

**GATE PSUs**

**State Engg. Exams**

**MADE EASY**  
**WORKBOOK 2025**



**Detailed Explanations of  
Try Yourself *Questions***

---

**Computer Science & IT**  
Databases



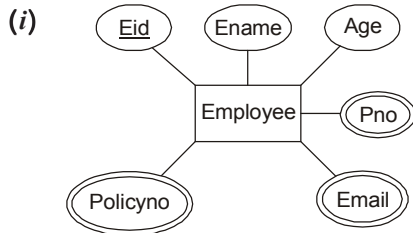
# 1

## Introduction to DBMS and Integrity Constraints and ER Model



### Detailed Explanation of Try Yourself Questions

#### T1 : Solution



Policyno., Email and Pno. are multivalued attributes i.e. each employee may have one or more set of these values.

Hence multivalued attributes are combined with key to make separate tables.

$R_1(\underline{\text{Eid}}, \text{Ename}, \text{Age}, \underline{\text{Eid}}, \text{Pno.}, \text{Policyno.}, \text{Email})$

Therefore, 1 table is required.

(ii) For BCNF the functional dependencies should be such that left side of FD is key.

$R_1(\underline{\text{Eid}}, \text{Ename}, \text{Age})$

$R_2(\underline{\text{Eid}}, \text{Pno.}, \text{Policyno.}, \text{Email})$

Eid uniquely determines Ename and Age in  $R_1$

In  $R_2$  there is no redundancy due to functional dependency so it is in BCNF (redundancy due to multivalued functional dependency allowed).

(iii)  $R_1(\underline{\text{Eid}}, \text{Ename}, \text{Age})$

$R_2(\underline{\text{Eid}}, \text{Pno.})$

$R_3(\underline{\text{Eid}}, \text{Policyno.})$

$R_4(\underline{\text{Eid}}, \text{Email})$

There are no multivalued dependencies, so all four tables are also in 4NF.

Hence 4 tables are sufficient.

**T2 : Solution****(c)**

R (A, B, C, D, E)

The given FD's are  $ABC \rightarrow DE$  and  $D \rightarrow AB$  because  $C \rightarrow C$  is a trivial FD so  $DC \rightarrow ABC$  will also be a FD.  
 $\therefore$  DC will be a candidate key and ABC is another candidate key.

The following are the super key's possible listed below.

- |          |         |
|----------|---------|
| 1. ABCDE | 2. ABCD |
| 3. ABC   | 4. ABCE |
| 5. DC    | 6. DCA  |
| 7. DCB   | 8. DCAE |
| 9. DCE   | 10. DCB |

$\therefore$  10 super key's are possible.

**T3 : Solution****(c)**

As per definition of DML (Data Manipulation Language), it is used for selecting, inserting, deleting and updating data in database.

Hence option (c) is true.

**T4 : Solution****(c)**

Inserting tuples, deleting tuples is work of DML i.e. (Data Manipulation Language).

**T5 : Solution****(a)**

Views in a database system are important because of the following reasons:

- (i) They help provide data independence.
- (ii) They help with access control by allowing users to see only a particular subset of the data in database.

**T6 : Solution**

The key for R will be AF

$\therefore$  A and F individually cannot determine all attributes alone, but AF can determine ABCDEFGH using given FD's.

**Alternate**

$EFG \leftarrow H$  can be removed

$\therefore$  F itself can determine G and H, so  $FG \rightarrow H$  is redundant and E is determined by A itself.

So for AF to be key for above FD we don't need  $EFG \rightarrow H$ .

**T7 : Solution****(b)**

Journal (Volume, Number, Startpage, Endpage, Title, Year, Price)

**Primary key:** Volume, Number, Startpage, Endpage**FD's:** Volume Number Startpage Endpage → Title

Volume number → Year

Volume Number, Startpage Endpage → Price

Given relation 1NF but not 2NF. This DB is redesigned following schemas

 $R_1$ (Volume, Number Startpage Endpage Title Price) which has FD's

Volume Number, Startpage Endpage → Title

Volume Number Startpage Endpage → Price

Which is in BCNF.

 $R_2$ (Volume, Number, Year)

Volume Number → Year

Which is also in BCNF.

Journal in 1NF

 $R_1 R_2$  in BCNFWeakest NF which satisfy  $R_1$  and  $R_2$  and fails for journal is 2NF.

