SQL SUBQUERIES

Q1)
Write a SQL query to display maximum salary from Emp table

E_ID	E_NAME	DEPT	SALARY
1	RAM	HR	10000
2	AMRIT	MRKT	20000
3	RAVI	HR	30000
4	NITIN	MRKT	40000
5	Varun	IT	50000

ANS :-

```
SELECT MAX(SALARY) FROM
EMP;
```

Q2)
Write a SQL query to display Employee Name who is taking maximum salary?

```
[SELECT E_NAME FROM EMP WHERE SALARY=
(SELECT MAX(SALARY) FROM EMP); ]
```

Q3)
Write a SQL query to display second hignest salary from Emp table

```
SELECT MAX(SALARY) FROM EMP <> [SELECT
MAX(SALARY) FROM EMP
```

Q4) Write a SQL query to display employee name who is having second Highest salary

```
SELECT FROM E_NAME FROM EMP WHERE
SALARY =((SELECT MAX(SALARY) FROM EMP)<>
[SELECT MAX(SALARY) FROM EMP))
```

Q5) write a query to display all the dept names along with all emps working in it

output :-

HR	2
MRKT	2
IT	1

HR - 2
MRKT - 2
IT - 1

SELECT DEPT FROM EMP GROUP BY DEPT;

```
SELECT DEPT, COUNT(DEPT) FROM EMP GROUP BY
DEPT
```

SELECT E_NAME FROM EMP WHERE DEPT IN[[
SELECT DEPT FROM ~~DEP~~EMP GROUP BY DEPT]
HAYING COUNT(DEPT) < 2 ;]]

Write a query to display highest salary department wise and name of emp who is taking that salary.

SELECT E_NAME FROM MAX(SALARY)
EMP WHERE [SELECT DEPT, FROM EMP GROUP BY
SALAY
IN DEPT]

R {A B C D E}

F.D = { A → B, BC → D, E → C, D → A }

E = BDCA

E+ = { EC }

∴ CK = { AE, BE, DE }

(AE)+ = { A B C D E }

(BE)+ = { B E C D A }

(CE)+ = { E C

FUNCTIONAL DEPENDENCIES

1st Normal Form	2nd Normal form	3rd Normal form	BCNF	4th Normal format	5NF
No multi-valued attributes	should be in 1 NF No partial Dependencies A N.P.A cannot determine a N.P.A Should be Full Depen on CK	Should be in 2NF No transitivity No N.P.A should determine N.P.A	In 3rd N.F LHS must be CK or S.K	BCNF + No M.V. Depen	4CNF + Lossless Dec

Equi Join

Join = cross product + condition

find the emp name who worked in a department having location same as address.

E_NO	E_Name	Address
1	Ram	Delhi
2	varun	chd
3	Ravi	chd
4	Amrit	Delhi

Emp

Dept No	Location	E-No (fk)
D1	Delhi	1
D2	pune	2
D3	patna	4

Dept

Select E_name from Emp, Dept where Emp.Eno = Dept.e_no. and emp.Address = dept.location;

LEFT OUTER JOIN

Emp

Emp-NO	E-Name	Dept_NO
E1	varun	D1
E2	Amrit	D2
E3	Ravi	D1
E4	Nihin	—

Dept

Dept-No	D-Name	Loc
D1	IT	Delhi
D2	HR	Hyd
D3	Finance	pune

Select emp_no, E_name, ~~Dept.~~ d_name, Loc
from emp left outer join dept
on (emp.dept_no = dept.dept_no)

JOINS

Natural Join

find the Emp Names who is working in a department

E-NO	E-Name	Address
1.	Ram	Delhi
2.	varun	chd
3.	Ravi	chd
4.	Amrit	Delhi

m (EMP)

Dept No	Name	Eno
D_1	HR	1
P_2	IT	2
P_3	MRKT	4

n (DEPT)

m X n

Query: [SELECT FROM EMP, DEPT WHERE EMP. ENO = DEPT. E-No.]

self Join

find the student id who is enrolled in at least two course

2, 3, ∞

Study

S_id	c_id	since
S1	C1	2016
S2	C2	2017
S1	C2	2017

SELECT FROM STUDY as T_1, STUDY as T_2
where $T_1.S\_id = T_2.sid$
and $T_1.c\text{-}id <> T_2.C\_id$

Join = cross prod + cnd

RIGHT OUTER JOIN

Emp

Emp-No	E-Name	Dept-No
E1	Varun	D1
E2	Amrit	D2
E3	Ravi	P3

Dept

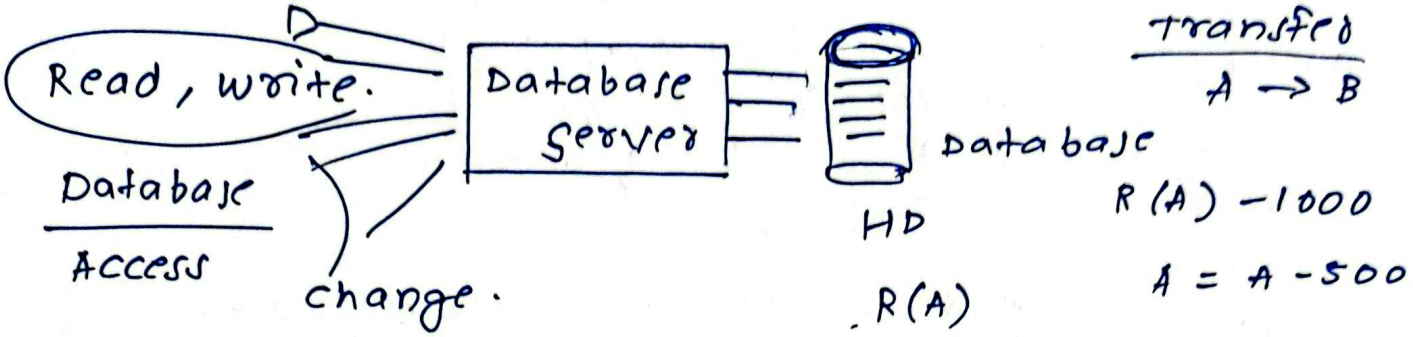
Dept-No	D-Name	Loc
D1	IT	Delhi
D2	HR	Hyd
D3	Finance	pune
D4	Testing	Noida

select emp-no, e-name, ~~dept~~ d-name, Loc
from Emp ~~Right~~ outer join dept on
(emp.dept-no = dept-dept-no)

Transaction : It is a logical set of operation used to perform a logical unit of work

A transaction generally represent change in database.

work - withdraw money



Transfer

A = 1000
B = 2000

R(A) - 1000
A = A - 500
W(A) - 500
R(B) - 2000
B = B + 500
W(B) = 2500
commit

TRANSACTION EXAMPLE

ACID properties

↓

Atomicity Consistency Isolation Durability

Either all or NONE

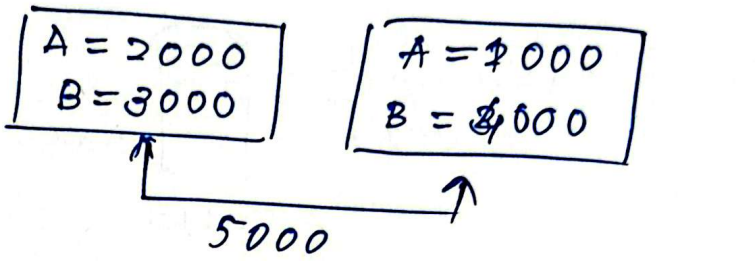
Roll Back

T1
R(A)
A = A - 50
W(A)
R(B)
R(C)
commit

Consistency: After the transaction completed sum of Money should be same.

~~A → 100~~ A → B

T1	
R(A)	2000
A = A - 1000	
W(A)	1000
B = B + 1000	
W(B)	4000
commit	



Sum of Money Remains same.

Isolation

T1	T2	T3
R(A)		
	R(B)	
		R(A)

parallel schedule can be converted to serial schedule

Durability After a transaction completes successfully, the changes it has made to the database persists, even if the system fails.

cascadless-schedule

A = 100

T_1	T_2	T_3
R(A)		
W(A)		
	R(A)	R(A)

Jab tak commit hojaneke baad R(A) ✓ R(A) ✓

T_1	T_2
100 R(A)	
	R(A) 100
W(A)	
:	
fail	W(A)

Serializability → conflict
→ view

S

T_1	T_2
R(A)	
W(A)	
commit	
	R(A)
	W(A)

$T_1 \rightarrow T_2$

S

T_1	T_2
	R(A)
	W(A)
	commit
R(A)	
W(A)	
commit.	

$T_1 \leftarrow T_2$

S

T_1	T_2	T_3
	R(A)	
		R(A)
		W(A)
	W(A)	
R(B)		
W(B)		
	W(B)	

$3! = 3 \times 2 \times 1 = 6$

- $T_1 \rightarrow T_2 \rightarrow T_3$
- $T_1 \rightarrow T_3 \rightarrow T_2$
- $T_2 \rightarrow T_3 \rightarrow T_1$
- $T_2 \rightarrow T_1 \rightarrow T_3$
- $T_3 \rightarrow T_1 \rightarrow T_2$
- $T_3 \rightarrow T_2 \rightarrow T_1$

Any one possibility should be true.

Conflict Equivalent.

$S \equiv S'$

S

T_1	T_2
R(A)	
W(A)	
	R(A)
	W(A)
R(B)	

S'

T_1	T_2
R(A)	
W(A)	
R(B)	
	R(A)
	W(A)

R(A) R(A) } Non-conflict pairs

~~R(A)~~ W

R(A) W(A) }
W(A) R(A) } conflict pairs
W(A) W(A) }

R(B) R(A) }
W(B) R(A) }
R(B) W(A) } Non-conflict
W(A) W(B) }

T_1	T_2
R(A)	
W(A)	
	R(A)
R(B)	
	W(A)

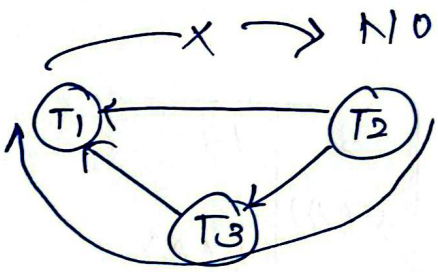
$\therefore S \equiv S'$

check conflict pairs in o/p transaction and draw edges

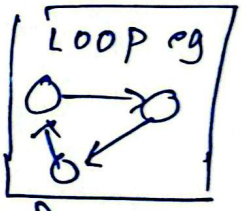
T_1	T_2	T_3
~~R(X)~~		~~R(Y)~~
		~~R(Y)~~
		~~R(X)~~
	~~R(Y)~~	
	~~R(Z)~~	
		~~W(Y)~~
	~~W(Z)~~	
~~R(Z)~~		
~~W(X)~~		
~~W(Z)~~		

checked

precedence Graph



NO-LOOP



Loop / cycle ?

