Ashok Malhotra
11137020501

DBMS C2 Review Test

$R = \{A, B, C, D, E\}$

$F = \{A \to BC,\ C \to D,\ DE \to A\}$

Since E doesn't occur on RHS of all FD's hence dependencies, E is an essential attribute & will be present in every candidate key.

$E^+ = E$

$DE^+ = DEABC \Rightarrow DE$ is candidate key

$AE^+ = ABCED \Rightarrow AE$ is candidate key

$CE^+ = CEDAB \Rightarrow CE$ is candidate key

$BE^+ = BE$

Therefore, Candidate key (CK) = $\{DE, AE, CE\}$

Primary Attribute, PA = $\{A, C, D, E\}$,

Non-Primary attribute, NPA = $\{B\}$

$A \to BC$, $C \to D$ ✓ , $DE \to A$ ✓

2nd NF ✗

$R_1 = AB$

$(A \to B)$

$CK = \{A\}$

$R_2 = ACDE$

$(A \to C, C \to D, DE \to A)$

$CK = \{AE, DE, CE\}$

Both Table R_1 & R_2 satisfy BCNF condition, so no transition dependency should exist, so both the tables are in 3NF

Dependency Preserving or not

$FD_1 = \{A \to BY$, $FD_2 = \{A \to C, C \to D, DE \to AY\}$

Now, if $FD_1 \cup FD_2 = FD$ (original)

then dependency is preserved.

$FD_1 \cup FD_2$

$A^+ = ABCD$, $DE^+ = DEACB$

$C^+ = CD$

$FD = \{A \to BC, C \to D, DE \to AY\}$

can be obtained from $FD_1 \cup FD_2$ or not ✓ ✓ ✓

Now, $FD_1 \cup FD_2 = FD$, so its dependency preserving.

Lossless or not

For lossless, common attribute should be CK of either R_1 or R_2 or both.
Here, A is CK of R_1, so condition for lossless is fulfilled ⇒ Therefore, lossless.

2. $R = \{P, Q, R, S, T, U, V, W, X\}$

$F = \{P \to RUX,\ QS \to T,\ Q \to VW\}$

PQS are the essential attributes as they don't occur in RHS of functional dependency F.

$PQS^+ = PRUXQSTVW$

Hence the candidate key, $CK = \{PQS\}$.

Primary attributes $= \{P, Q, S\}$

Non-Primary attributes $= \{R, T, U, V, W, X\}$

Verifying for 2NF

$P \to RUX$ partial dependency as RUX non-prime & P proper subset of CK.

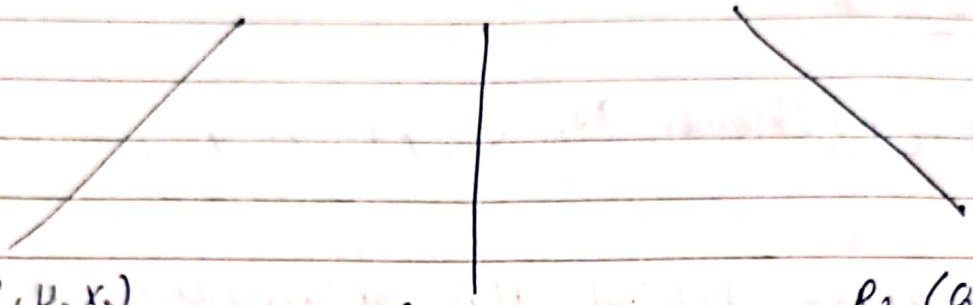
$QS \to T$ partial dependency as T non-prime & as proper subset of CK.

$Q \to VW$ partial dependency as VW non prime & Q proper subset of CK.

For 2NF

⇒ no partial dependency.

R(P, Q, R, S, T, U, V, W, X)



$R_1(P, R, U, X)$	$R_2(Q, S, T)$	$R_3(Q, V, W)$
$P \to RUX$	$QS \to T$	$Q \to VW$
CK = P	CK = QS	CK = Q
PA = {P}	PA = {Q, S}	PA = {Q}
NPA = {R, U, X}	NPA = {T}	NPA = {V, W}
No partial dependency	no partial dependency.	no partial dependency

Hence, 2NF ✓.