# 2

## Normalization



### Detailed Explanation of Try Yourself Questions

#### T1 : Solution

- (a)  $R(ABCD)$ ,  $FD = \{AB \rightarrow C, C \rightarrow D, D \rightarrow A\}$  are keys:  $AB, BD, BC$   
 $AB \rightarrow C$  :  $AB$  is a key  $\Rightarrow$  BCNF  
 $C \rightarrow D$  :  $D$  is key attribute  $\Rightarrow$  3NF  
 $D \rightarrow A$  :  $A$  is key attribute  $\Rightarrow$  3NF  
 $\therefore R$  is in 3NF
- (b)  $R(ABCD)$ ,  $FD = \{B \rightarrow C, B \rightarrow D\}$   $AB$  is a key  
 $B \rightarrow C$  :  $B$  is partial dependency.  
 $B \rightarrow D$  :  $B$  is partial dependency.  
 $\therefore R$  is in 1NF
- (c)  $R(ABCD)$ ,  $FD = \{AB \rightarrow C, BC \rightarrow D, CD \rightarrow A, AD \rightarrow B\}$  keys:  $AB, BC, CD, AD$   
 $\therefore$  All FD's are in BCNF (LHS is a key)  
 $\Rightarrow R$  is in BCNF
- (d)  $R(ABCD)$ ,  $FD = \{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A\}$  keys  $A, B, C, D$   
All FD's are in BCNF  
 $\Rightarrow R$  is in BCNF
- (e)  $R(ABCDE)$   $\{AB \rightarrow C, DE \rightarrow C, B \rightarrow D\}$  key:  $ABE$   
 $AB \rightarrow C$  : Partial dependency  $\Rightarrow$  1NF  
 $DE \rightarrow C$  : Transitive dependency  $\Rightarrow$  2NF  
 $B \rightarrow D$  : Partial dependency  $\Rightarrow$  1NF  
 $\therefore R$  is in 1NF

- (f) R(ABCDE): FD = {AB → C, C → D, B → D, D → E} key AB  
 AB → C : is in BCNF  
 C → D : Transitive dependency ⇒ 2NF  
 B → D : Partial dependency ⇒ 1NF  
 D → E : Transitive dependency ⇒ 2NF  
 ∴ R is in 1NF
- (g) R(ABCDE) FD = {A → B, B → C, C → D, D → A} keys = AE, BE, CE, DE  
 A → B : B is prime attribute  
 B → C : C is prime attribute  
 C → D : D is prime attribute  
 D → A : A is prime attribute  
 ∴ R is in 3NF

**T2 : Solution**

- (a) R(ABCD), FD = {AB → C, C → D, D → A} are keys: AB, DB, CB  
 AB → C : is in BCNF  
 C → D : is in 3NF violates BCNF  
 D → A : is in 3NF violates BCNF  
 ∴ R is in 3NF  
 BCNF decomposition : {CD, DA, BC}  
 Not dependency preserving and fails to preserve AB → C dependency
- (b) R(ABCD), FD = {B → C, B → D} AB is a key  
 B → C : is violates 2NF  
 B → D : is violates 2 NF  
 2NF decomposition = {B, CD, AB}  
 3NF decomposition = {BCD, AB}  
 BCNF decomposition = {BC, BD, AB} and it preserve dependency.
- (c) R(ABCD), FD = {AB → C, BC → D, CD → A, AD → B} keys: AB, BC, CD, AD  
 All FD's are in BCNF  
 No decomposition is required.
- (d) R(ABCD), FD = {A → B, B → C, C → D, D → A} keys A, B, C, D  
 All FD's are in BCNF  
 ∴ R is in BCNF, hence no decomposition required.
- (e) R(ABCDE) FD = {AB → C, DE → C, B → D} key: ABE  
 AB → C : Violates 2NF (Partial dependency)  
 DE → C : Violates 3NF (Transitive dependency)  
 B → D : Violates 2NF  
 2NF decomposition : {ABC, BD, ABE}  
 3NF decomposition : ?  
 Canonical cover of FD = {ABE → C}  
 3NF decomposition = {ABEC, ABDE}  
 BCNF decomposition : {ABC, DEC, BD, ABE}

- (f) R(ABCDE): FD = {AB → C, C → D, B → D, D → E} key AB  
C → D : Violates 3NF  
B → D : Violates 2NF  
D → E : Violates 3NF  
2NF decomposition = {BDE, ABC}  
3NF decomposition = ?  
Canonical FD = {BE, ABCD}  
3NF decomposition = {BE, ABCD}  
BCNF decomposition = {CD, BD, DE, ABC}

**T3 : Solution**

- (d)  
Normalization is to eliminate redundant data stored in the database.  
It reduces the anomalies.  
Statement 1 and 3 are correct.

■ ■ ■ ■

# 3

## Relational Algebra, Tuple Relational Calculus and SQL



### Detailed Explanation of Try Yourself Questions

#### T5 : Solution

(a)

$R \cup R = R$  always true (since union eliminates duplicate)

#### T6 : Solution

(a)

$R(A, B)$  that has no null values,  $R$  has  $n$  tuples

Select  $rr.A, rr.B, ss.A, ss.B$

from  $B$  as  $rr, R$  as  $ss$

where  $rr.A = ss.A$  and  $rr.B = ss.B$

**Condition follows on output results 'm'**

When all tuples are same then maximum  $n^2$  elements are output.

When all tuples are distinct then minimum  $n$  elements are present as output.

So,  $n \leq m \leq n * n$

#### T7 : Solution

(d)

$R(A, B) S(A, B)$

Elements are not distinct. If  $m$  is result of  $R \cap S$  then  $m$  should follow condition.

