Conflict Serializability

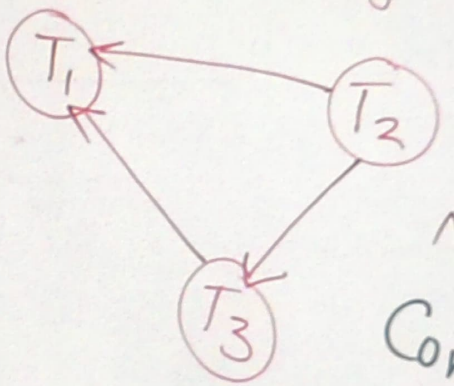
R-W
W-R
W-W

$\equiv$ S

T_1	T_2	T_3
R(x)		
		R(y)
		R(x)
	R(y)	
	R(z)	
		W(y) ← R(y), W(y)
	W(z)	
R(z)		
W(x)		
W(z)		

- Check Conflict pairs in other transactions and draw edges

Precedence graph



Loop / Cycle

No loop / Cycle

Conflict Serializable

Serializable

Consistent

Timestamp Ordering Protocol

→ Unique value assign to every transaction

→ Tells the order (when they enters into System)

→ Read_TS (RTS) = Last (latest) transaction no which performed Read successfully

Write_TS (WTS) = Last (Latest) transaction no. which performed Write successfully

TS(T_i)

Old 100 T_1 | 200 T_2 young

R(A) ← 10

W(A)

Co

Rules:

1) Transaction T_i issues a Read(A) operation
   a) if WTS(A) > TS(T_i), Rollback T_i
   b) Otherwise execute R(A) operation
      Set RTS(A) = Max{RTS(A), TS(T_i)}
2) Transaction T_i issues Write(A) operation
   a) if RTS(A) > TS(T_i) then Rollback T_i
   b) if WTS(A) > TS(T_i) then Rollback T_i
   c) otherwise execute Write(A) operation
      Set WTS(A) = TS(T_i)

$T_1 \rightarrow T_2$

100 old T_1 | 200 young T_2

W(A) 20

20 → R(A)

Co

*

100 T_1 | 200 T_2

W(A) ←

W(A)

