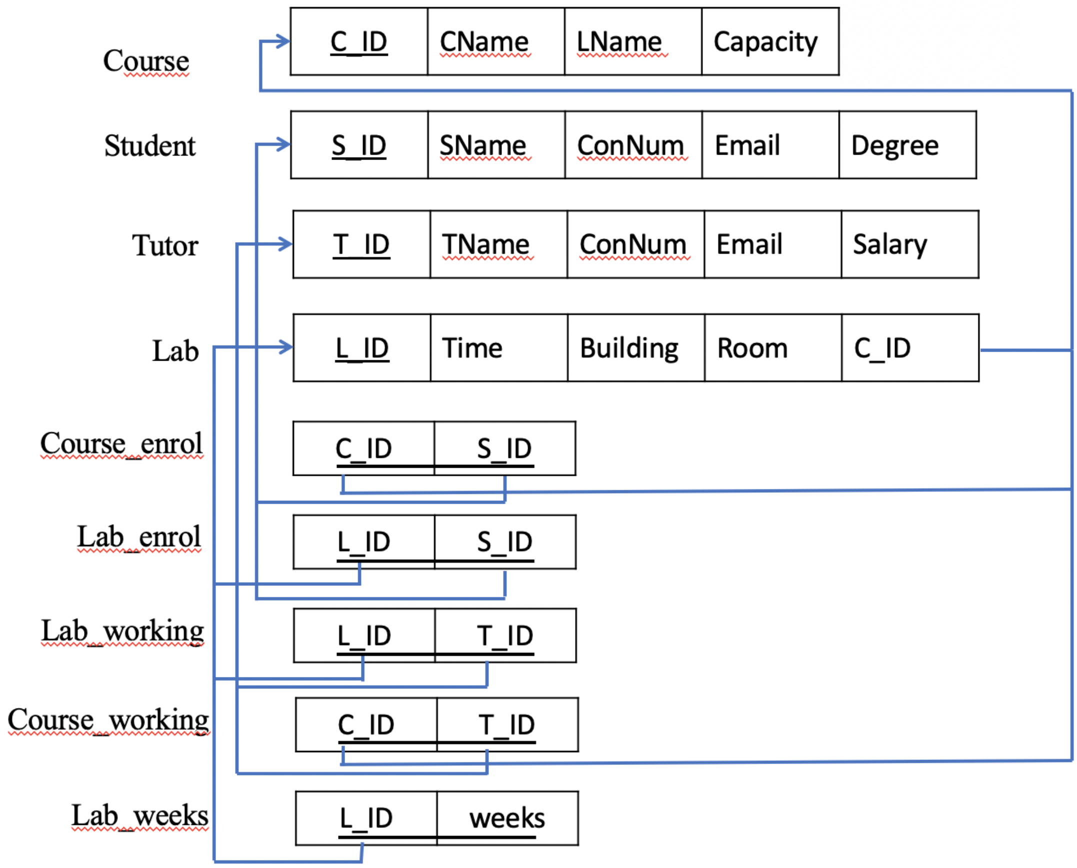
Q2:

(a):



ID
Name
Lecturer
Capacity
Course
Student Count
n
Tutor (Course)
m
ID
Name
Email
Tutor
Contact num.
Salary
1
n
Enrolment (Course)
Has Labs
Tutor (Lab)
Room
Building
m
n
m
Name
Student
n
Enrolment (Lab)
m
Lab
Location
Degree
ID
Contact num.
Email
ID
time
Weeks

(b):



Course
C_ID
CName
LName
Capacity
Student
S_ID
SName
ConNum
Email
Degree
Tutor
T_ID
TName
ConNum
Email
Salary
Lab
L_ID
Time
Building
Room
C_ID
Course_enrol
C_ID
S_ID
Lab_enrol
L_ID
S_ID
Lab_working
L_ID
T_ID
Course_working
C_ID
T_ID
Lab_weeks
L_ID
weeks

Q3:

(a):

(1):

Reduce right side.

F'={BD->C, BD->H, BC->H, BC->I, EI->H, H->A, H->B, I->E, EJ->I}

Reduce left side.

BD->C,

B^+={B}; thus B->C is not inferred by F'.

Hence, BD->C cannot be replaced by D->C.