Maximum result is  $\min(r, s)$  when both having same element and size of 1<sup>st</sup> is less than 2<sup>nd</sup> or vice-versa.

Minimum result 0 when no element is common.

i.e.,  $0 \leq m \leq \min(r, s)$



**T8 : Solution**

(d)

- SQL not permit attribute name to be repeat in same relation.
- SQL query done not eliminate duplicate automatically.
- SQL can work if then is no index on the relation i.e., when relation size is small then no need of index.

**T9 : Solution**

(c)

Two relations R(A, B) S(A, B) name exact same schema.

1.  $R \cap S = R - (R - S)$  true.
2.  $R \cap S = S - (S - R)$  true.
3.  $R \cap S = R \bowtie S$  ( $\bowtie$  on both attribute) so true.
4.  $R \cap S = R \times S$  false since  $R \times S \supset R \cap S$

So, 1, 2, 3 are correct.

**T10 : Solution**

(c)

X	Y	X	Y
1	2	1	2
1	2	1	2
2	3	2	3
3	4	3	4
3	4	3	4
4	1	4	1
4	1	4	1
4	1	4	1
4	2	4	2

```
select a1.x, a2.y, count (*)
from Arc a1, Arc a2
where a1.y = a2.x
group by a1.x, a2.y
```

a <sub>1</sub> .x	a <sub>2</sub> .y	count
1	3	2
2	4	2
3	2	4
3	1	6
4	2	6

So only (1 3 2) and (3 1 6) are present.

**T11 : Solution****(c)**

Relation  $R(a, b)$  may contain duplicate tuple:

1. select  $a$  from  $R$  where  $a = 1$  contain duplicate.
2. select  $\text{Max}(b)$  from  $R$  group by  $a$  does contain duplicate.
3. select  $a, b$  from  $R$  group by  $a, b$  does not contain duplicate.
4. select  $a$

from  $R$

where  $a$  not in (select  $a$   
from  $R$ )

result empty always.

**T12 : Solution****(c)**

Two relations  $r(R)$  and  $S(S)$ . Where  $R$  and  $S$  are schema of relation.

Then  $r \div S$  relation on schema  $R - S$  with  $S \subseteq R$ .

So option 1 and 2 are correct only.

**T13 : Solution****(c)**

Tuple relational calculus is non-procedural i.e., we know what to retrieve but don't know how to retrieve.

- Declarative DML is non-procedural query language.
- SQL is non-procedural query language.
- In procedural query language user instruct the system that what to get and how to get i.e., relational algebra.

So 1 and 4 are true only.

**T14 : Solution****(a)**

Operation  $A - B$

$A$  and  $B$  are union compatible and all rows of  $A$  are common to  $B$ .

$A$  with 4 column and 20 rows  $B$  with 4 column and 15 rows.

The number of rows and column are 0 and 4. Since column will remain but no row will be present i.e.,

$A \{1, 1, 2, 2, 3\} - \{1, 2, 3\} = \{\phi\}$

**T15 : Solution**

(a)

Select A

from  $T_1, T_2$

where  $T_1.C = T_2.C$

tuple which pass the selection condition

A	B	C	C	D
1	2	5	5	8
2	4	5	5	8
4	3	5	5	8
1	2	5	5	6
2	4	5	5	6
4	3	5	5	6
1	2	5	5	7
2	4	5	5	7
4	3	5	5	7
1	3	4	4	7
4	8	9	9	7
4	8	9	9	8
3	6	8	8	6
3	7	8	8	6

Now select  $t_A = \{1, 2, 4, 1, 2, 4, 1, 2, 4, 1, 4, 4, 3, 3\}$

Since no need of distinct element.

So answer is 14.

(b)

$\pi_A(T_1 \bowtie T_2)$

A	B	C	C	D
1	2	5	5	8
2	4	5	5	8
4	3	5	5	8
1	2	4	5	6
2	4	5	5	6
4	3	5	5	6
1	2	4	5	7
2	4	5	5	7
4	3	5	5	7
1	3	4	4	7
4	8	9	9	7
4	8	9	9	8
3	6	8	8	6
3	6	8	8	6

$\pi_A$  select only distinct value of A. So output is  $\{1, 2, 4, 3\}$

So answer is 4.

Sum of tuples from both queries =  $14 + 4 = 18$  and  $n_1 > n_2$ .

**T16 : Solution**

1.  $\{t \mid \exists p \in R (t[A] = p[A])\}$
2.  $\{t \mid t \in R \wedge (B = 17)\}$
3.  $\{t \mid p \in R \exists q \in S [t(A) = p(A) \wedge t(A) = p(B) \wedge t(C) = p(C) \cap t(D) = q(D) \wedge t(F) = q(E) \wedge t(F) = q(F)]\}$
4.  $\{t \mid \exists p \in R \exists q \in S [t(A) = p(A) \wedge t(B) = p(B) \wedge t(C) = p(C) \wedge t(D) = q(D) \wedge t(E) = q(E) \wedge t(F) = q(F) \wedge P(C) = q(D)] \wedge t(A) = p(A) \wedge t(F) = q(F)\}$

**T17 : Solution**

(2)

Total	Name	Capacity	is result of first query.
	Ajmer	20	
	Bikaner	40	
	Churu	30	
	Dungargarh	10	

Total average	Capacity	is result of second subquery.
	25	

Select name  
from Total, Total\_Avg  
Where total capacity  $\geq$  Total\_Avg capacity  
Query results two records.