Co

Convert a relation from 1st NF to 2nd NF

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

$F = \{A \xrightarrow{PD} B, B \rightarrow C\}$

$A^+ = ABC$

$R_1 = \{A, B, C\}$

2NF

$A^+ = ABC$

$F_1 :- \{A \rightarrow BC, B \rightarrow C\}$

$A^+ = ABC$

$B^+ = BC$

$C^+ = C$

AD

$D^+ = D$

$Ck = A$

$R_2 = \{A, D\} \Rightarrow$ BCNF

$R_2\{B, D\}$

$R_2\{C, D\}$

$F_2 = \{\}$

$A^+ = A$

$D^+ = D$

$F_1 \cup F_2 \equiv F$

$ABCD^+ = ABCD$

$AD^+ = ADBC$

$A^+ = ABC$

$D^+ = D$

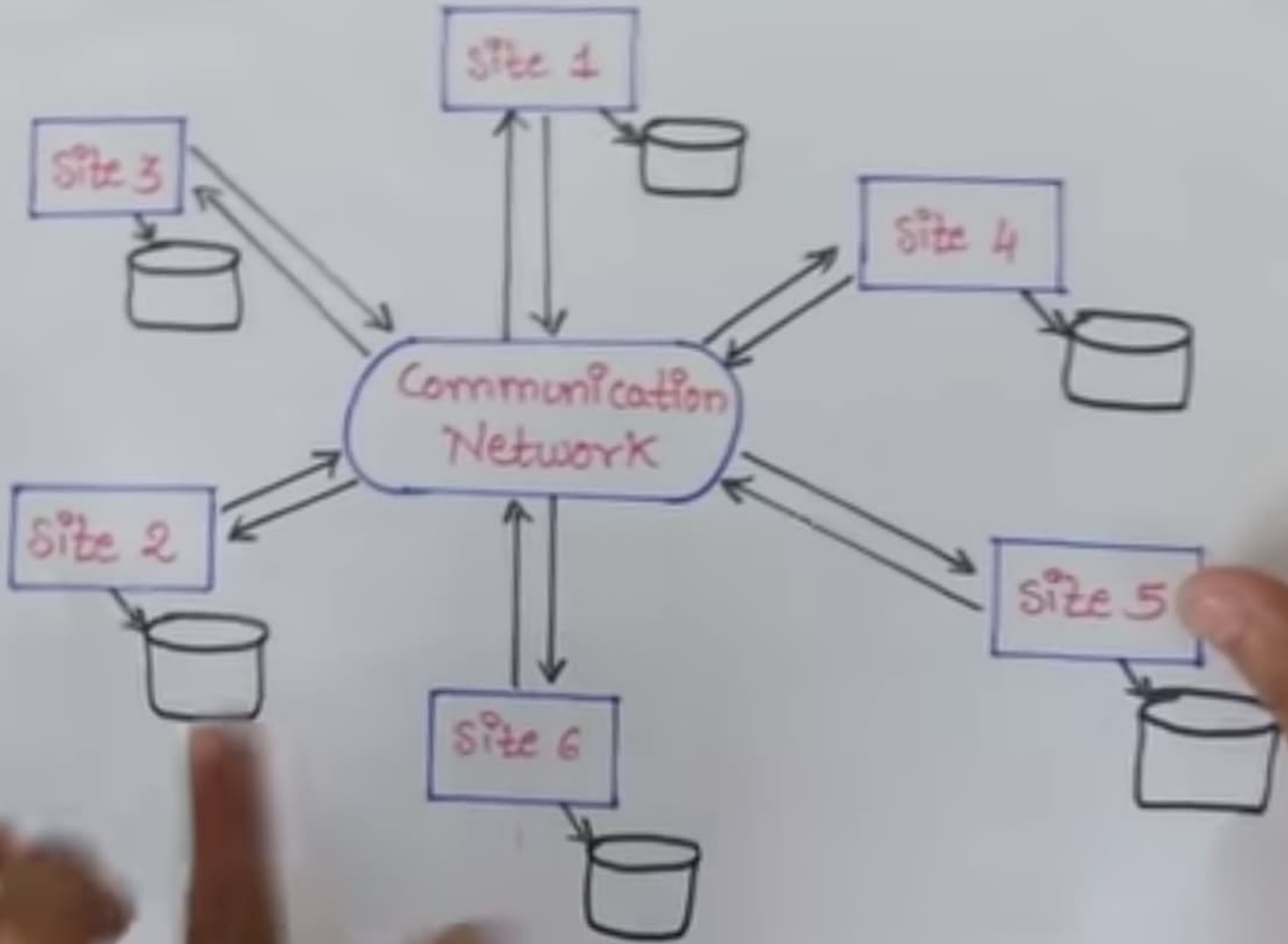
$Ck = AD$

$PA = A, D$

Features :

- It is a collection of logically - related shared data.
- Data in DDBMS is split into no. of fragments.
- Fragments can be replicated in a distributed system.
- Fragments / replicas allocated to different sites.

DISTRIBUTED DATABASE



- In Distributed System, sites are linked by the comrn vnication network.
- Data at each site is under control of DBMs.
- DBMs at each site has its own right, i.e it can handle local application independently.
- Each DBMS in a distributed System must have at least one gbbal application.

Transperency:

Hiding details from the user.

