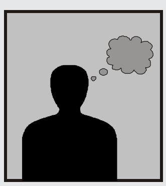
GATE

MADE EASY Workbook 2026



Detailed Explanations of Try Yourself *Questions*

Computer Science & IT

Databases

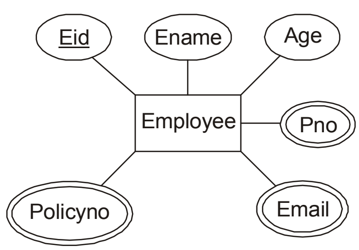
MADE EASY PUBLICATIONS

1 Introduction to DBMS and Integrity Constraints and ER Model

Detailed Explanation of Try Yourself Questions

T1 : Solution

(*i*)



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 (<u>Eid</u>, Ename, Age, <u>Eid, Pno., Policyno., Email</u>)

Therefore, 1 table is required.

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

R_1 (<u>Eid</u>, Ename, Age)

R_2 (<u>Eid, Pno., Policyno., Email</u>)

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 (<u>Eid</u>, Ename, Age)

R_2 (<u>Eid, Pno.</u>)

R_3 (<u>Eid, Policyno.</u>)

R_4 (<u>Eid, Email</u>)

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 is database.

T6 : Solution

The key for R will be AF

$\because$ 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

$\because$ 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 $\rightarrow$ Title

Volume number $\rightarrow$ Year

Volume Number, Startpage Endpage $\rightarrow$ 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 $\rightarrow$ Title

Volume Number Startpage Endpage $\rightarrow$ Price

Which is in BCNF.

R_2(Volume, Number, Year)

Volume Number $\rightarrow$ Year

Which is also in BCNF.

Journal in 1NF

R_1 R_2 in BCNF

Weakest NF which satisfy R_1 and R_2 and fails for journal is 2NF.

T8 : Solution

(b)

As R is referring to R_2 and S is primary key of R_2, $\pi_R(r_1) - \pi_S(r_2)$ will give empty relation or empty table as number of values in R column of table r_1 will always refer to of respective values in S column of r_2.

T9 : Solution

(4)

Minimum number of tables required are 4.

1. Bank {<u>Code, Ph No</u>, Name, Addr} where code, Ph_No. is primary key.
2. Branches {<u>Branch No, Code</u>} where branch number or code is the primary key.
3. Bank_Branch {Addr, <u>Branch_No</u>, Bank_Name}
4. Loan Taken {<u>Loan_No</u>, Amount, Type, Branch_No.} where <u>Loan No</u> as the primary key.

Here we can not merge Branches relation and Bank Branch entity because foreign key "code" is not the candidate key of Bank entity. So, we can not combined these two.

T10 : Solution

(5)

The RDBMS tables that are need to be drawn will be:

(*i*) $E_3\ R_2$ which have 'A' as its candidate key.

(*ii*) $E_2\ R_4$ which have 'D' as its candidate key.

(*iii*) $E_1\ R_3$ which have 'D' as its candidate key.

(*iv*) R_1 which has 'AD' as the candidate key.

(*v*) $E_3\,P$; since P is a multi-valued attribute, which has 'A' as its candidate key.

T11 : Solution

(50)

For relationship set R candidate and E_1 candidate key is same because between E_1 to E_3, E_1 to E_2 and E_1 to E_4 there is one to many relationship.

T12 : Solution

(5)

$$\begin{array}{c|c|c|c|c} (\underline{AC}) & (\underline{A}B) & (\underline{A}D) & (\underline{D}E) & \underline{DF} \\ & A \to B & & D \to E & \end{array}$$

Multivalued attribute always combines with key.