# 4

## Transaction and Concurrency Control



### Detailed Explanation of Try Yourself Questions

#### T1 : Solution

$S_1 : r_1(A), r_2(A), r_3(A), w_1(B), w_2(B), w_3(B)$

For the schedule to be view serializable it must satisfy the following conditions: (1) Final Write (2) Initial Read and (3) WR Sequence.

#### Final Write

For data item A : No write operations.

For data item B :  $T_1, T_2, T_3$  (order of WRITE operation on data item B)

Therefore  $[T_1, T_2] \xrightarrow[\text{before } T_3]{\text{should execute}} T_3$

#### Initial Read

No write operation on A as well as no read operation on B. Hence this condition do not specify any order of execution.

#### WR Sequence

No such sequence. Therefore no condition on order of execution.

∴ The following are the view equivalent schedules.

$$\begin{aligned} T_1 \rightarrow T_2 \rightarrow T_3 \\ T_2 \rightarrow T_1 \rightarrow T_3 \end{aligned}$$

$S_2 : r_1(A), r_2(A), r_3(A), r_4(A), w_1(B), w_2(B), w_3(B), w_4(B)$

As we can see this schedule is similar to previous schedule and "INITIAL\_READ" and "WE-SEQUENCE" do not give any order. The only conditions  $[T_1, T_2, T_3]$  should execute before  $T_4$ .

$$\begin{aligned} T_1 \quad T_2 \quad T_3 \quad T_4 \\ T_1 \quad T_3 \quad T_2 \quad T_4 \\ T_2 \quad T_1 \quad T_3 \quad T_4 \\ T_2 \quad T_3 \quad T_1 \quad T_4 \\ T_3 \quad T_1 \quad T_2 \quad T_4 \\ T_3 \quad T_2 \quad T_1 \quad T_4 \end{aligned}$$

$S_3: r_1(A), r_3(D), w_1(B), r_2(B), w_3(B), r_4(B), w_2(C), r_5(C), w_4(E), r_5(E), w_5(B)$

**Final write:**

**A:** No WRITE operation

**B:**  $T_1 T_3 T_5$  i.e.  $[T_1, T_3] \rightarrow T_5$

**C:**  $T_2$

**D:** No write operation on D

**Initial Read:**

**A:** Only  $T_1$  reads, but no update

**B:** No initial read operation

**C:** No initial read operation

**D:** Only  $T_3$  reads but no update operation

**E:** No initial read

$\therefore$  No condition on order of execution.

**WR Sequence:**

**A:** No updation on A

**B:**  $T_1 \rightarrow T_2$

$T_2 \rightarrow T_3$

$T_3 \rightarrow T_4$

**C:**  $T_2 \rightarrow T_5$

**D:** No updation D

**E:**  $T_4 \rightarrow T_5$

$\therefore$  Therefore only one serial schedule is view equivalent.

$$T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_4 \rightarrow T_5$$

$S_4: w_1(A), r_2(A), w_3(A), r_4(A), w_5(A), r_6(A)$

Based on WR sequence there is only one serial schedule which is view equivalent.

$$T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_4 \rightarrow T_5 \rightarrow T_6$$

$S_5: r_2(A), r_1(A), w_1(C), r_3(C), w_1(B), r_4(B), w_3(A), r_4(C), w_2(D), r_2(B), w_4(A), w_4(B)$

**WR Sequence**

**B:**  $T_1 \rightarrow T_4, T_1 \rightarrow T_2$

**C:**  $T_1 \rightarrow T_3$

**Final Write**

**A:**  $T_3, T_4$  i.e.  $T_3 \rightarrow T_4$  ( $T_3$  followed by  $T_4$ )

**B:**  $T_1, T_4$  i.e.  $T_1 \rightarrow T_4$

**Initial Read**

**A:**  $T_1, T_2$  reads it initially and later updated by  $T_3$  and  $T_4$ .

$\therefore (T_1, T_2) \rightarrow (T_3, T_4)$

**B:** No initial reads

**C:** No initial reads

**D:** No initial reads

$\therefore$  Based on the all the above conditions there's only one serial schedule which is view equivalent

i.e.  $T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow T_4$

**T2 : Solution**

**S<sub>1</sub> :**

T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>
r(x)		
w(x)	r(z)	
		r(x)
w(x)		r(y)
C <sub>1</sub>		w(y)
		C <sub>3</sub>
	r(y)	
	w(z)	
	w(y)	
	C <sub>2</sub>	

Strict recoverable  
Cascadeless  
Recoverable

**S<sub>2</sub> :**

T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>
r(x)		
r(z)	r(z)	
		r(x)
w(x)		r(y)
C		w(y)
	r(y)	
	w(z)	
	w(y)	
C <sub>1</sub>	C <sub>2</sub>	C <sub>2</sub>