D^+={D}; thus D->C is not inferred by F'.

Hence, BD->C cannot be replaced by B->C.

EI->H,

E^+={E}; thus E->H is not inferred by F'.

Hence, EI->H cannot be replaced by E->H.

I^+= {E, H, I}; thus I->H is inferred by F'.

Hence, EI->H can be replaced by I->H.

Iteratively reduce left side, then we can get:

F'' = {BD->C, BD->H, BC->H, BC->I, I->H, H->A, H->B, I->E, EJ->I}

Remove redundant FDs.

$BD^+|_{F'' - \{BD\text{->}C\}}$ = {A, B, D, H}; thus BD->C is not inferred by F''– {BD->C}. That is, BD->C is not redundant.

$BD^+|_{F'' - \{BD\text{->}H\}}$ = {A, B, C, D, E, H, I}; thus BD->H is redundant.

Thus, we can remove BD->H from F'' and get F'''.

F'''= {BD->C, BC->H, BC->I, I->H, H->A, H->B, I->E, EJ->I}

$BC^+|_{F''' - \{BC\text{->}H\}}$ = {A, B, C, E, H, I}; thus BC->H is redundant.

Thus, we can remove BC->H from F'' and get F''''.

F''''= {BD->C, BC->I, I->H, H->A, H->B, I->E, EJ->I}

Iteratively, we can get F_{min}

Thus, **F_{min} = {BD->C, BC->I, I->H, H->A, H->B, I->E, EJ->I}.**

(2):

Find a super key X.

Let X:={BCDEGJHI},

Try to remove B, $\{CDEGJHI\}^+$= {A,B,C,D,E,G,H,I,J}

Thus, X:= {CDEGJHI}

Try to remove C, $\{DEGJHI\}^+$= {A,B,C,D,E,G,H,I,J}

Thus, X:= {DEGJHI}

Try to remove D, $\{EGJHI\}^+$= {A,B,E,G,H,I,J}

Thus, D cannot be removed.

Try to remove E, $\{DGJHI\}^+$= {A,B,C,D,E,G,H,I,J}

Thus, X:= {DGJHI}

Try to remove G, $\{DJHI\}^+$= {A,B,C,D,E,H,I,J}

Thus, G cannot be removed.

Try to remove J, $\{DGHI\}^+$= {A,B,C,D,E,H,I}

Thus, J cannot be removed.
Try to remove H, $\{DGJI\}^{+}$= {A,B,C,D,E,G,H,I,J}
Thus, X:= {DGJI}
Try to remove I, $\{DGJ\}^{+}$= {D,G,J}
Thus, I cannot be removed.
So {DGJI} is a candidate key and add to T.

Find another super key X.
Let X:= {BCDEGJH},
Try to remove B, $\{CDEGJH\}^{+}$= {A,B,C,D,E,G,H,I,J}
Thus, B can be removed.
Try to remove C, $\{DEGJH\}^{+}$= {A,B,C,D,E,G,H,I,J}
Thus, C can be removed.
Try to remove D, $\{EGJH\}^{+}$= {A,B,E,G,H,I,J}
Thus, D cannot be removed.
Try to remove E, $\{DGJH\}^{+}$= {A,B,C,D,E,G,H,I,J}
Thus, E can be removed.
Also, we can find that G,J,H cannot be removed.
So {DGJH} is a candidate key and add to T.

Find another super key X.
Let X:= {BCDEGJ},
Try to remove B, $\{CDEGJ\}^{+}$= {A,B,C,D,E,G,H,I,J}
Thus, B can be removed.
Try to remove C, $\{DEGJ\}^{+}$= {A,B,C,D,E,G,H,I,J}
Thus, C can be removed.
Also, we can find that D,E,G,J cannot be removed.
So {DEGJ} is a candidate key and add to T.

Find another super key X.
Let X:= {BCDGJ},
Try to remove B, $\{CDGJ\}^{+}$= {C,D,G,J}
Thus, B cannot be removed.
Try to remove C, $\{BDGJ\}^{+}$= {A,B,C,D,E,G,H,I,J}
Thus, C can be removed.
Also, we can find that D,G,J cannot be removed.
So {BDGJ} is a candidate key and add to T.