T13 : Solution

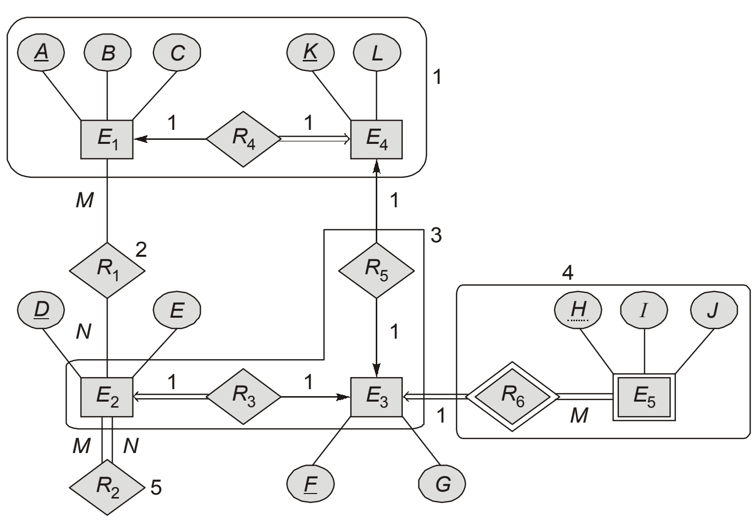
(2)

Because of partial participation E_1 must separate, $E_2 R$ can merge

$$\begin{bmatrix} E_1(\underline{A}BC) & E_2R(\underline{AD}EF) \\ A \to BC & D \to EF \\ & D \to A \end{bmatrix}$$

T14 : Solution

(b)



■■■■

2 Normalization

Detailed Explanation of Try Yourself Questions

T1 : Solution

(a) R(ABCD), FD = {$AB \to C$, $C \to D$, $D \to A$} are keys: AB, BD, BC

$AB \to C$: AB is a key $\Rightarrow$ BCNF

$C \to D$: D is key attribute $\Rightarrow$ 3NF

$D \to A$: A is key attribute $\Rightarrow$ 3NF

$\therefore$ R is in 3NF

(b) R(ABCD), FD = {$B \to C$, $B \to D$} AB is a key

$B \to C$: B is partial dependency.

$B \to D$: B is partial dependency.

$\therefore$ R is in 1NF

(c) R(ABCD), FD = {$AB \to C$, $BC \to D$, $CD \to A$, $AD \to B$} 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 = {$A \to B$, $B \to C$, $C \to D$, $D \to A$} keys A, B, C, D

All FD's are in BCNF

$\Rightarrow$ R is in BCNF

(e) R(ABCDE) {$AB \to C$, $DE \to C$, $B \to D$} key: ABE

$AB \to C$: Partial dependency $\Rightarrow$ 1NF

$DE \to C$: Transitive dependency $\Rightarrow$ 2NF

$B \to D$: 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}

T4 : Solution

(56)

$$A^+ = \{A\ B\ C\ D\ E\ F\}$$
$$B^+ = \{A\ B\ C\ D\ E\ F\}$$
$$D^+ = \{A\ B\ C\ D\ E\ F\}$$

Clearly, there are three candidate keys.

Thus, $N(A \cup B \cup D) = N(A) + N(B) + N(D) - \{N(A \cap B) + N(B \cap D) + N(A \cap D) + N(A \cap B \cap D)\}$

Where, $N(A \cap B \cap D)$ represents number of super keys where A or B or D is candidate keys.

$$\Rightarrow \quad N(A \cup B \cup D) = 3 \times 2^5 - 3 \times 2^4 + 2^3$$
$$= 96 - 48 + 8 = 56$$

T5 : Solution

(d)

$\boxed{A \to C}$	$\boxed{B \to C}$	$\boxed{A \to B}$	$B \to A$	$A \to B$
$A \to B$	$B \to A$	$A \to C$	$B \to C$	$A \to C$
$\boxed{B \to A}$	$\boxed{A \to B}$	$\boxed{B \to A}$	$\boxed{C \to A}$	$B \to A$
$B \to C$	$A \to C$	$B \to C$	$C \to B$	$\boxed{B \to C}$
$C \to A$	$\boxed{C \to A}$	$C \to A$	$A \to B$	$C \to A$
$\boxed{C \to B}$	$C \to B$	$C \to B$	$\boxed{A \to C}$	$\boxed{C \to B}$
$\Downarrow$	$\Downarrow$	$\Downarrow$	$\Downarrow$	$\Downarrow$
$\left\{\begin{array}{l} A \to B \\ B \to C \\ C \to A \end{array}\right\}$	$\left\{\begin{array}{l} B \to A \\ A \to C \\ C \to B \end{array}\right\}$	$\left\{\begin{array}{l} A \to C \\ B \to C \\ C \to AB \end{array}\right\}$	$\left\{\begin{array}{l} B \to AC \\ A \to B \\ C \to B \end{array}\right\}$	$\left\{\begin{array}{l} A \to BC \\ B \to A \\ C \to A \end{array}\right\}$
1	2	3	4	5

■■■■

3 Queries

Detailed Explanation of Try Yourself Questions

T3 : Solution

Select Student
From FREQUENTS
Where parlor in (select parlor from SERVES where ice-cream in (select ice-cream from LIKES where LIKES.student = FREQEUNTS.student))

T4 : Solution

(b)

(a) finds those drinkers who are frequents atleast one bar on same city where he lives.