No loop / cycle

Conflict serializable

↓

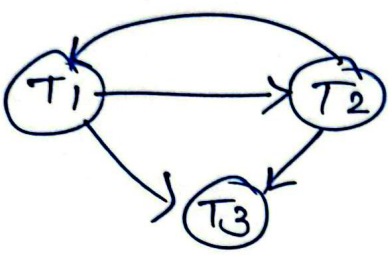
serializable ✓

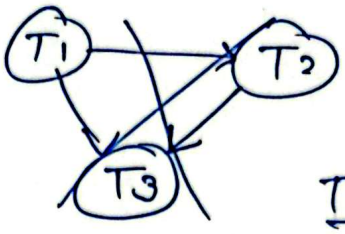
↓

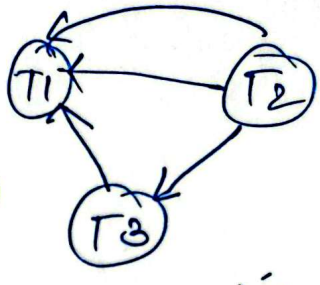
consistent ✓

check whether schedule is conflict serializable or not.

T1	T2	T3
~~R(A)~~		
	~~W(A)~~	
W(A)		
		W(A)







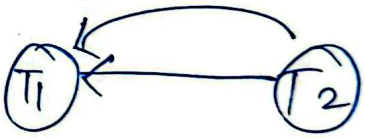
There exist a loop

Not a conflict serializable schedule because of loop

if the final outputs are same but are different then they are view equivalent

T1	T2	T3
R(A)		
	A=A-40	
	W(A)	
A=A-40		
		A=A-20
		W(A)

T1	T2	T3
R(A)		
W(A)		
	W(A)	
		W(A)



previous year question

T1	T2
A=100	
B=10	
	~~R(A)~~ 100
	~~R(A)~~ A=A+10
	~~R(B)~~ B=B+1
	W(B)
B=11	
	R(B)
	W(A)

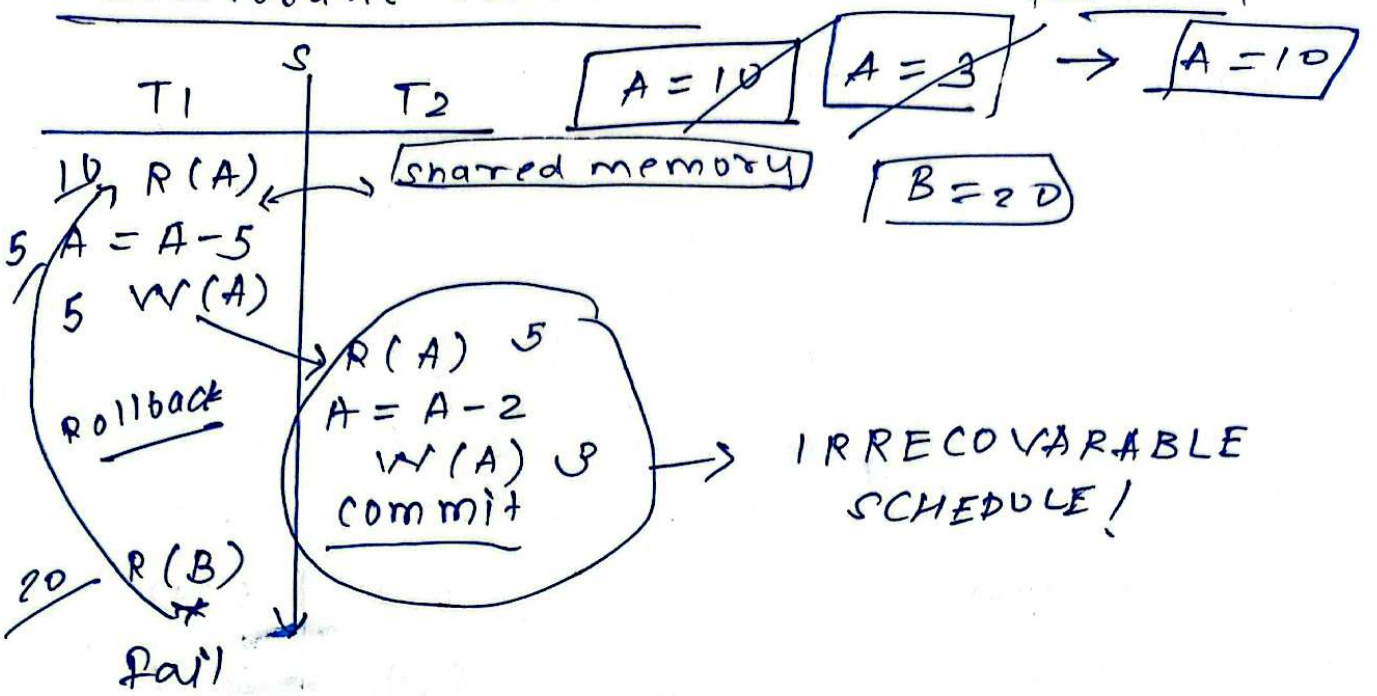
110

No loop or cycle

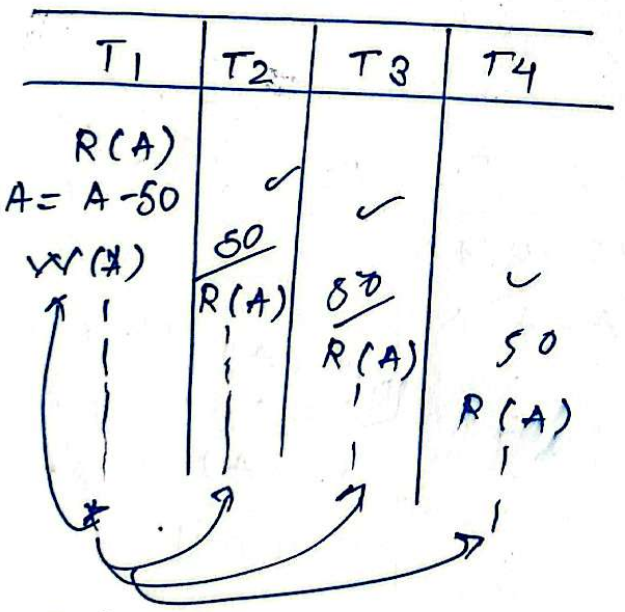
∴ ~~Not~~ its a conflict ~~serializable~~

Yes

IRRecovarable schedule

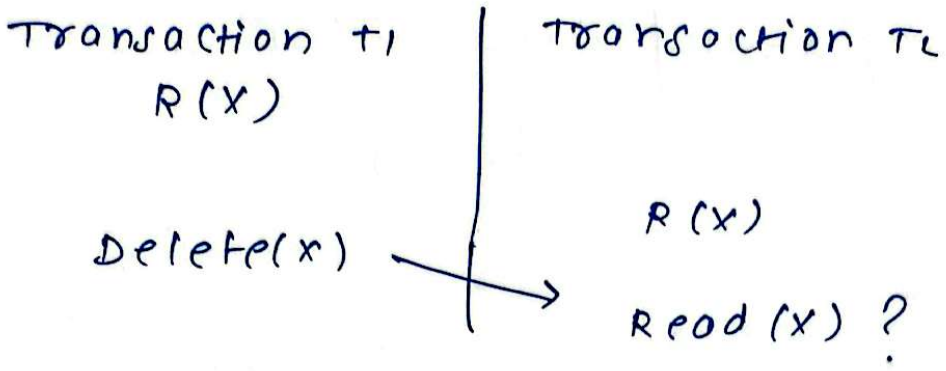


Cascading schedule

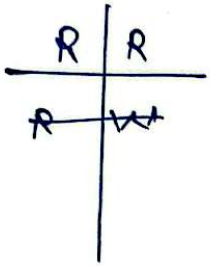


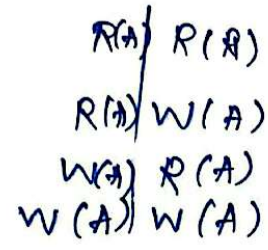
Disadvantage: CPU Utilization is and performance is affected

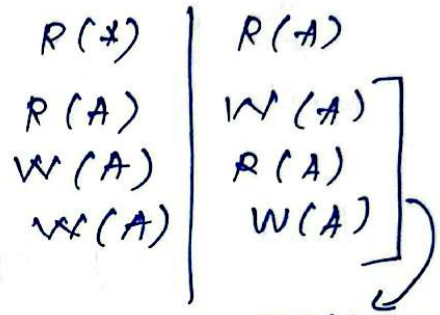
Unrepeatable Read and phantom read

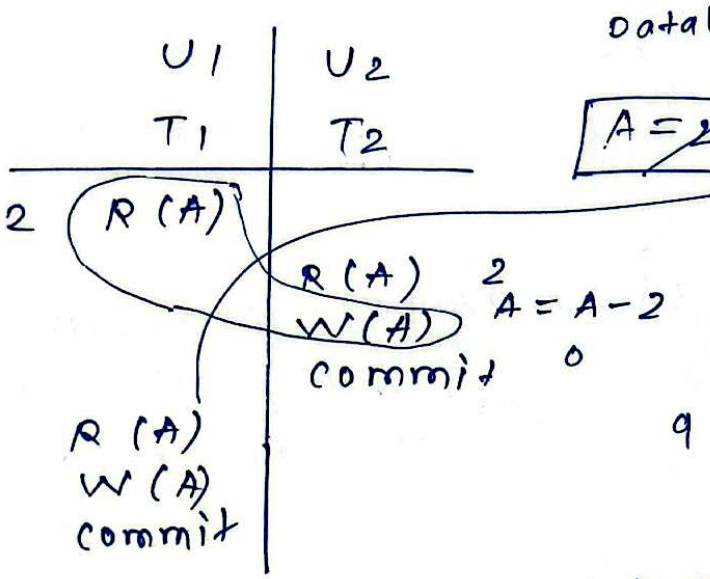


Read - write conflict or unrepeatable read

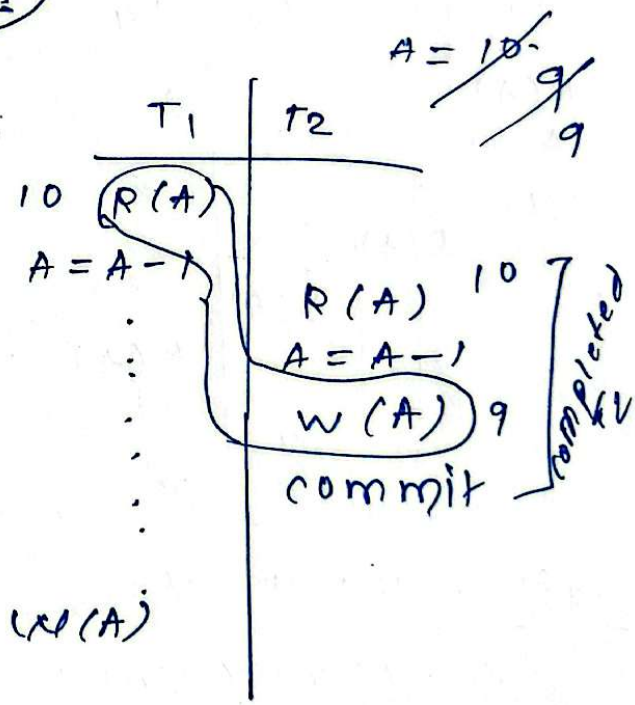


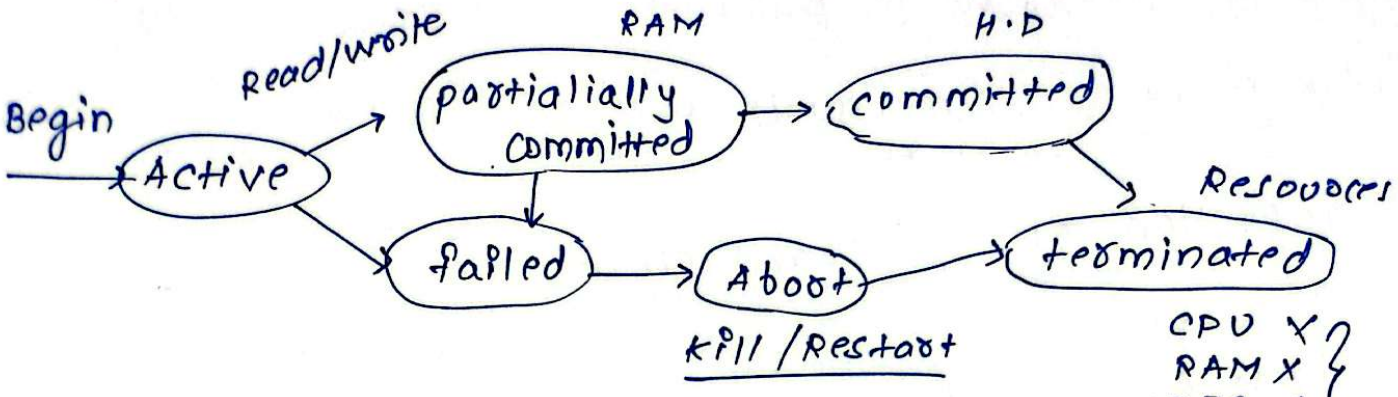






Read - write conflict





R(A)
A = A - 50
W(A)
R(..)
.
.
.