Cannot find any other super keys.
So, candidate keys are **{BDGJ}, {DEGJ}, {DGJH}, {DGJI}.**

(3):

No.

	A	B	C	D	E	G	H	I	J
R1	a	a	a	a	b	b	**a**	b	b
R2	b	b	b	b	a	a	**a**	a	a

	A	B	C	D	E	G	H	I	J
R1	a	a	a	a	b	b	a	b	b
R2	a	a	b	b	a	a	a	a	a

No row is entirely made up by "a" value, so the decomposition is not lossless join.

(4):

1NF, since H is a non-prime attribute, while it is partially functionally dependent on EI.

BCNF:

Consider BD->CH, BD is not a superkey, split R into R1(B,D,C,H) and R2(A,B,D,E,G,I,J)

Consider BC->H, BC is not a superkey, split R1 into R11(B,C,H) and R12(B,D,C)

Consider H->B, H is not a superkey, split R11 into R111(H,C) and R112 (B,H)

Consider I->E, I is not a superkey in R2, split R2 into R21(E,I) and R22(A,B,D,G,I,J)

Consider BD->A (BD->H and H->A), BD is not a super key. Split R22 into R221 (BDA) and R222(BDGIJ).

Consider BD->I (BD->C and BC->I), BD is not a super key. Split R222 into R2221 (BDI) and R2222(BDGJ).

(b):

(1):

$R_0 = Customer \bowtie Review \bowtie Restaurant$

$R_1 = \pi_{\{rID\}}(R_0 \div (\pi_{\{gender\}} Customer)$

$Result = \pi_{\{rID\}} Review - R_1$

(2):

$R_0 = \pi_{\{cID\}}((Customer \bowtie Review \bowtie Restaurant) \div \pi_{\{rID\}}(\text{Restaurant}))$

$R_1 = \pi_{\{cID\}} Customer - \pi_{\{cID\}} Review$

$\text{Resut} = R_0 \cup R_1$

(3):

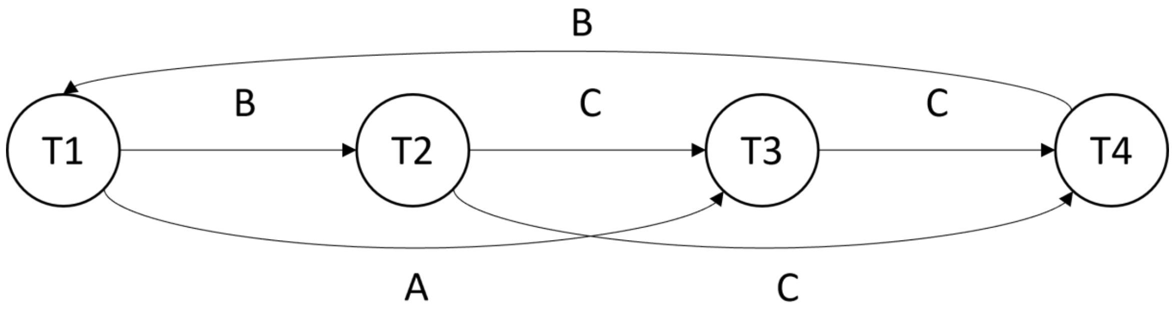
$R_0 = \pi_{\{cID,age,gender\}}(Customer \bowtie Review)$

$Result = \gamma_{\{gender, \quad AVG(age)\}}(R_0)$

Q4:

(a):

(1):