T5 : Solution

(d)

(a) $\pi_{A1}(R_1 - R_2) \not\equiv \pi_{A1}(R_1) - \pi_{A1}(R_2)$ because

R_1

A_1	A_2
2	4
3	4
2	5
3	5

R_2

A_1	A_2
2	4
2	5
3	5

LHS results:

A_1
3

RHS result:

A_1
Empty

(c) $\pi_{A1}(\sigma_{C1}(R_1)) \rightarrow \sigma_{C1}(\pi_{A1}(R_1))$ because LHS is always superset of RHS.

(d) $\pi_{A1}(\pi_{A2}(\sigma_{C1}(\sigma_{C2}(R_1)))) \rightarrow \pi_{A1}(\sigma_{C2}(\sigma_{C1}(R_1)))$ with condition $A_1 \subset A_2$ it gives the same results when LHS is replaced by RHS.

T6 : Solution

(d)

Course_id
CS-101
CS-347
PHY-101

–

Course_id
CS-101
CS-315
CS-311
FIN-201
HIS-351
MU-199

=

Course_id
CS-347
PHY-101

T7 : Solution

(a)

Query I returns vertices whose out degree is only 2. But Query II returns whose out degree is atleast 2.

T8 : Solution

(d)

1. $\rho(R_1, \pi_{sid}(\pi_{cid}(\sigma_{branch = 'CS'}(Course)) \bowtie Enrols))$

   $\rho(R_2, \pi_{sid}(\pi_{cid}(\sigma_{branch = 'IT'}(Course)) \bowtie Enrols))$

   $R_1 \cap R_2$

Find the Sid who enrolled atleast one course of CS branch then find the Sid who enrolled atleast one course of IT branch. Then take inter-section both Sid.

2. $\{T \mid \exists T_1 \in \text{enrols} (\exists x \in \text{courses} (x.\text{branch} = \text{'CS'} \wedge x.\text{cid} = T_1.\text{cid}) \wedge \exists T_2 \in \text{Enrols} (\exists y \in \text{courses} (y.\text{branch} = \text{'IT'} \wedge y.\text{cid} = T_2.\text{cid}) \wedge T_2.\text{sid} = T_1.\text{sid}) \wedge T.\text{sid} = T_1.\text{sid})\}$

Find the Sid who enrolled atleast one course of CS branch then find the Sid who enrolled atleast one course of IT branch with same Sid. Then return Sid.

3. Select Sid
   From courses P, Enrols C
   where P.branch='CS' AND P.cid = C.cid AND EXISTS (Select Sid
   From courses P2, Enrol C2
   where P2.branch = 'IT' AND C2.sid = C.sid
   AND P2.cid = C2.cid)

Find the Sid who enrolled atleast one course of CS branch then find the same Sid enrolled for atleast one course of IT branch and return it.

■■■■

4 File Structure and Indexing

Detailed Explanation of Try Yourself Questions

T2 : Solution

(a)

$$\text{File size} = 10{,}000$$

$$\text{Block size} = 1024$$

Number of entries in 1st level of Dense Index = Number of records in file.

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

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

$$\therefore \quad \text{Number of index } 1^{st} \text{ level blocks} = \left\lceil \frac{10000}{64} \right\rceil = 157 \text{ blocks}$$

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

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

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

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

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

(c) Number of levels in dense index

3^{rd} Level		2^{nd} Level		1^{st} Level		DB File
3 Entries $\Rightarrow$ 1 Block	$\rightarrow$	157 entries 157/64 = 3 Blocks	$\rightarrow$	10000 entries 1000/64 = 157 Blocks	$\rightarrow$	10000 record

$\therefore$ Three levels are required.