For 3NF

a) Should be in 2NF ✓

b) No transitive dependency.

R_1

$P \to RUX$

no transitive dependency.

R_2 ⇒ $QS \to T$

no transitive dependency

R_3

$Q \rightarrow VW$

no transitive dependency.

Hence, the decomposition is in 3NF ✓.

For BCNF

1) In 3NF ✓
2) LHS of each functional dependency should be a super key or candidate key.

R_1

$P \rightarrow RUX$

P is a super/candidate key

So in BCNF

R_2

$QS \rightarrow T$

QS is a super / candidate key

So in BCNF

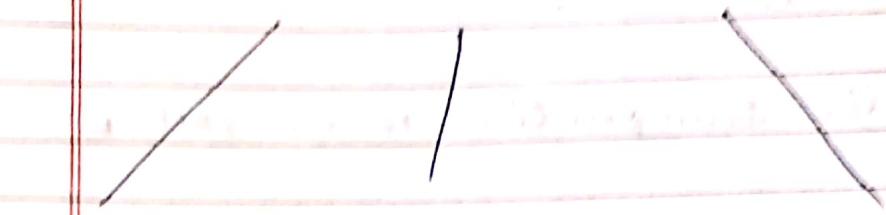
R_3 $Q \rightarrow VW$

Q is a super / candidate key

So in BCNF

Hence, the decomposed relation is in BCNF.

R(P, Q, R, S, T, U, V, W, X)



R_1(P, R, U, X) R_2(Q, S, T) R_3(Q, V, W)

Decomposition of R_2 & R_3 is lossless. as there is a common attribute Q & Q is pk of R_3

But R_1 & (R_2, R_3) is lossy as there is no common attribute hence it may lead to redundancy in table.

Decomposition is dependancy preserving as all the FD's are present in one of the relation.

$P \rightarrow RUX$ in R_1

$QS \rightarrow T$ in R_2

$Q \rightarrow VW$ in R_3

3.

```
Select cust.cust_name From Borrower as cust
Innerjoin Loan as Ln
On Ln.loan_number = cust.loan_number
where Ln.amount > 1 lakh
AND Ln.branch_name = 'Axis Bank'
```

Q:

$$\Pi_{t.A}\left(\sigma_{t.c>s.c \wedge t.D>5 \wedge s.B='IIITA'}\left(f_t(branch) \bowtie f_{s(branch)}\right)\right)$$

Optimised :

$$\Pi_{t.c}\left(f_t(branch) \bowtie_{t.c>s.c \wedge t.D>5 \wedge s.B='IIITA'} f_s(branch)\right)$$

The reason behind this optimisation is, the select condition can be applied before the join operation, to reduce the number of tuples, hence making the query run faster and more efficient.

1) Schedule is a process of lining the transactions and executing them one by one

Serial Schedules	Non-serial schedules
Schedules in which the transactions, are executed non-interleaved, i.e. one in which no transaction starts, until a running transaction has ended are called serial schedules	It is a type of scheduling where the operations of multiple transactions are interleaved. The transactions are executed in a non-serial manner, keeping the end result correct & same as the serial schedule. It can be further divided into Serializable and non-serializable schedule.

Conflict serializable schedule : A schedule is called conflict serializable if after swapping of non-conflicting operations, it can transform into serial schedule as well as if it is conflict equivalent to a serial schedule.

5 2) No, S can not execute T_1 & T_2 concurrently as the lock_S(B) statement in T_1, creates a deadlock because of the incompatibility of shared & exclusive lock as the exclusive lock on B has already been aquired by T_1 before. Also there is a shared lock on A by T_1 & T_2 can't get lock_X on A.

6. 1) I, II, III (D)

2) 3 (C)

3) (5,2), (7,2) and (9,5) (C)

4) 180000 (D)

5) P has no duplicates & Q is non-empty (B)