Not strict recoverable  
Irrecoverable  
Cascadeless

**S<sub>3</sub> :**

T <sub>1</sub>	T <sub>2</sub>	T <sub>3</sub>
r(x)		
	r(z)	r(x)
r(z)		
	r(y)	r(y)
w(x)		
C <sub>1</sub>		
	w(z)	
	w(y)	w(y)
	C	C <sub>3</sub>
	C <sub>2</sub>	

No strict  
Recoverable  
Cascadeless

**T3 : Solution**

(1)

No conflict serializable  
Not recoverable concept here  
View serializable  $T_1 \rightarrow T_2$   
No cascade abort  
No cascade abort  
Not strict recoverable

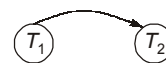
T <sub>1</sub>	T <sub>2</sub>
r(x)	
w(x)	r(x)
	w(x)



(2)

Conflict serializable  
View serializable  $T_1 \rightarrow T_2$   
Not recoverable concept  
Not strict recoverable  
Not cascade abort  
Serializable

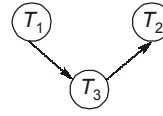
T <sub>1</sub>	T <sub>2</sub>
w(x)	
r(y)	r(y)
	r(x)



**(3)**

- Conflict serializable
- Not recoverable
- Not strict
- No cascadeless
- Conflict serializable  $T_1 T_3 T_2$
- View serializable
- $T_3 \rightarrow T_2 T_1$
- $T_1 T_3 \rightarrow T_2$
- $T_3 \cdot T_1 \rightarrow T_2$

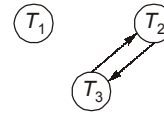
$T_1$	$T_2$	$T_3$
$r(x)$		
	$r(y)$	
	$r(x)$	$w(x)$
$r(y)$		



**(4)**

- No conflict serializable
- Not view serializable
- $T_3 \rightarrow T_2 \rightarrow$  Not possible b/c  $R(y)$  is initial read
- No recoverable
- No cascadeless
- Not strict

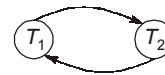
$T_1$	$T_2$	$T_3$
$r(x)$		
$w(x)$	$r(y)$	
	$r(y)$	
$w(x)$		$w(y)$
	$r(y)$	



**(5)**

- Not conflict
- View serializable not possible
- $T_2 \rightarrow T_1$  but  $T_1$  come first since initial read
- Recoverable
- Not strict recoverable
- Not cascadeless
- Not serializable

$T_1$	$T_2$
$r(x)$	
$w(x)$	$w(x)$
	abort
commit	



**(6)**

- Not conflict serializable
- Not view serializable
- $T_2 \rightarrow T_1$
- $T_2 \rightarrow T_1$
- Not recoverable
- Not cascadeless
- Not strict
- Not serializable

$T_1$	$T_2$
$r(x)$	
$w(x)$	$w(x)$
	$r(x)$
commit	commit



**(7)**

- Not conflict
- View serializable  $T_1 \rightarrow T_2$
- Recoverable
- Not strict
- Cascadeless
- Serializable

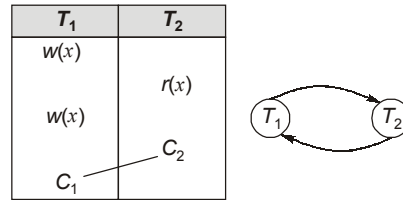
$T_1$	$T_2$
$w(x)$	
$w(x)$	$r(x)$
	$r(x)$
$C_1$	





(8)

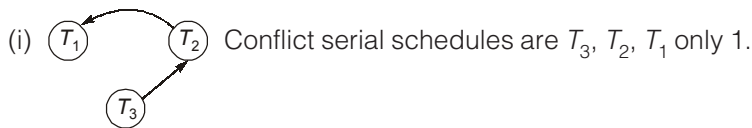
- Not conflict
- Not view serializable
- Not recoverable
- Not cascadeless
- Not serializable
- Not strict



**T4 : Solution**

(a)

$T_1$	$T_2$	$T_3$
$r(a)$		
	$r(b)$	
		$r(c)$
$w(b)$		
	$w(c)$	
		$w(d)$



- (ii) Number of view equal serial schedules:
- $T_2 \rightarrow T_1$  and  $T_3 \rightarrow T_1$
- So  $T_3 \rightarrow T_2 \rightarrow T_1$

(iii)

$T_1$	$T_2$	$T_3$
$s(a)$		
$r(a)$		
	$s(b)$	
	$s(c)$	
	$r(b)$	
	unlock (B)	
		$s(c)$
		$X(d)$
		$r(c)$
		unlock (c)
$X(b)$		
$w(b)$		
unlock (b)		
unlock (a)		
	$X(c)$	
	$w(c)$	
		$w(d)$

cannot get  $X(c)$

No allowed because  $T_2$  is in shrinking phase

Schedule is not allowed by 2PL  
Not allowed by strict 2PL.