(d) Number of levels in sparse index

2^{nd} Level		1^{st} Level		DB File
15 Entries $\Rightarrow$ 1 Block	$\rightarrow$	190 entries $\Rightarrow$ 15 Blocks	$\rightarrow$	10000 entries (910 Blocks)

T3 : Solution

(a) File size = 30000 records

Number of entries in 1^{st} level dense index = Number of records in database

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

$$\text{Number of } 1^{st} \text{ level index blocks} = \frac{30000}{69} = 435 \text{ blocks}$$

$$\text{Number of } 2^{nd} \text{ level index blocks} = \frac{435}{64} = 70 \text{ blocks}$$

Number of 3^{rd} level index blocks = 1 (7 entries only to be entries in index block).

(b) Number of entries in 1^{st} level secondary index = Number of blocks in file

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

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

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

$$\text{Number of blocks in } 1^{st} \text{ level} = \frac{2728}{\text{Block factor of index block}} = \frac{2728}{69} = \lceil 39.5 \rceil = 40$$

Number of blocks in 2^{nd} level = 1 (only 40 entries per index block and block factor is 69)

(c) 3 levels.

(d) 2 levels.

T4 : Solution

(a) Database file size = 1250 records

Dense index is used

For minimum

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

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

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

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

$\therefore$ By using Dense index minimum 140 index blocks and 4 index levels are required.

(b) **For maximum**

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

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

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

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

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

$\therefore$ Using Dense index maximum 302 index blocks and 5 index levels are required.

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

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

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

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)

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

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

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

Number of index blocks in 4th index file = 1 block

$\therefore$ Using sparse index 102 blocks and 4 levels.

T5 : Solution

(a)

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

Updations do not cause unbalance in the tree.

T6 : 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)

T7 : 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.

T8 : Solution

(a)

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

■■■■

5 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 an y order of execution.

WR Sequence

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

$\therefore$ The following are the view equivalent schedules.

$$T_1 \to T_2 \to T_3$$
$$T_2 \to T_1 \to T_3$$

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

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.

$$T_1 \quad T_2 \quad T_3 \quad T_4$$
$$T_1 \quad T_3 \quad T_2 \quad T_4$$
$$T_2 \quad T_1 \quad T_3 \quad T_4$$
$$T_2 \quad T_3 \quad T_1 \quad T_4$$
$$T_3 \quad T_1 \quad T_2 \quad T_4$$
$$T_3 \quad T_2 \quad T_1 \quad T_4$$

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

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

∴ 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 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.

∴ $(T_1, T_2) \rightarrow (T_3, T_4)$

B : No initial reads

C : No initial reads

D : No initial reads

∴ 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_1 :

T_1	T_2	T_3
$r(x)$		
	$r(z)$	
$w(x)$		
		$r(x)$
		$r(y)$
$w(x)$		
C_1		
		$w(y)$
		C_3
	$r(y)$	
	$w(z)$	
	$w(y)$	
	C_2	

Strict recoverable
Cascadeless
Recoverable

S_2 :

T_1	T_2	T_3
$r(x)$		
	$r(z)$	
$r(z)$		
		$r(x)$
		$r(y)$
$w(x)$		
C		
		$w(y)$
	$r(y)$	
	$w(z)$	
	$w(y)$	
C_1		
	C_2	C_2

Not strict recoverable
Irrecoverable
Casscodeless

S_3 :

T_1	T_2	T_3
$r(x)$		
	$r(z)$	
		$r(x)$
$r(z)$		
	$r(y)$	
		$r(y)$
$w(x)$		
C_1		
	$w(z)$	
		$w(y)$
	$w(y)$	
	C	C_3
	C_2	

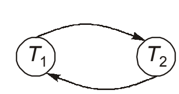
No strict
Recoverable
Cascadless

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

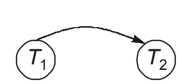
T_1	T_2
$r(x)$	
	$r(x)$
$w(x)$	
	$w(x)$



(2)

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

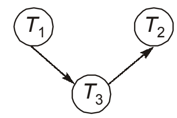
T_1	T_2
$w(x)$	
	$r(y)$
$r(y)$	
	$r(x)$



(3)

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

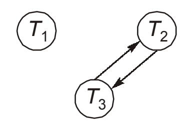
T_1	T_2	T_3
$r(x)$		
	$r(y)$	
		$w(x)$
	$r(x)$	
$r(y)$		



(4)