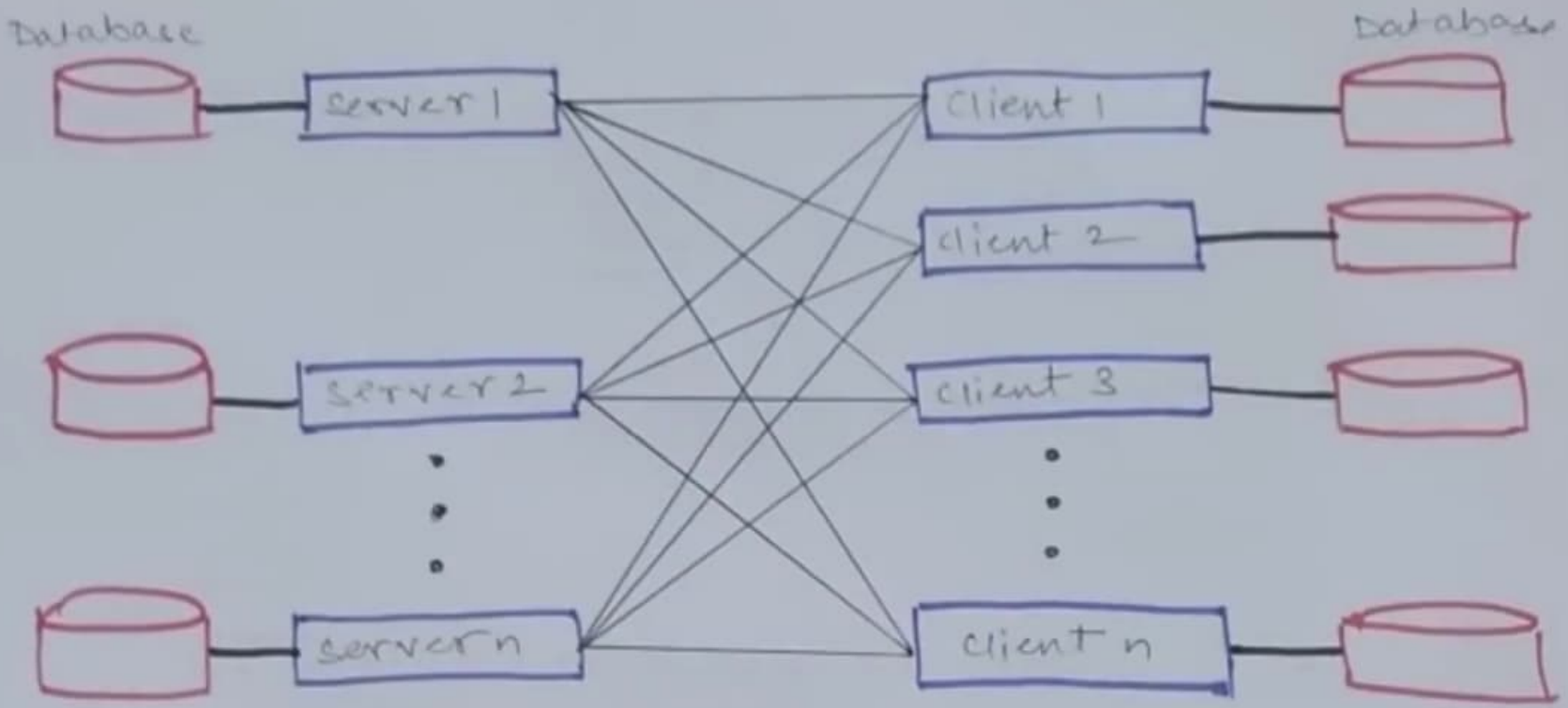
Types of Transperency.