(iv) If  $(T_1, T_2, T_3) = (10, 20, 30)$

Set of rollbacks are  $(T_1 \rightarrow T_1)$  for  $W(B) \rightarrow R(B)$ ,  $(T_2 \rightarrow T_3)$  for  $W(c) \rightarrow R(c)$ .

So transaction  $T_1$  and  $T_2$  are rollback.

- If  $(T_1, T_2, T_3) = (30, 20, 10)$

No rollback i.e., time-stamp ordering is  $T_3 \rightarrow T_2 \rightarrow T_1$ .

- If  $(T_1, T_2, T_3) = (20, 10, 30)$

Transaction  $T_2$  is rollback on C.

- If  $(T_1, T_2, T_3) = (30, 10, 20)$

Transaction  $T_2$  is rollback on C.

- If  $(T_1, T_2, T_3) = (20, 30, 10)$

Transaction  $T_1$  is rollback on B.

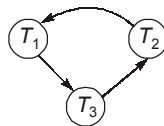
- If  $(T_1, T_2, T_3) = (10, 30, 20)$

Transaction  $T_1$  is rollback on B.

**(b)  $r_1(a), r_2(b), r_3(c), w_1(b), w_2(c), w_3(a)$**

$T_1$	$T_2$	$T_3$
$r(a)$	$r(b)$	$r(c)$
$w(b)$	$w(c)$	$w(a)$

**(i) Conflict serializable**



Since, there is a cycle in the precedence graph, hence the schedule is not conflict serializable.

**(ii) View serializable**

For A  $\rightarrow T_1 \rightarrow T_3$

For B  $\rightarrow T_2 \rightarrow T_1$

For C  $\rightarrow T_3 \rightarrow T_2$

Hence, none of the schedule can lead to a view serializable schedule.

**(iii) Basic 2PL**

$T_1$	$T_2$	$T_3$
$S(a)$	$S(b)$	
$r(a)$	$r(b)$	
$w(b)$	$w(c)$	$S(c)$ $r(c)$
		$w(a)$

No basic 2PL possible.

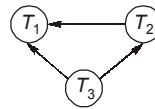
(iv) Strict 2PL

$T_1$	$T_2$	$T_3$
s(A) r(A) unlock(A)	s(C) r(B) S(b) unlock(b)	S(a) S(c) r(c) Unlock(c)
X(B) W(B) unlock(b)	X(C) W(C) unlock(c)	X(A) W(A) unlock(A)

(c)  $r_1(A), r_2(B), r_3(C), r_1(b), r_2(C), r_3(d), w_1(C), w_2(D), w_3(E)$

(i) Conflict serializable

$T_1$	$T_2$	$T_3$
r(A)	r(B)	r(C)
r(b)	r(c)	r(d)
w(c)	w(d)	w(e)



∴ Conflict serializable.

Order  $\Rightarrow T_3 \rightarrow T_2 \rightarrow T_1$

(ii) View serializable:  $T_3 \rightarrow T_2 \rightarrow T_1$

∴ Because, on data item 'C' initial read is by  $T_3$  and final write is by  $T_1$ . Hence,  $T_3$  should be followed by  $T_1$ .

∴ Because, on data item 'd' initial read is by  $T_3$  and final write is by  $T_2$ .

Hence  $T_3$  should be followed by  $T_2$ .

(iii) Basic 2PL

$T_1$	$T_2$	$T_3$
S(a), S(b) r(a)	S(b), S(c) r(b)	S(c), S(d), X(e) r(c)
r(b)	r(c)	r(d)
Not allowed X(c) w(c)	w(d)	w(e)

∴ Not allowed under basic 2PL.

(iv) Strict 2PL:

$T_1$	$T_2$	$T_3$
$S(a), S(b)$ $r(a)$ unlock (a)	$S(b), S(c)$ $r(b)$ unlock (b)	
		$S(b), S(d)$ $r(c)$ unlock (c)
$r(b)$ unlock (b)	$r(c)$ unlock (c)	
$X(c)$ $w(c)$ unlock (c)		$r(d)$ unlock (d)
	$w(d)$ $X(d)$ unlock (d)	
		$w(e)$ $X(e)$ unlock (e)

(d)  $r_1(A), r_2(B), r_3(C), r_1(B), r_2(C), r_3(D), w_1(A), w_2(A), w_3(C)$

(i) Conflict serializable

$T_1$	$T_2$	$T_3$
$r(a)$		
	$r(b)$	
$r(b)$		$r(c)$
	$r(c)$	
$w(a)$		$r(d)$
	$w(a)$	
		$w(c)$

∴ Schedule is conflict-serializable.

(ii) View serializable

Hence only for data item 'a', initial read is done by  $T_1$  and final write is done by  $T_2$ . Hence  $T_1$  should be followed by  $T_2$ .