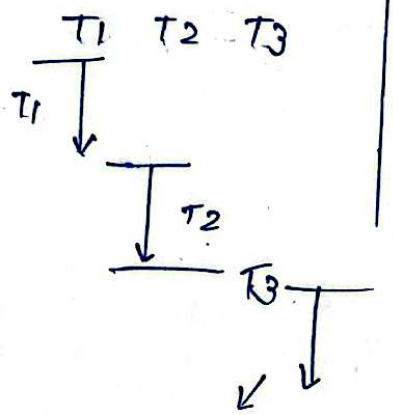
commit (n-1)

SCHEDULE

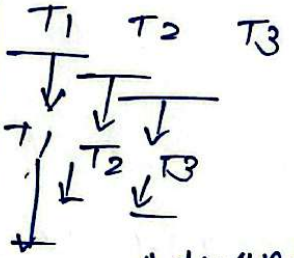
Schedule:- It is chronological execution sequence of multiple transactions.

T_1 T_2 T_3 ..... T_n

Serial



parallel



Advantage:- Less waiting period
faster performance of CPU

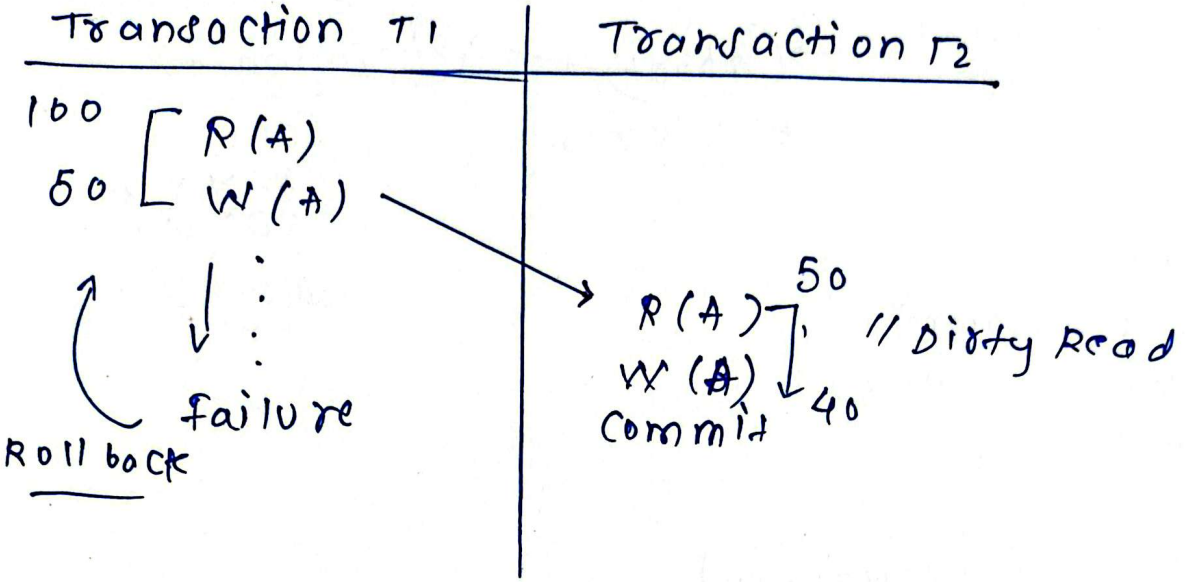
No other Transaction can interfer only one after another.

Advantage:- more consistent

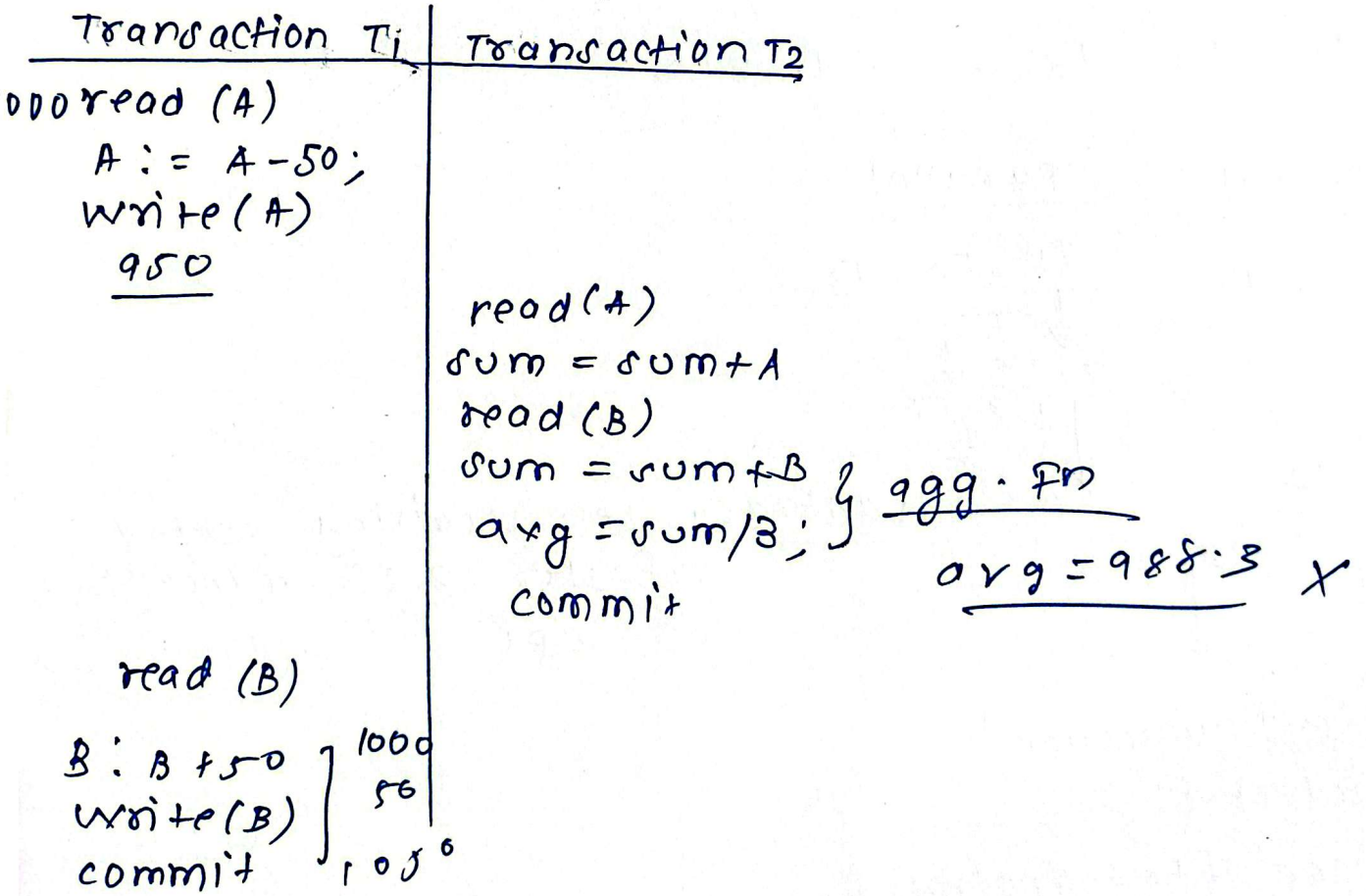
Disadvantage:- waiting period / time period & perform...