1. Network Transperency

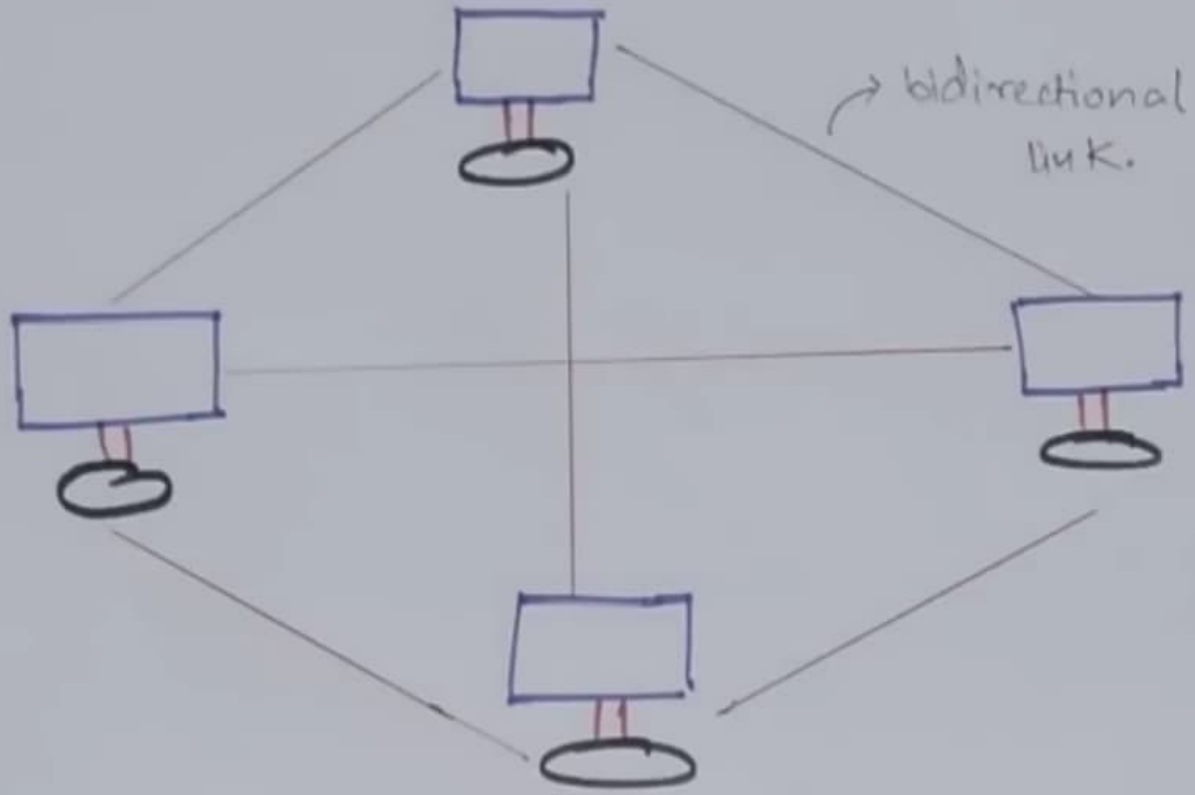
Location Transperency

Naming Transperency.

2. Client-server Architecture:



3. Peer-to-Peer Architecture :



Disadvantages: i) Complexity of management and control.
ii) Deadlock handling
iii) Security
iv) Lack of Standards.

Types of Distributed Databases:

Homogeneous	Heterogeneous
i) Share a common Global Schema.	ii) diff. sites can have diff. schema
ii) Run identical DBMS S/W.	ii) Run diff. DBMS S/W.
iii) Each site provides part of its autonomy in terms of right to change schema or S/W.	iii) Each site maintains its own right to change the schema or S/W.
iv) Same S/W - No Problem in transaction processing.	iv) Diff. S/W - Major Problem in transaction Processing.
v) Same schema - No Problem in Query Processing.	v) Diff. Schema - Problem in Query Processing

Distributed Data Storage: Various Approaches are the following:

i) **Fragmentation:** In this, the system partitions/ divides the relation into several fragments, and stores each fragment at a different site.

Horizontal Fragmentation

- It Breaks relation 'R' by assigning each tuple of R to one or more fragments.
- Each fragment is a subset of tuples in original rel^n 'R'.

Vertical Fragmentation

- It Breaks relation 'R' by decomposing schema.
- each fragment is a subset of the attributes of the original rel^n 'R'.

DBMS	RDBMS
1⇒ In DBMS data are Store as a file.	1⇒ In RDBMS data are Store in a tabular form.
2⇒ It does not support Client Server architecture	2⇒ It support Client Server architecture.
3⇒ Normalization is not available in DBMS.	3⇒ Normalization is available in RDBMS.
4⇒ It allows one user at a time.	4⇒ It Allows more than one user at a time.
5⇒ Hierarchical arrangement of data.	5⇒ Stores data in the form of rows and columns.

DBMS.

4⇒ It allows one user at a time.