For any other data item, there is no such dependency. Hence, the schedule possible under view serializable are:  $T_3, T_1, T_2, T_1, T_3, T_2, T_1, T_2, T_3$

(iv) Strict 2PL

$T_1$	$T_2$	$T_3$
$X(a)$ $r(a)$	$S(b)$ $r(b)$	$S(c), S(d)$ $r(c)$ unlock (c)
$S(b)$ $r(b)$ unlock (b)	$S(c)$ $r(c)$ unlock (c)	$r(d)$ unlock (d)
$w(a)$ unlock (b)	$X(a)$ $w(a)$ unlock (a)	$X(c)$ $w(c)$ unlock (c)

(iii) Basic 2PL

$T_1$	$T_2$	$T_3$
$r(a)$	$S(b), S(c)$ $r(b)$	$S(c), S(d)$ $r(c)$
$r(b)$	$r(c)$	$r(d)$
$w(a)$ unlock (b) unlock (a)	$X(a)$ $w(a)$ unlock (a) unlock (b) unlock (c)	$X(c)$ $w(c)$ unlock (c) unlock (d)

∴ Allowed under basic 2PL.

(e)  $r_1(A), r_2(B), r_3(C), r_1(B), r_2(c), r_3(A), w_1(A), w_2(B), w_3(C)$

(i) Conflict serial schedule

$T_1$	$T_2$	$T_3$
$r(a)$		
	$r(b)$	
		$r(c)$
$r(b)$		
	$r(c)$	
		$r(a)$
$w(a)$		
	$w(b)$	
		$w(c)$

$\therefore$  Conflict serializable schedule:  $T_3, T_1, T_2, T_1, T_3, T_2, T_1, T_2, T_3$

(ii) View serializable

Since the schedule is conflict serializable hence view serializable two.

(iii) Basic 2PL

$T_1$	$T_2$	$T_3$
$X(a), s(b)$ $r(a)$		
	$S(b), s(c)$ $r(b)$	
		$s(c)$ $r(c)$
$r(b)$		
	$r(c)$	$s(a)$ $r(a)$
$w(a)$		$w(c)$

not possible

$\therefore$  Hence, schedule is not possible under basic 2PL.

(iv) Strict 2PL

$T_1$	$T_2$	$T_3$
$s(a), s(b)$ $r(a)$		
	$s(b), s(c)$ $r(b)$	
		$s(c)$ $r(c)$
$r(b)$ unlock (b)		
	$r(c)$ unlock (c)	
		$s(a)$ $r(a)$ unlock (a)
$X(a)$ $w(a)$ unlock (a)		
	$X(b)$ $w(b)$ unlock (b)	
		$X(c)$ $w(c)$ unlock (c)

**T5 : Solution**

(c)

Concurrency control ensures isolation of the transactions.

Recovery management is responsible for Atomicity. Application manager (user) ensures consistency.

**T6 : Solution**

(c)

Ensuring consistency for an individual transaction is the responsibility of application programmer.

**T7 : Solution**

(b)

Every view serializable is not conflict serializable i.e., when schedule is view serializable and there is a write-write conflict then it is not conflict serializable.

**T9 : Solution**

(c)

Wait-die scheme for deadlock prevention is non-preemptive because when transaction P1 request data item currently held by transaction P2 then P1 is allowed to wait only if it has time-stamp smaller than P2 otherwise P1 is die.

Bound-wait preemptive because when transaction P1 request data item currently held by transaction P2 then P1 is allowed to wait only if it has time-stamp larger than P2 otherwise P1 is rollback.

So option (2) and (3) is wrong.

**T10 : Solution**

(c)

Schedule allowed by 2PL may lead to deadlock.

Schedule allowed by 2PL are free from cascading rollback and lost update problem.

**T11 : Solution**

(a)

A schedule allowed by basic time stamp ordering protocol is free from deadlock and not free from cascading rollback problem with irrecoverability.

**T12 : Solution**

(c)

No uncommitted reads so that its cascadeless rollback recoverable

because  $T_1$   $w_1(x)$  before  $T_1$  commit / Rollback  $T_2$   $w_2(x)$ .

So not strict recoverable.

$T_1$	$T_2$
$r_1(x)$	$r_2(x)$
	$r_2(y)$
$w_1(x)$	
$r_1(y)$	
$a_1$	$w_2(x)$
	$a_2$



# 5

## File Structure and Indexing



### Detailed Explanation of Try Yourself Questions

#### T1 : Solution

(a)

File size = 10,000

Block size = 1024

Number of entries in 1<sup>st</sup> level of Dense Index = Number of records in file.

$$\therefore \text{Number of index blocks} = \frac{10000}{\text{Block factor of index block}}$$

$$\text{Block factor of index block} = \frac{1024}{(9+7)} = 64$$

$$\therefore \text{Number of index 1<sup>st</sup> level blocks} = \left\lceil \frac{10000}{64} \right\rceil = 157 \text{ blocks}$$

(b) Number of entries in sparse index = Number of blocks in file

$$\therefore \text{Number of blocks in file} = \frac{10000}{\text{Block factor in file}}$$