Types of problems in parallel schedule

Dirty Read or Uncommited Read or RAW



Incorrect summary



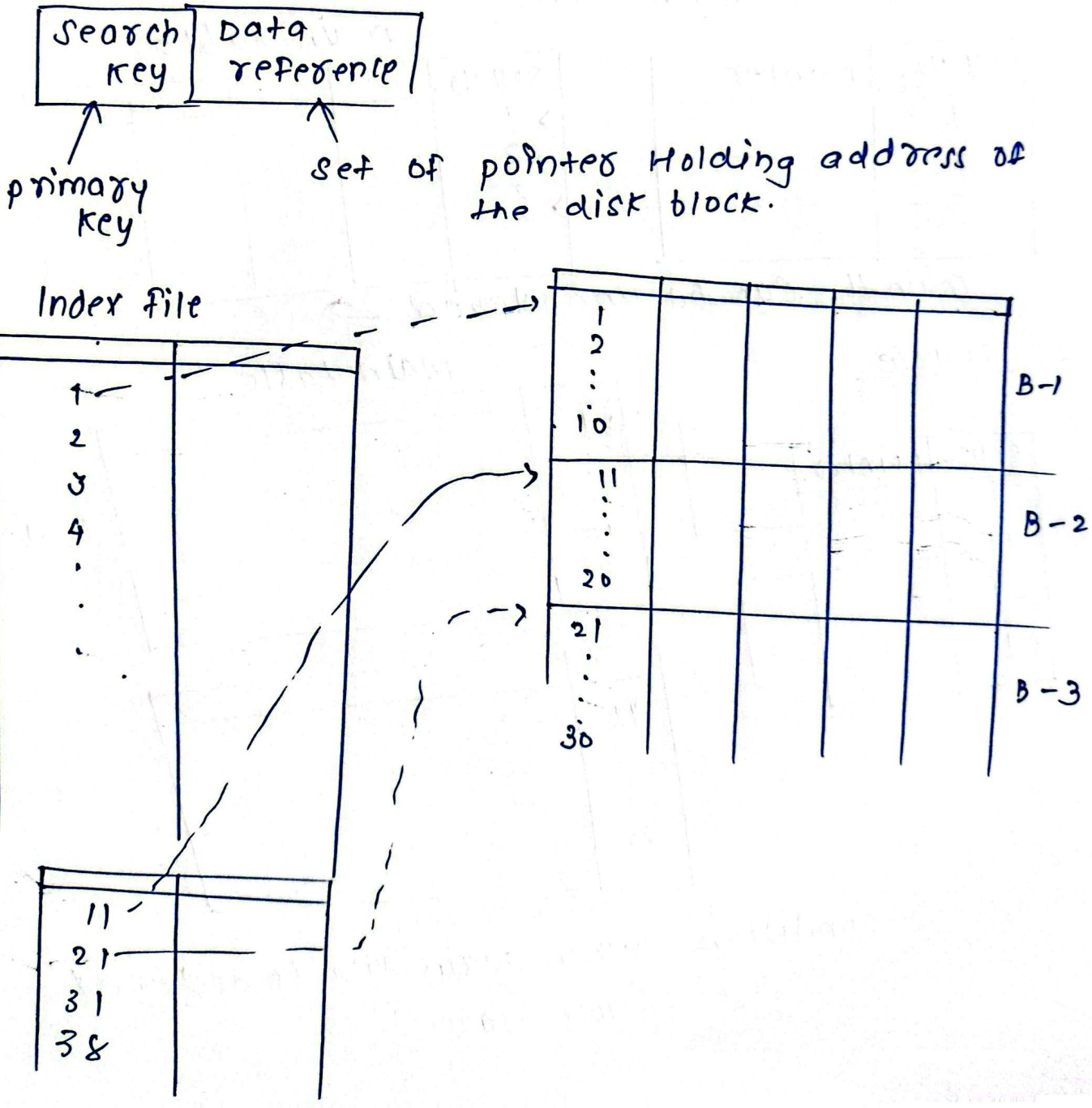
Indexing In DBMS

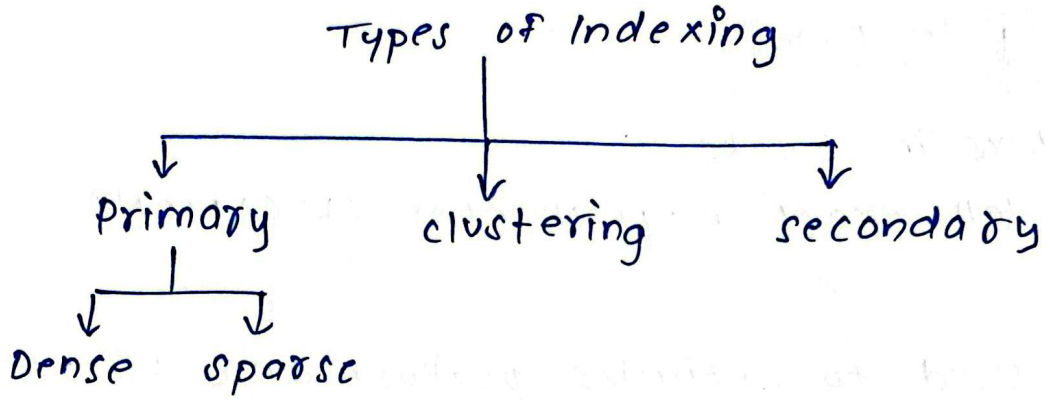
→ Just like indexs in Books

→ Index will tell exact page Number. (LOCATION) of that page.

→ Indexing is used to optimize performance of DB By minimizing the no. of disk access required when query is processed.

File size↑ search time↑

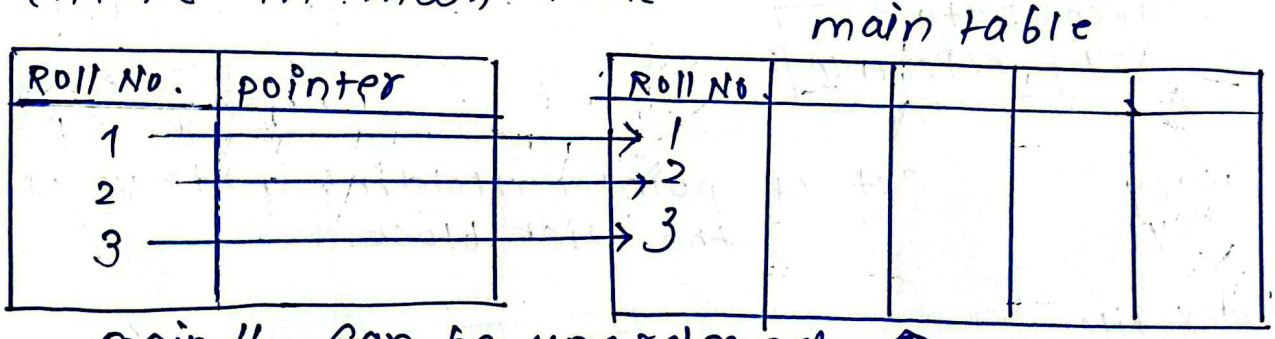




① primary

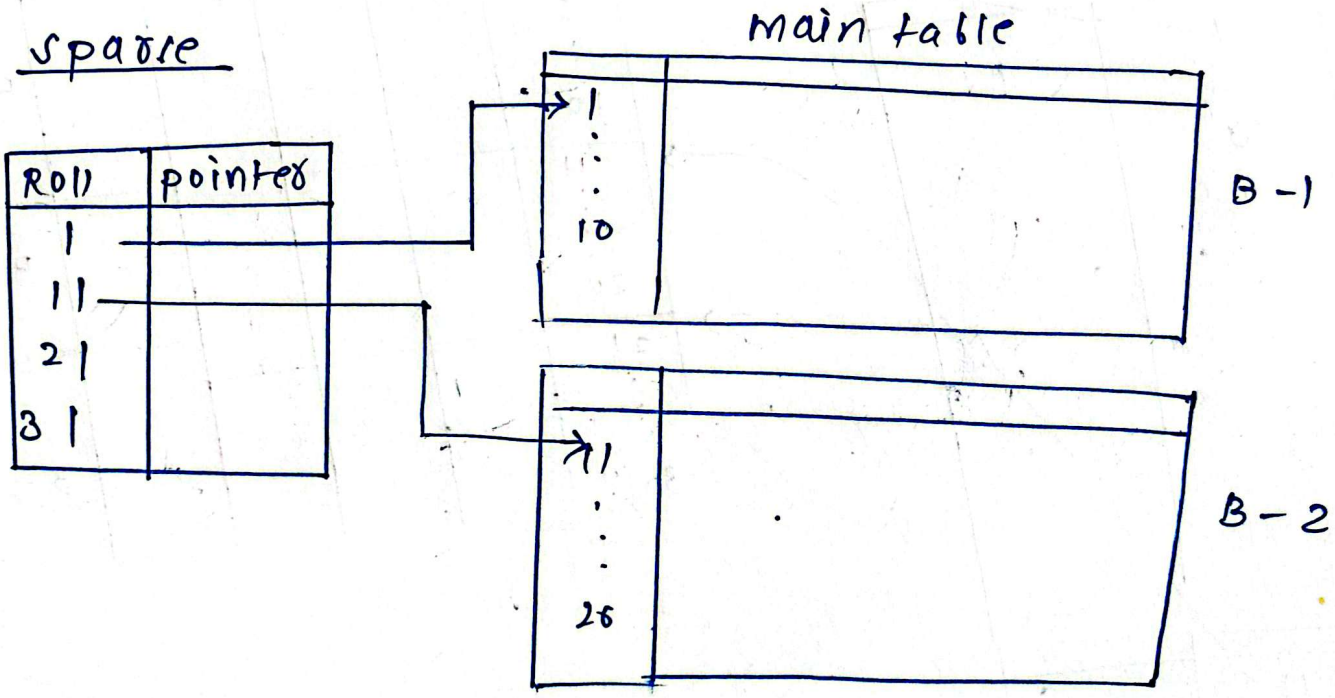
Dense

→ No. of entries in Index table is same as the no. of entries in main table.



main tb can be unordered ⇒

2) sparse



condition : when main file is ordered (for sparse)

B+ Trees in database

↳ used to implement database indexes

⇒ In B+ trees leaf nodes denotes actual data-pointers.

⇒ All leaf nodes remain at same height

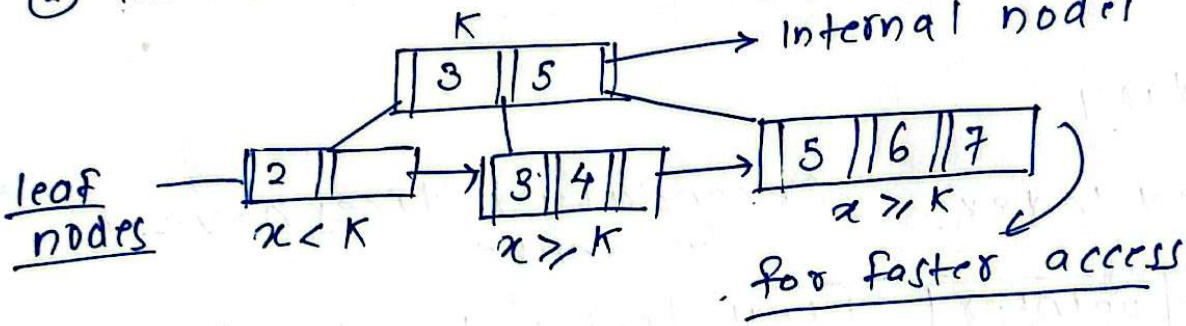
⇒ leaf nodes are linked using linked-List.

⇒ B+ trees occupy a little more space than B-trees.