5⇒ Hierarchical arrangement of data.

6⇒ Low software and hardware needs.

7⇒ It does not support ACID property.

8⇒ Data redundancy

available in RDBMS.

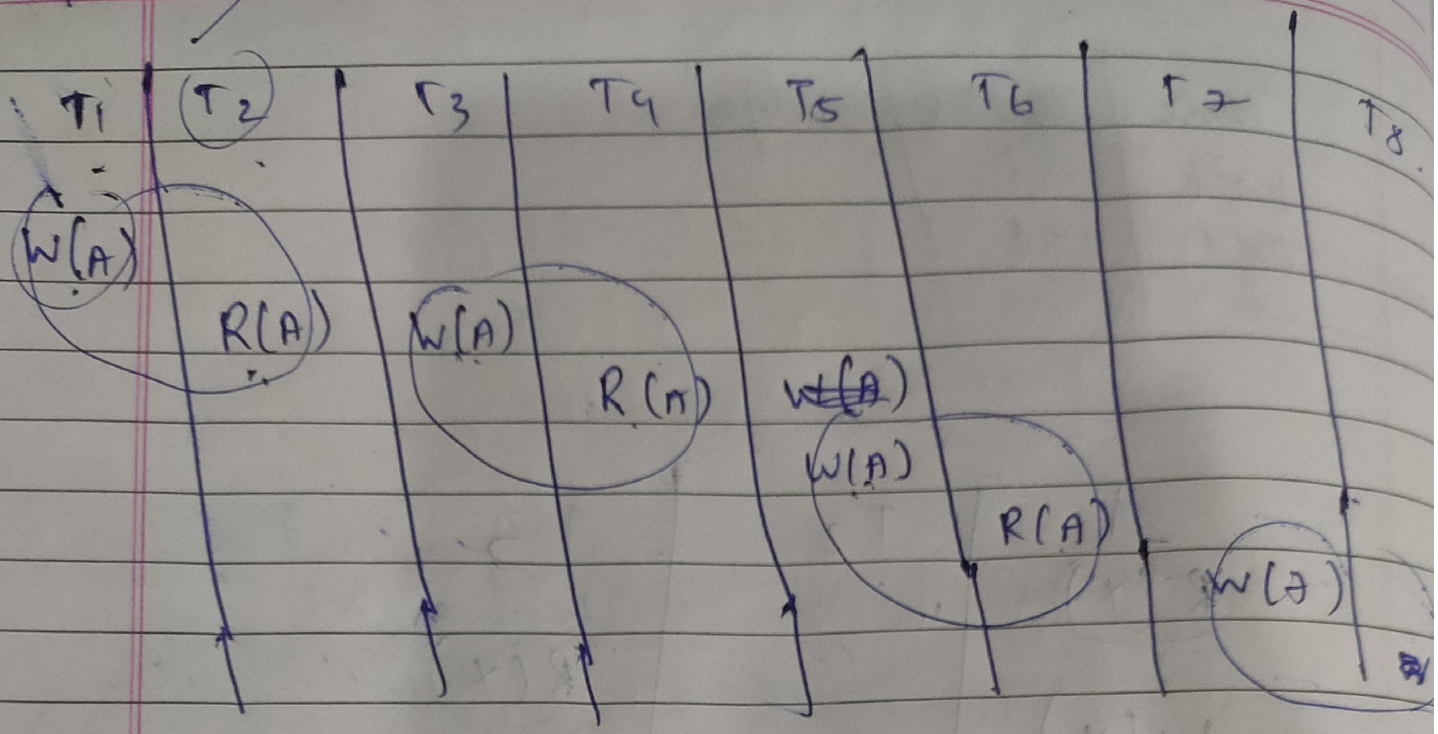
4⇒ It Allows more than one user at a time.

5⇒ Stores data in the form of rows and columns.

6⇒ Higher hardware and software needs.