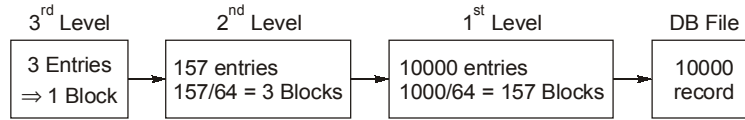
$$\text{Block factor} = \frac{1024}{100} = \left\lceil \frac{1024}{100} \right\rceil = 11 \text{ records/block}$$

$$\therefore \text{Number of blocks} = \frac{10000}{11} = 910 \text{ blocks.}$$

$$\text{Number of index blocks} = \frac{910}{64} = \lceil 14.21 \rceil = 15$$

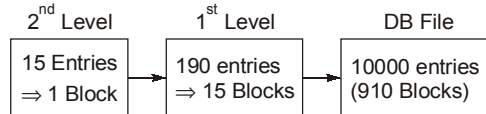


(c) Number of levels in dense index



∴ Three levels are required.

(d) Number of levels in sparse index



**T2 : Solution**

(a) File size = 30000 records  
Number of entries in 1<sup>st</sup> level dense index = Number of records in database

$$\text{Index block factor} = \frac{1024}{15} = 69$$

$$\text{Number of 1<sup>st</sup> level index blocks} = \frac{30000}{69} = 435 \text{ blocks}$$

$$\text{Number of 2<sup>nd</sup> level index blocks} = \frac{435}{64} = 70 \text{ blocks}$$

Number of 3<sup>rd</sup> level index blocks = 1 (7 entries only to be entries in index block).

(b) Number of entries in 1<sup>st</sup> level secondary index = Number of blocks in file

$$\text{Number of blocks in file} = \frac{30000}{\text{Block factor of block}}$$

$$\text{Block factor} = \frac{1024}{100} = 11$$

$$\text{Number of blocks in file} = \frac{30000}{11} = 2728 \text{ blocks}$$

$$\text{Number of blocks in 1<sup>st</sup> level} = \frac{2728}{\text{Block factor of index block}} = \frac{2728}{69} = [39.5] = 40$$

Number of blocks in 2<sup>nd</sup> level = 1 (only 40 entries per index block and block factor is 69)

(c) 3 levels.

(d) 2 levels.

**T3 : Solution**

(a) Database file size = 1250 records

Dense index is used

**For minimum**

$$\text{Number of index blocks in 1st level} = \frac{1250}{10} = 125 \text{ blocks}$$

$$\text{Number of index blocks in 2nd level} = \frac{125}{11} = 12 \text{ blocks}$$

$$\text{Number of index blocks in 3rd level} = \frac{12}{11} = 2 \text{ block needed.}$$

Number of index blocks in 4th level = Only 1 block needed.

∴ By using Dense index minimum 140 index blocks and 4 index levels are required.

(b) **For maximum**

$$\text{Number of index blocks in 1st level} = \frac{1250}{5} = 250 \text{ blocks}$$

$$\text{Number of index blocks in 2nd level} = \frac{250}{6} = 42 \text{ blocks}$$

$$\text{Number of index blocks in 3rd level} = \frac{42}{6} = 7 \text{ block needed.}$$

$$\text{Number of index blocks in 4th level} = \frac{7}{6} = 2 \text{ blocks}$$

Number of index blocks in 5th level = Only 1 block needed.

∴ Using Dense index maximum 302 index blocks and 5 index levels are required.

(c) Sparse index and minimum (maximum filling in nodes)

$$\text{Number of blocks in file} = \frac{1250}{3} = 417 \text{ blocks}$$

$$\text{Minimum number of index blocks at 1st level} = \frac{417}{10} = 42 \text{ (leaf level)}$$

$$\text{Minimum number of index blocks at 2nd level} = \frac{42}{11} = 4$$

Minimum number of index blocks at 3rd level = Only 1 block is needed and minimum 3 levels are required.

(d) Sparse index and Maximum (minimum filling in nodes)

$$\text{Number of index blocks in 1}^{\text{st}} \text{ index file} = \frac{417}{5} = 84 \text{ (leaf levels)}$$

$$\text{Number of index blocks in 2}^{\text{nd}} \text{ index file} = \frac{84}{6} = 14$$

$$\text{Number of index blocks in 3}^{\text{rd}} \text{ index file} = \frac{14}{6} = 3$$

Number of index blocks in 4<sup>th</sup> index file = 1 block

∴ Using sparse index 102 blocks and 4 levels.

**T4 : Solution**

(a)

B<sup>+</sup> tree records are stored in primary order. B<sup>+</sup> tree does not use hashing because it's not possible to answer range queries using hashing.

Updates do not cause unbalance in the tree.

**T5 : Solution**

(c)

The maximum number of new nodes created is "number of levels + 1".

In the given case the number of levels are four (including root).

Hence maximum number of new nodes (created are 5)

**T6 : Solution**

(b)

A data dictionary contains a list of all files in this database, the number of records in each file, and the names and types of each field.

Data database, only book-keeping information for managing it.

**T7 : Solution**

(a)

B<sup>+</sup> tree balanced because the length of the paths from the root to all leaf nodes are all equal and every internal node must be filled by.