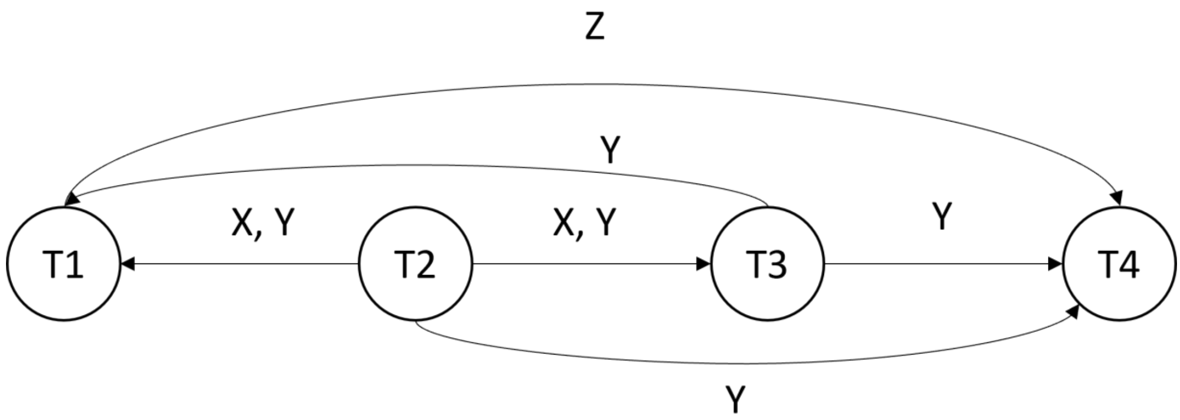


(2):

There is a dead lock.

(b):

(1):



(2):

Yes. T2-T3-T1-T4

Q5:

(a):

(1):

buffer size = 4,

Query stream: p1, p2, p3, p4, p5, p1, p2, p3, p4

Most Recently used:

	p1,	p2,	p3,	p4,	p5,	p1,	p2,	p3,	p4
1	p1	p1	p1	p1	p1	p1	p1	p1	p1
2		p2	p2	p2	p2	p2	p2	p2	p2
3			p3	p3	p3	p3	p3	p3	p4
4				p4	**p5**	p5	p5	p5	p5
	F	F	F	F	F	T	T	T	F

# of page faults = 6

Least recently used:

	p1,	p2,	p3,	p4,	p5,	p1,	p2,	p3,	p4
1	p1	p1	p1	p1	**p5**	p5	p5	p5	**p4**
2		p2	p2	p2	p2	**p1**	p1	p1	p1
3			p3	p3	p3	p3	**p2**	p2	p2
4				p4	p4	p4	p4	**p3**	p1 p2
	F	F	F	F	F	F	F	F	F

# of page faults = 9

First In First Out:

	p1,	p2,	p3,	p4,	p5,	p1,	p2,	p3,	p4
1	p1	p1	p1	p1	**p5**	p5	p5	p5	**p4**
2		p2	p2	p2	p2	**p1**	p1	p1	p1
3			p3	p3	p3	p3	**p2**	p2	p2
4				p4	p4	p4	p4	**p3**	p3
	F	F	F	F	F	F	F	F	F

# of page faults = 9

Since MRU results in the least number of page faults, it outperforms the other buffer updating policies.

(2):

buffer size = 4,

Query stream: p1, p2, p3, p4, p5, p2, p6, p3, p4

Most Recently used:

	p1,	p2,	p3,	p4,	p5,	p2,	p6,	p3,	p4
1	p1	p1	p1	p1	p1	p1	p1	p1	p1
2		p2	p2	p2	p2	p2	**p6**	p6	p6
3			p3	p3	p3	p3	p3	p3	**p4**
4				p4	**p5**	p5	p5	p5	p5
	F	F	F	F	F	T	F	T	F

# of page faults = 7

Least recently used:

	p1,	p2,	p3,	p4,	p5,	p2,	p6,	p3,	p4
1	p1	p1	p1	p1	**p5**	p5	p5	p5	**p4**
2		p2	p2	p2	p2	p2	p2	p2	p2
3			p3	p3	p3	p3	**p6**	p6	p6
4				p4	p4	p4	p4	**p3**	p3
	F	F	F	F	F	T	F	F	F

# of page faults = 8

First In First Out:

	p1,	p2,	p3,	p4,	p5,	p2,	p6,	p3,	p4
1	p1	p1	p1	p1	**p5**	p5	p5	p5	p5
2		p2	p2	p2	p2	p2	**p6**	p6	p6
3			p3	p3	p3	p3	p3	p3	p3
4				p4	p4	p4	p4	p4	p4
	F	F	F	F	F	T	F	T	T

# of page faults = 6

Since FIFO results in the least number of page faults, it outperforms the other buffer updating policies.

(b): Process omitted

(1):

2-core: {v0, v1, v2, v3, v4, v5, v6, v7, v8, v9}

(2):

3-core: {v2, v3, v4, v7, v8, v9}

(3):

(3,2)-core: {v2, v3, v4, v9}