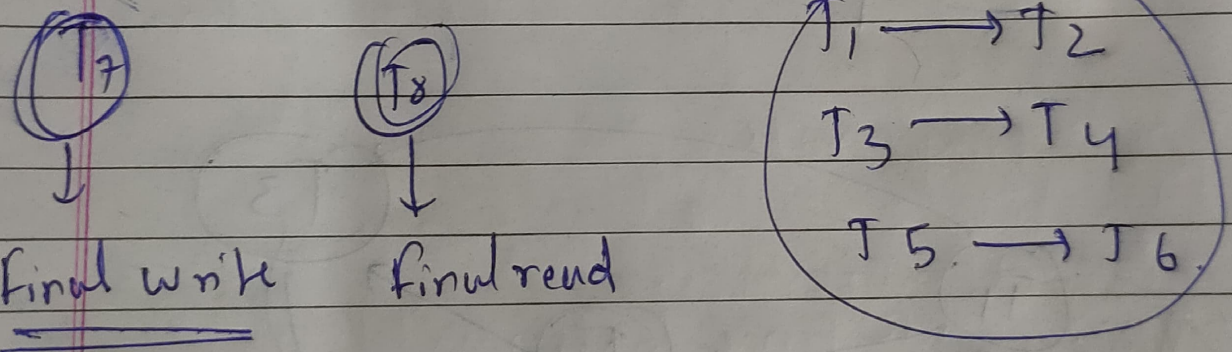
7⇒ It supports ACID property.

8⇒ data redundancy



T_1 T_3 T_5

W-R depending matrix



final write final read

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

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

Best answer

This 3 rules we need to follow.

here is blind write perform by T1, T3 and T5.

and T7 T8 are final operation , which is fixed.

T1->T2 and T3->T4 and T5->T6 can be any order but should maintain WR dependency means after T1 T2 only come.

so, **T1->T2** and T3->T4 and T5->T6 1st order

T1->T2 , T5->T6 , T3->T4 2nd order

T5->T6 T3->T4 **T1->T2** 3rd order

T3->T4 T5->T6 **T1->T2** 4th order

T5->T6 **T1->T2** T3->T4 5th order

T3->T4 **T1->T2** T5->T6 6th order

in this 6 way we can make all view equivalent serial schedules.

10. (I) Consider a concurrency control manager by timestamps. Below are several sequences of events, including start events, where **sti** means that transaction Ti starts and **coi** means Ti commits. These sequences represent real time, and the timestamp-based scheduler will allocate timestamps to transactions in the order of their starts.

In each of the case below, say what happens with the last request. **[2*5=10]**

(a) st1; st2; r1(A); r2(A); w1(B); w2(B);
(b) st1; st2; r2(A); co2; r1(A); w1(A)
(c) st1; st2; st3; r1(A); w3(A); co3; r2(B); w2(A)
(d) st1; st2; st3; r1(A); w1(A); r2(A);
(e) st1; st2; st3; r1(A); w2(A); w3(A); r2(A);

You have to choose between one of the following four possible answers:
(i) the request is accepted,
(ii) the request is ignored,
(iii) the transaction is delayed,
(iv) the transaction is rolled back.