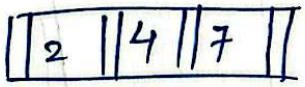
characteristics

① Data-records are only stored in Leaves

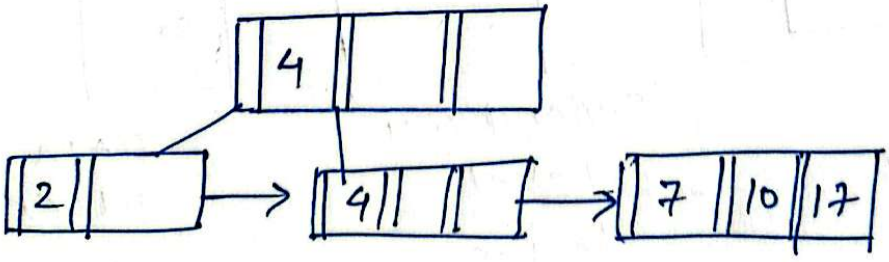
② internal nodes stored just keys

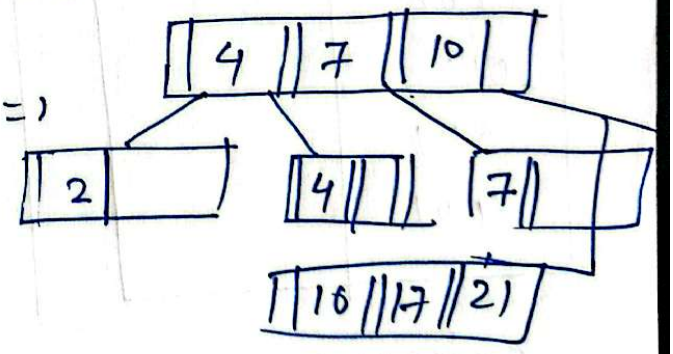


Example :- 2, 4, 7, 10, 17, 21, 28



insert 10

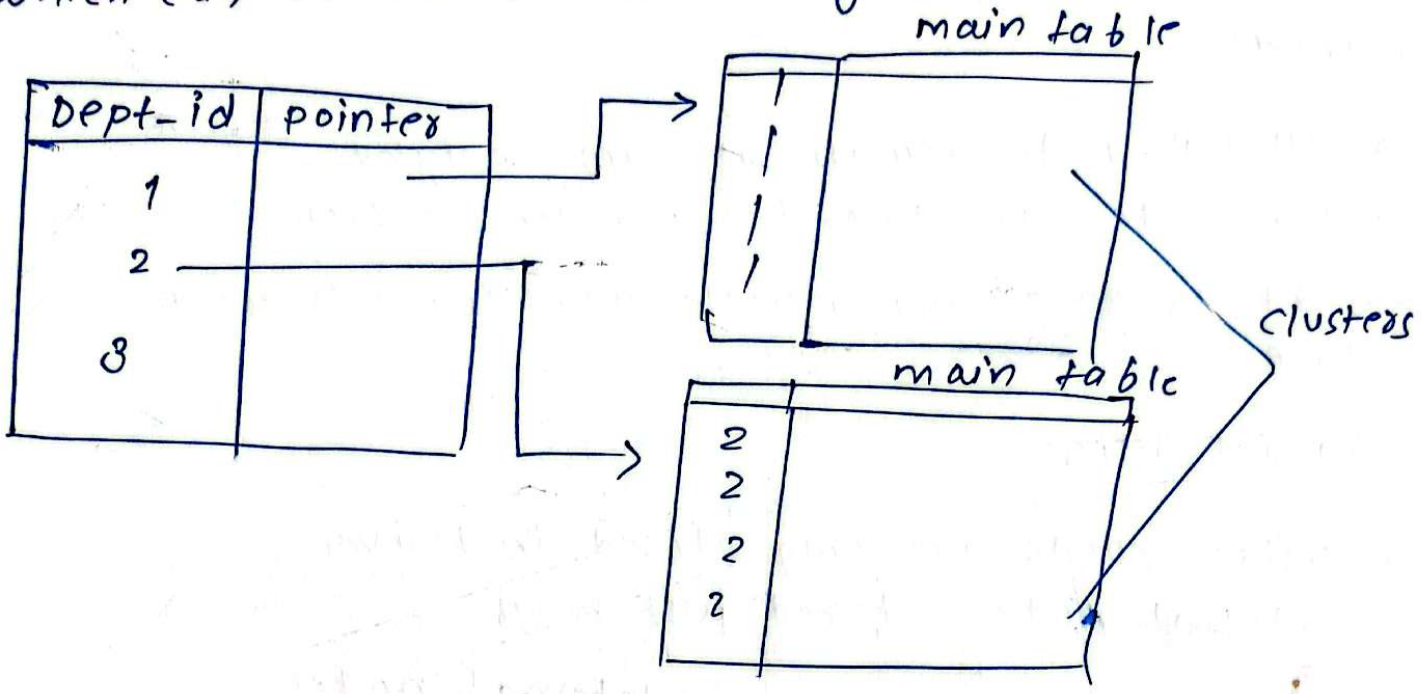




insert 21

2) clustering

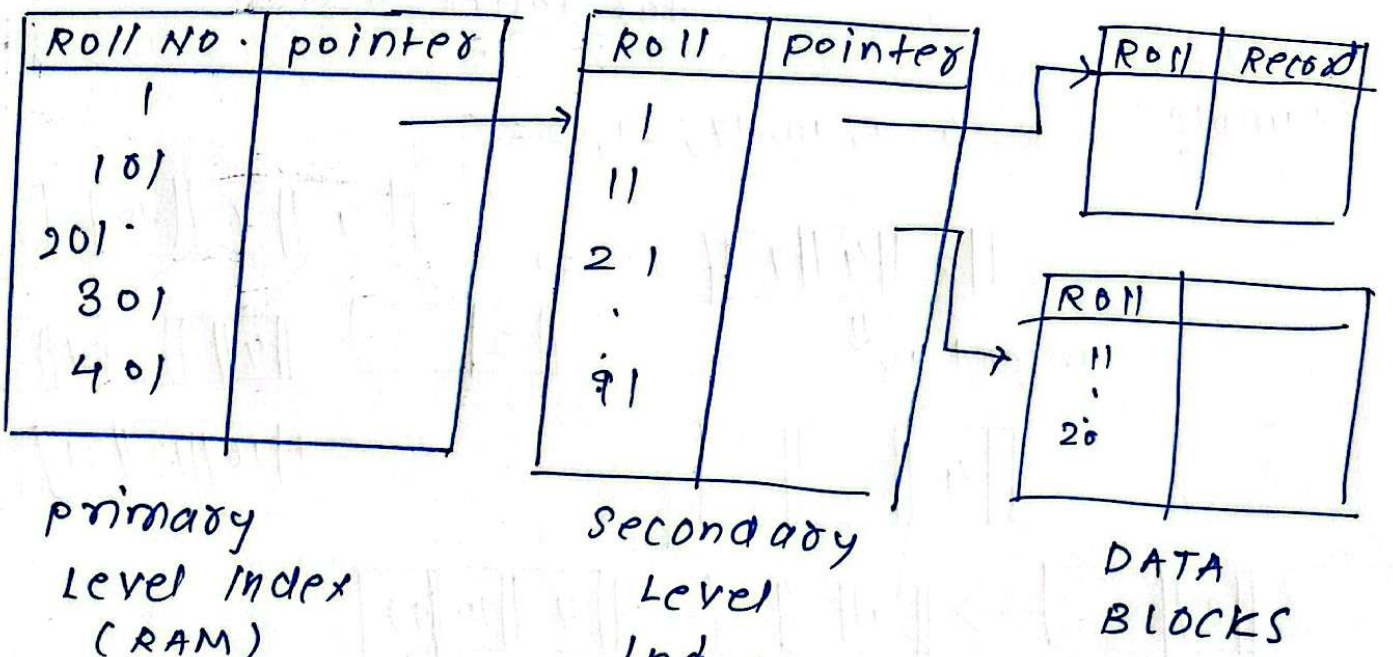
In case of a non-unique key such as dept-id (which can be same for many students)



3) secondary

Two-Level Index:

Secondary indices have to be dense



primary Level Index (RAM)

Secondary Level Index (Hard disk)

DATA BLOCKS

Index points to a bucket that contains pointers to all the actual records with particular search key value

B+ trees

order of tree (m) = 4

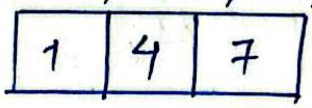
Maximum children = 4

minimum children = $\frac{m}{2} = \frac{4}{2} = 2$

max. keys = (m-1) = 4-1 = 3

min keys = $\left[\frac{m}{2}\right] - 1 = 1$

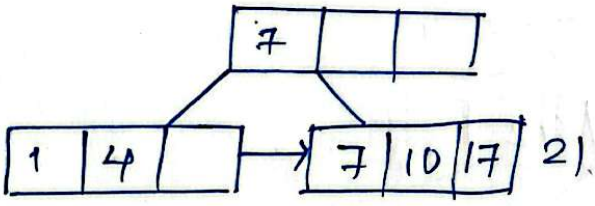
1, 4, 7, 10, 17, 21, 81, 25, 19, 20, 28, 42