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

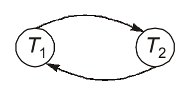
T_1	T_2	T_3
$r(x)$		
	$r(y)$	
$w(x)$		
	$r(y)$	
		$w(y)$
$w(x)$		
	$r(y)$	



(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

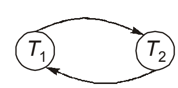
T_1	T_2
$r(x)$	
	$w(x)$
$w(x)$	
	abort
commit	



(6)

Not conflict serializable
Not view serializable
$T_2 \rightarrow T_1$
$T_2 \rightarrow T_1$
Not recoverable
Not cascadeless
Not stict
Not serializable

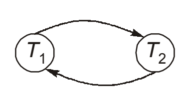
T_1	T_2
$r(x)$	
	$w(x)$
$w(x)$	
	$r(x)$
	commit
commit	



(7)

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

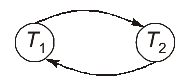
T_1	T_2
$w(x)$	
	$r(x)$
$w(x)$	
	$r(x)$
C_1	



(8)

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

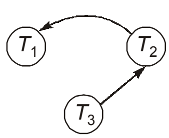
T_1	T_2
$w(x)$	
	$r(x)$
$w(x)$	
	C_2
C_1	



T4 : Solution

(a)

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

(i) 

Conflict serial schedules are T_3, T_2, T_1 only 1.

(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.

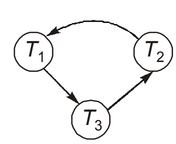
- If $(T_1, T_2, T_3) = (30, 20, 10)$
  No rollback i.e., time-stamp ordering is $T_3 \rightarrow T_2 \rightarrow T_1$.
- If $(T_1, T_2, T_3) = (20, 10, 30)$
  Transaction T2 is rollback on C.
- If $(T_1, T_2, T_3) = (30, 10, 20)$
  Transaction T_2 is rollback on C.
- If $(T_1, T_2, T_3) = (20, 30, 10)$
  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
$S(a)$		
$r(a)$	$S(b)$	
	$r(b)$	
		$S(c)$
		$r(c)$
$w(b)$		
	$w(c)$	
		$w(a)$

No basic 2PL possible.

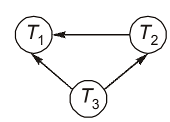
(iv) Strict 2PL

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)	
Not allowed		r(d)
X(c)		
w(c)		
	w(d)	
		w(e)

∴ Not allowed under basic 2PL.

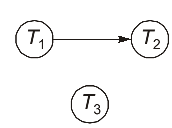
(iv) Strict 2PL:

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)	
		$r(d)$ unlock (d)
$X(c)$ $w(c)$ unlock (c)		
	$w(d)$ $X(d)$ 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
$r(a)$		
	$r(b)$	
		$r(c)$
$r(b)$		
	$r(c)$	
		$r(d)$
$w(a)$		
	$w(a)$	
		$w(c)$



∴ 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

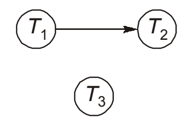
T_1	T_2	T_3
$r(a)$		
	$S(b), S(c)$	
	$r(b)$	
		$S(c), S(d)$
		$r(c)$
$r(b)$		
	$r(c)$	
		$r(d)$
$w(a)$		
unlock (b)		
unlock (a)		
	$X(a)$	
	$w(a)$	
	unlock (a)	
	unlock (b)	
	unlock (c)	
		$X(c)$
		$w(c)$
		unlock (c)
		unlock (d)

$\therefore$ 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
$r(a)$		
	$r(b)$	
		$r(c)$
$r(b)$		
	$r(c)$	
		$r(a)$
$w(a)$		
	$w(b)$	
		$w(c)$



∴ 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
$X(a)$,$s(b)$ $r(a)$		
	$S(b)$,$s(c)$ $r(b)$	
		$s(c)$ $r(c)$
$r(b)$		
	$r(c)$	
		$s(a)$ not possible $r(a)$
$w(a)$		
	$w(b)$	
		$w(c)$

∴ 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.

T12 : Solution

(c)
No uncommitted reads so that its casecadeless 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
	$r_2(x)$
$r_1(x)$	
	$r_2(y)$
$w_1(x)$	
$r_1(y)$	
	$w_2(x)$
a_1	
	a_2

■■■■