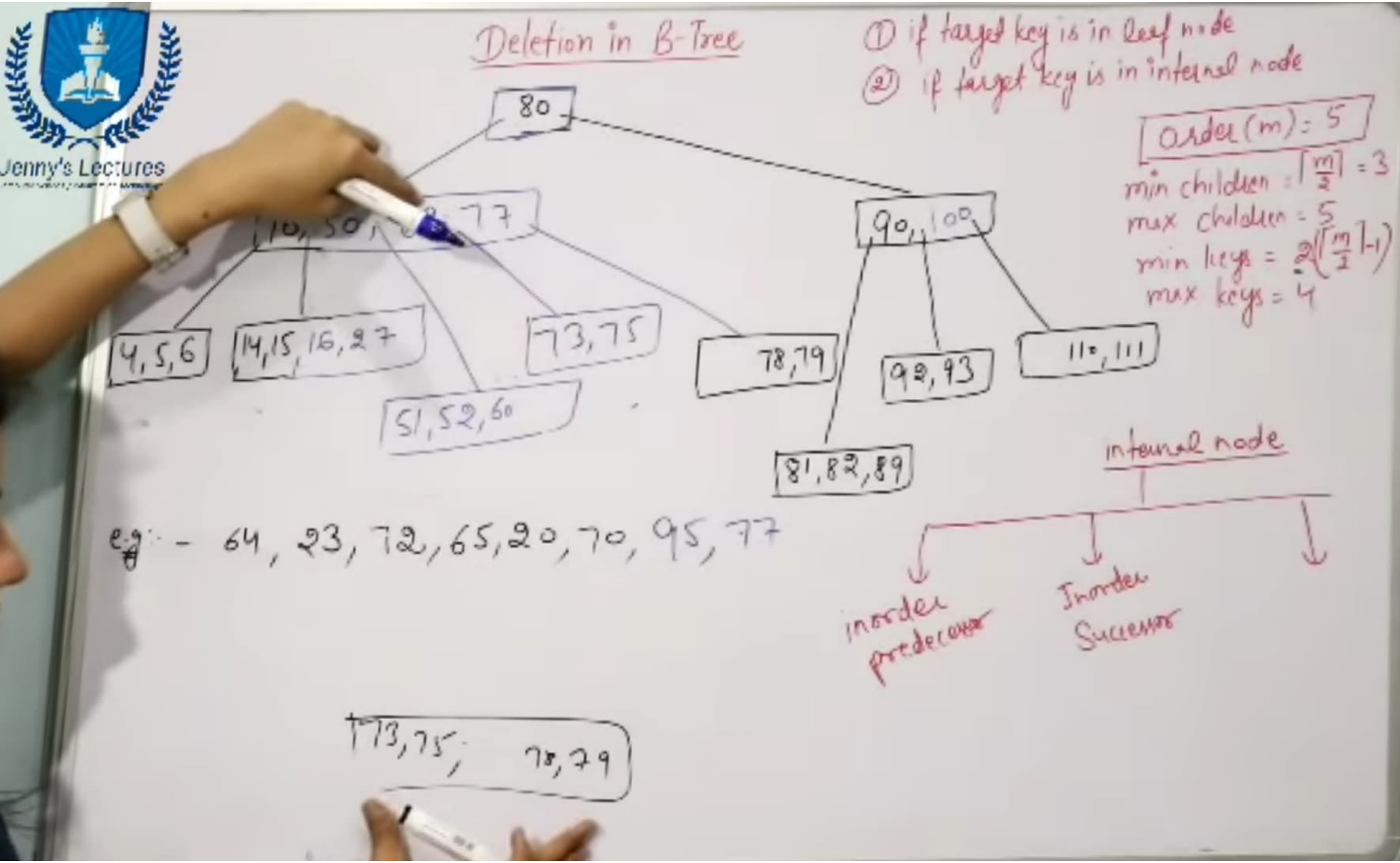
(II) Write the problems faced when concurrent transactions are executed in an uncontrolle

(ii) the request is ignored,
(iii) the transaction is delayed,
(iv) the transaction is rolled back.

(II) Write the problems faced when concurrent transactions are executed in an uncontrolled manner? Give an example & explain. **[6]**

(III) Determine whether the schedule is strict, cascadeless, recoverable, or nonrecoverable

S: $r_1(X)$, $r_2(Z)$, $r_1(Z)$, $r_3(X)$, $r_3(Y)$, $w_1(X)$, c_1; $w_3(Y)$, c_3; $r_2(Y)$, $w_2(Z)$, $w_2(Y)$, c_2; **[4]**


Deletion in B-Tree
① if target key is in leaf node
② if target key is in internal node
80
10, 50, 77
90, 100
4, 5, 6
14, 15, 16, 27
51, 52, 60
73, 75
78, 79
92, 93
110, 111
81, 82, 89
Order (m) = 5
min children = ⌈m/2⌉ = 3
max children = 5
min keys = 2(⌈m/2⌉ - 1)
max keys = 4
e.g. - 64, 23, 72, 65, 20, 70, 95, 77
internal node
inorder predecessor
Inorder Successor
73, 75, 78, 79