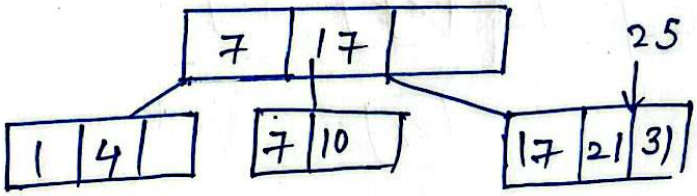
10

Take 7 as min element

RBT

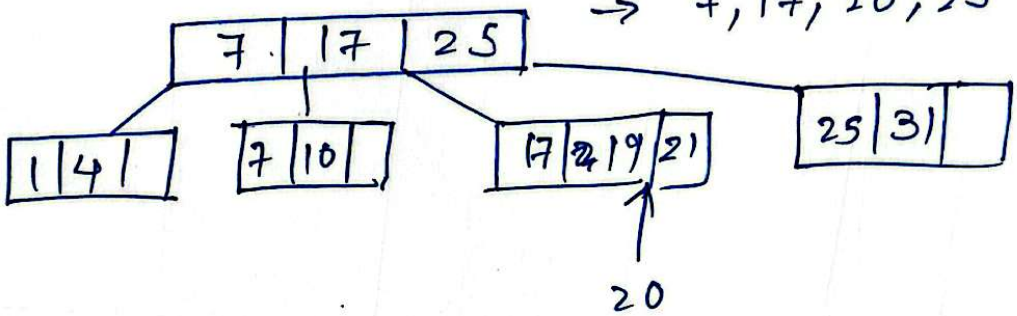


21

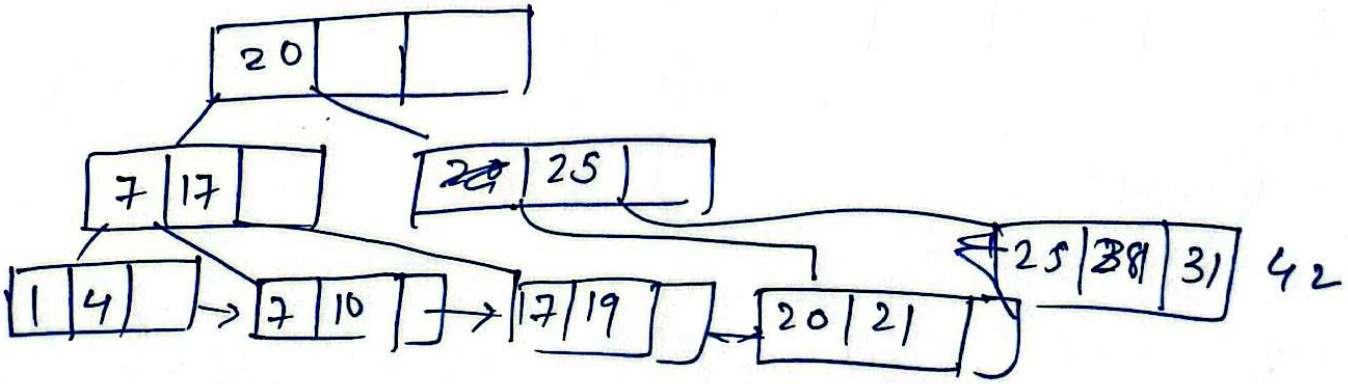


1,4 7,10 17,19 20,21 25,31

→ 7, 17, 20, 25



20



42

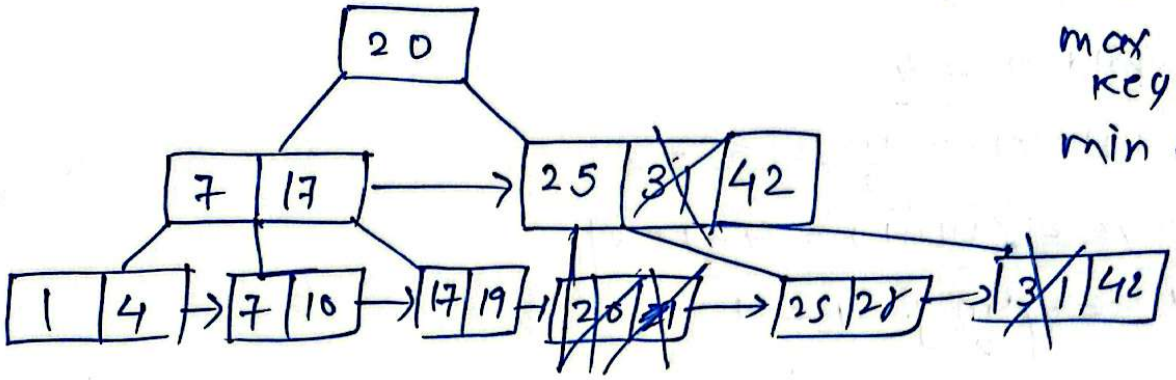
Deletion of B+ trees

max children (m) = 4

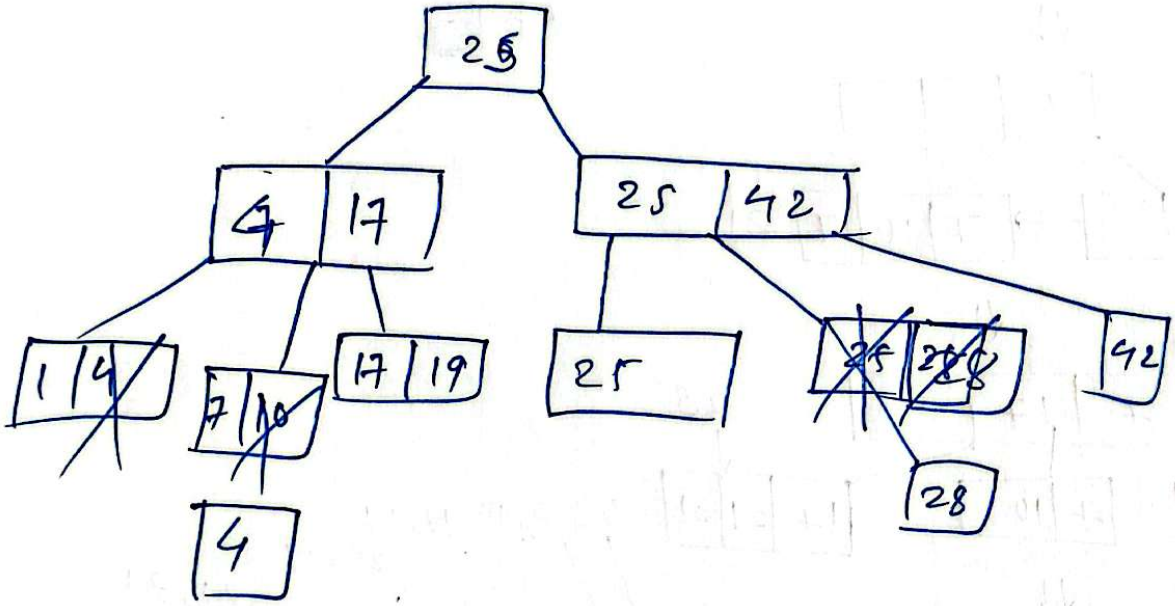
min children = $\frac{m}{2} = 2$

max keys = 3

min keys $\left(\frac{m}{2}\right) - 1 = 1$



Delete 21, 31, 20, 10, 7, 25, 42

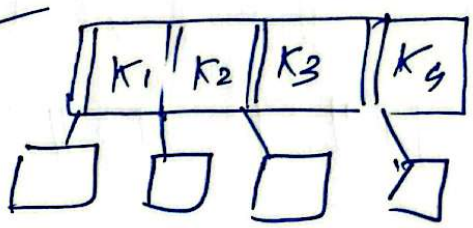


□

B-Tree

- balanced (m)-way-tree (m → order)
- Generalization of BST in which a node can have more than one key & ~~mad~~ more than 2-children.
- All leaf node must be at same level.
- B-tree of order m has following properties

→ Every node has max m children



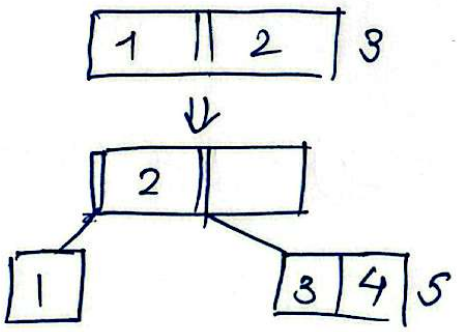
min children :→ leaf → 0
root → 2
internal nodes → $\lceil \frac{m}{2} \rceil$

→ Every node has max (m-1) keys
min keys : root node → 1
other nodes $(\frac{m}{2}) - 1$

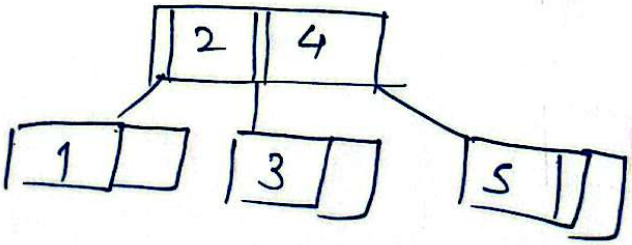
B-Tree :- create a B-Tree of order 3 by inserting values from 1 to 10

max = 3
Max key = (m-1) = 3-1 = 2

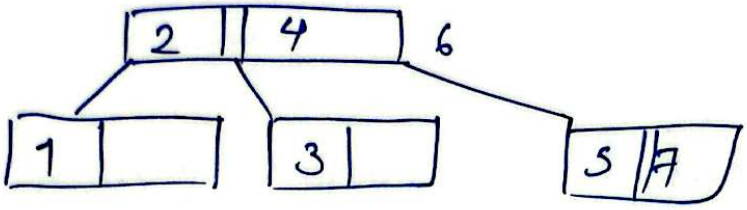
1, 2, 3, 4, 5, 6, 7, 8, 9, 10

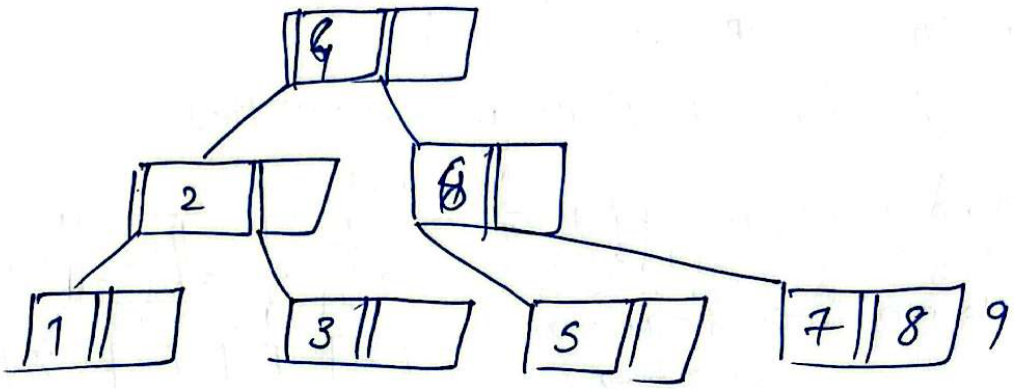


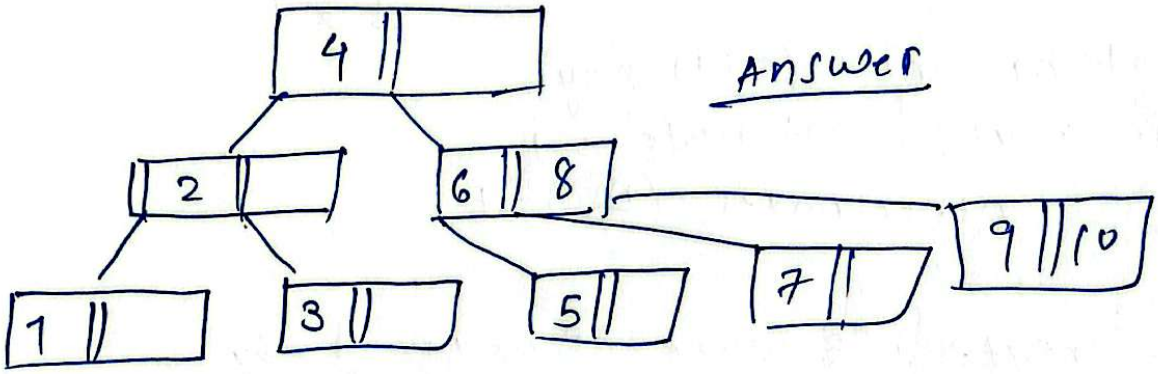
B-Tree of order 3



=>







Answer

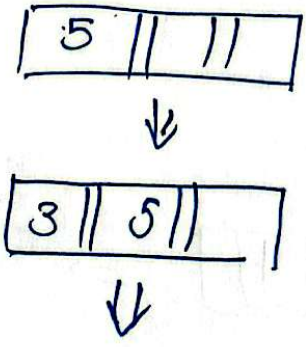
construct a B-Tree of order 4 with following set of data.

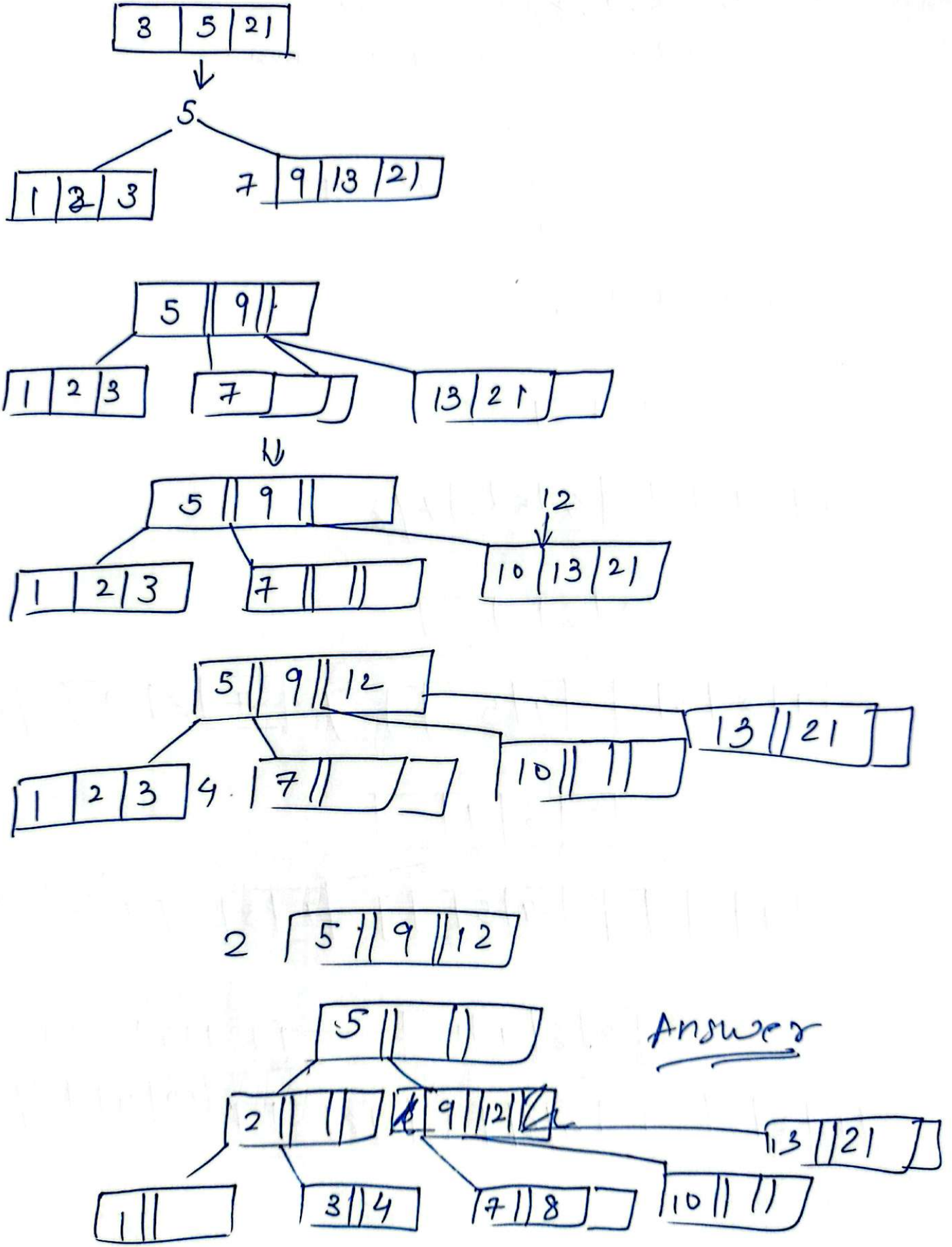
5, 3, 21, 9, 1, 13, 2, 7, 10, 12, 4, 8

$m = 4$

keys $= (m-1) = 4-1$ (4, 3) $min = \left(\frac{4}{2}\right) - 1 = 2 - 1 = 1$

solution




3 5 21
5
1 2 3
7 9 13 21
5 9
1 2 3
7
13 21
5 9
12
1 2 3
7
10 13 21
5 9 12
1 2 3
4
7
10
13 21
2
5 9 12
5
Answer
2
9 12
13 21
1
3 4
7 8
10 11

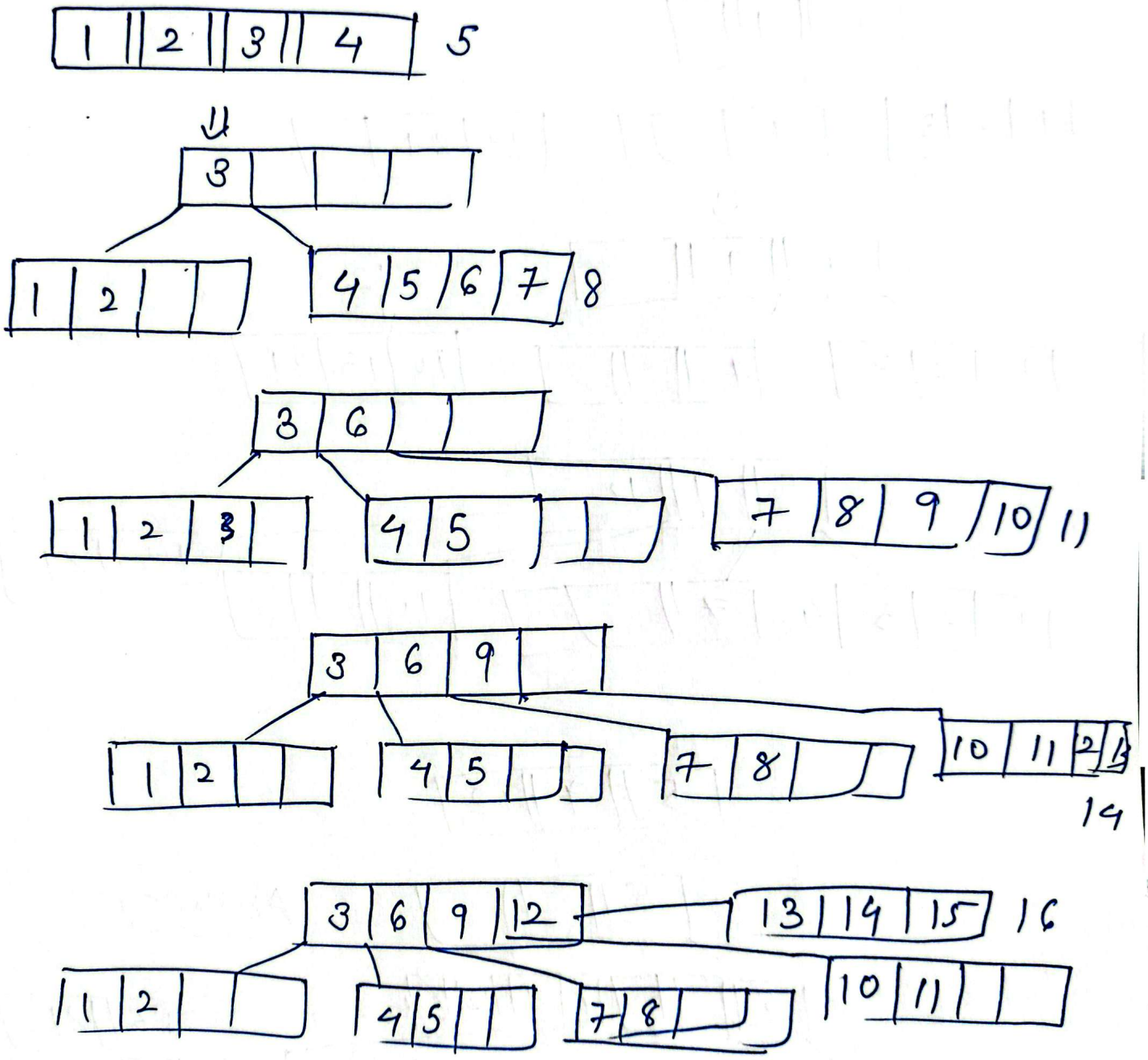
1, 2, 3, 4, 5, 6, 7, 8, . . . 20

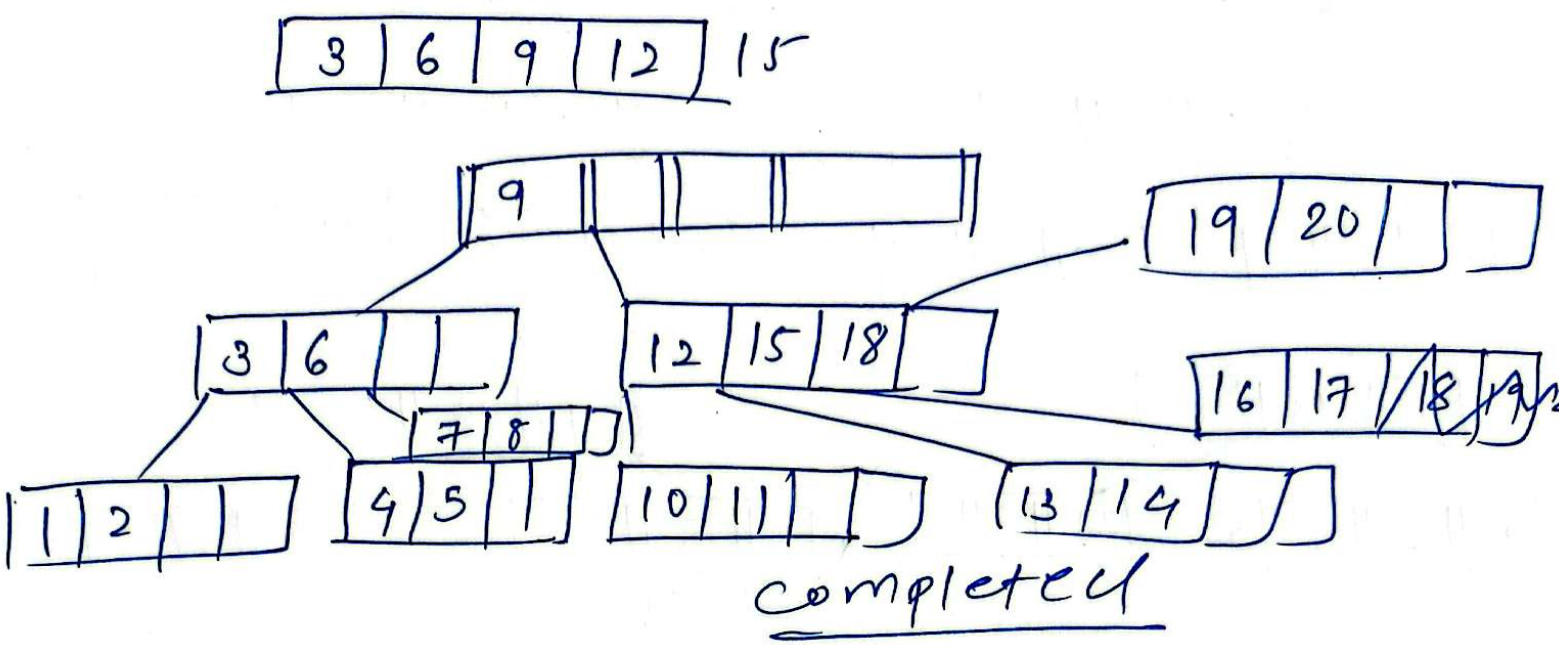
order 5 find B -tree

$m = 5$ max children = 5

$m - 1$ = max keys

4





completed

NORMALIZATION

FIRST NORMAL FORM

→ Table should not contain any multivalued attributes

ROLL NO.	Name	course
1	sai	C/C++
2	Harsh	Java
3	onkar	C/DBMS

multivalued

ROLL NO.	Name	course
1	sai	C
2.	sai	C++
2	Harsh	Java
3	onkar	C
3	onkar	DBMS

Let primary key be ROLL NO

ROLL NO.	Name	course 1	course 2
1	sai	C	C++
2	Harsh	Java	NULL
3	onkar	C	DBMS

Another method

ROLL NO.	NAME
1	sai
2	Harsh
3	onkar

ROLL NO.	course
1	C
1	C++
2	Java
3.	C
3	DBMS

primary key
ROLL NO

foreign key
ROLL NO

Finding closture of functional dependencies.

Closture method

R(ABCD)

C.K = {

F.D { A → B, B → C, C → D }

A+ = ABCD

∴ A is a candidate key

Non prime attribute { BCD }

B+ = BCD

C+ = CD

D+ = D

(AB)+ = ABCD

(AB) = CK ✗

super key ✓

R(ABCD)

F.D = { A → B, B → C, C → D, D → A }

(A)+ = ABCDA

(B)+ = BCDA

(C)+ = CDAB

(D)+ = ABCD

CK = { A, B, C, D }

prime Attribute = { A, B, C, D }

NON prime attribute { ϕ }

SECOND NORMAL FORM (2nd NF)

→ Table should be in 1st Normal form
→ Table or relation must be in 1 NF
→ All the Non-prime attributes should be Fully functional dependent on candidate key(s) or 1

Customer ID	Store ID	Location
1	1	Delhi
1	3	Mumbai
2	1	Delhi
3	2	Bangalore
4	3	Mumbai

Candidate key :

CK = {customer ID, store ID}

Prime Attributes = {customer ID, store ID}

Non-prime Attribute : {Location}.

2NF

Store ID	Location
1	Delhi
2	Bangalore
3	Mumbai

THIRD NORMAL FORM (3rd NF)

→ Table ~~should~~ must be in second Normal form
→ There should to be no. transitive dependency in table.

Roll NO	state	city
1.	Punjab	Mohali
2.	Haryana	Ambala
3.	Punjab	Mohali
4.	Haryana	Ambala
5	Bihar	patna

$CK = \{Roll\ NO\}$

F.D →

Roll no → state ✓

state → city

N.P.A

N.P.A ✗

$CK = \{Roll\ no\}$

$P.A = \{Roll\ no\}$

$N.P.A = \{state, city\}$

∴ Table is not in 3 CNF form

R (ABCD)

F.D AB → C C → D

R(ABCD)

CK: $(AB)^+ = ABCD$

∴ $CK = \{AB\}$ $P.A = \{A, B\}$

$N.P.A = \{C, D\}$

AB → C Valid ✓

C → D Not valid ✗

C → D

N.P.A N.P.A

Not in 3NF form

R (ABCD)

F.D $AB \rightarrow CD$ $D \rightarrow A$

$(AB)^+ = ABCD$ $(AD)^+ = (AD$

$(BD)^+ = \{BDAC\}$

$\therefore CK = \{AB, BD\}$

$\therefore$ prime Attributes $= \{A, B, D\}$

Non prime Attributes $= \{C\}$

$\therefore$ $AB \rightarrow CD$ ✓

P.A N.P.A

$\therefore$ In 3NF ✓

$D \rightarrow A$ ✓

P.A P.A

Note: LHS of all F.D is CK or S.K or RHS is a prime Attribute

BCNF (Boyce codd Normal form)

Roll no.	Name	voter id	age
1.	Ravi	K0123	20
2	Varun	M034	21
3.	Ravi	K786	23
4	Rahul	D286	21

BCNF

CK : {Roll no, voter id}

F.D = {

Roll no. → name

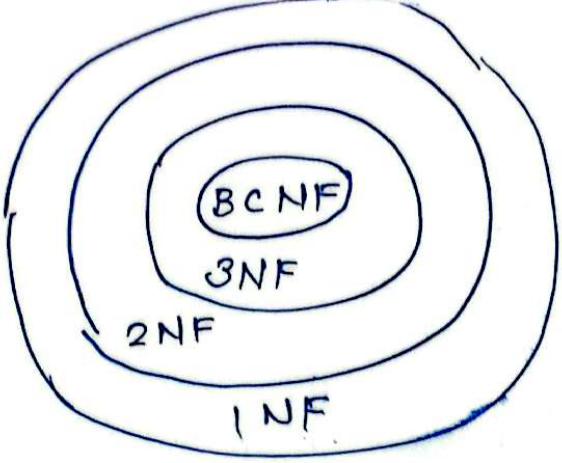
Roll no → voter id

voter id → age

voter id → Roll no.

}

L.H.S of each fonctional dependencies should be candidate key or a super key.



LOSSLESS / LOSSY JOIN DECOMPOSITION

R

A	B	C
1	2	1
2	2	2
3	3	2

→ $R_1(AB)$

A	B
1	2
2	2
3	3

→ $R_2(BC)$

B	C
2	1
2	2
3	2

Find the value of C if the value of A = 1

Select R_2.C from R_2 Natural Join R_1.A = '1'
WHERE R_1.A = 1

A	B	C
1	2	1
1	2	2
2	2	1
2	2	2
3	3	2

common Attribute should be CK or SK of either R_1 or R_2 or Both

Lossy Decomposition

For the Functional dependencies, Find correct Minimal cover

$\{A \to B, C \to B, D \to ABC, AC \to D\}$

$A+ = A$

$C+ = C$

$D+ = \cancel{BB}\ DBC$

step 1

$(A \to B)$, $C \to B$, $D \to A\cancel{BC}$, $\boxed{D \to B}$, $D \to C$, $AC \to D$

Redundant

step 2

$A \to B, C \to B, D \to A, D \to C, AC \to D\}$

step 3

$A \to B, C \to B, D \to A, D \to C, AC \to D\}$

$\times\ \times$

$AC \to D$

$(C)^+ = C$

concurrency control protocols.

Shared - Exclusive Locking

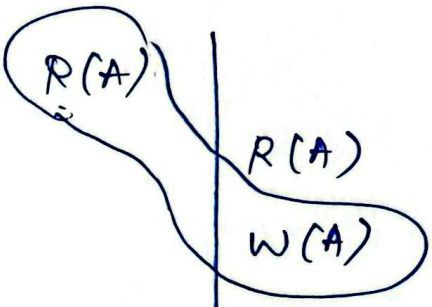
Shared Lock (S) :- If transaction locked data item in shared mode then allowed to read only

Exclusive Lock (X) :- If transaction data item in exclusive mode then allowed to read and write both

T1	T2
S(A)	
R(A)	
	X(A)
	R(A)
U(A)	
	W(A)
	U(A)

	S	X
S	yes	No
X	No	No

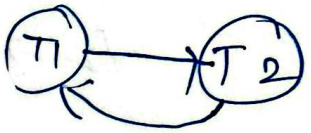
(A)



Exclusive lock

shared locking

T1	T2
X(A)	
R(A)	
W(A)	
U(A)	
	S(A)
	R(A)
	U(A)
X(B)	
R(B)	
W(B)	
U(B)	



loop producing

problems in S/X locking

① May not be sufficient to produce only serializable schedule.

2) May Not be free from ~~~~ Irrecoverable

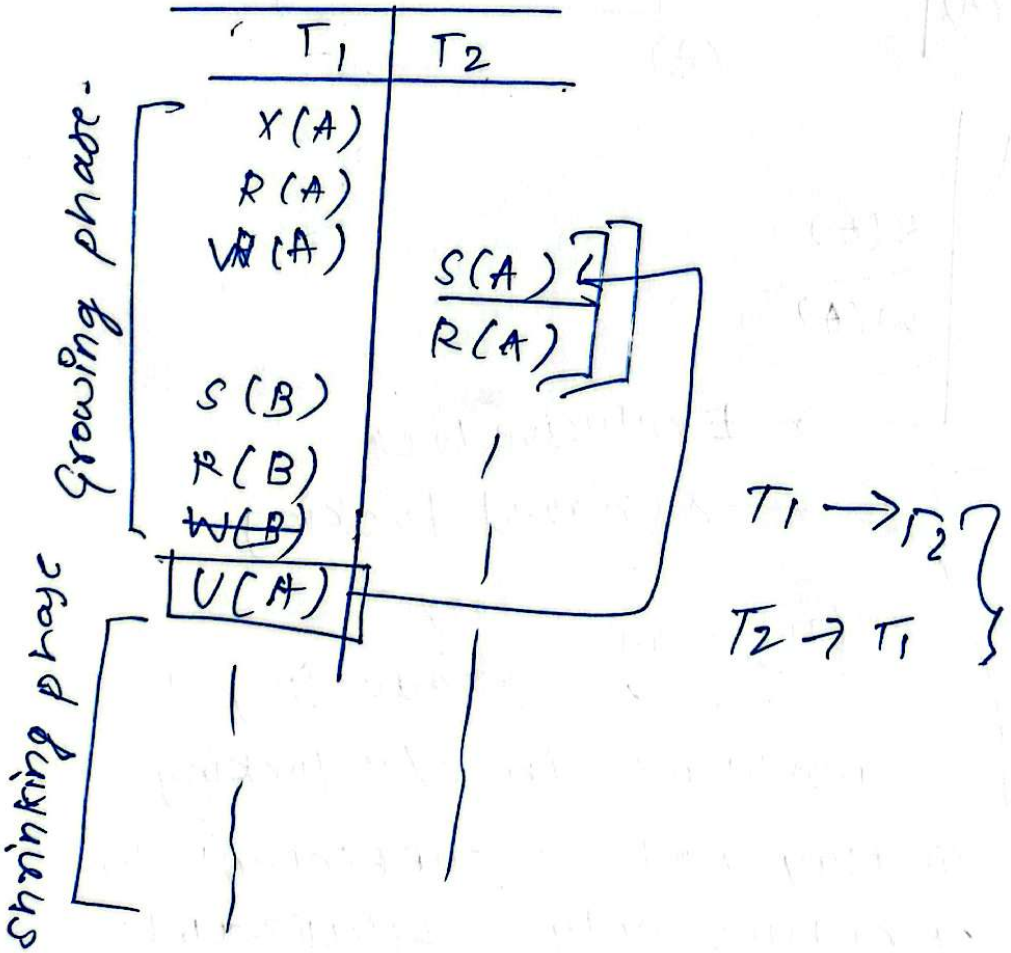
3) May Not be free from deadlock

4) May not free from starvation

2-phase Locking (2PL)

Growing phase:- locks are acquired and no locks are released

Shrinking phase:- locks are released and no locks ax acquired.



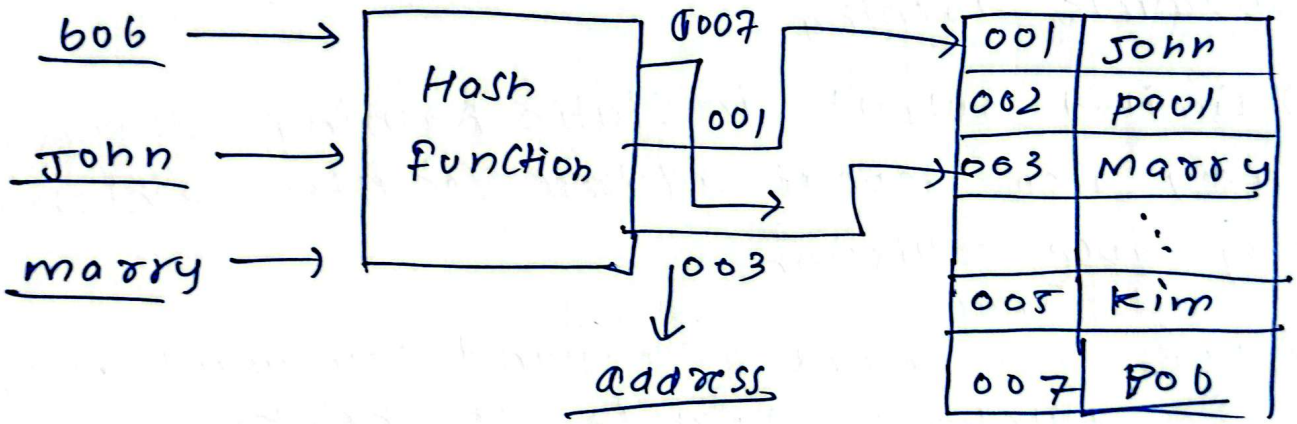
Hashing Techniques

Hashing!- Hashing is an effective technique to calculate the direct location of a data stored or data record on disk without using index structure.

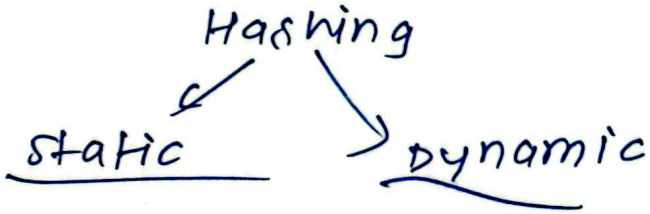
Hash organization :-

Bucket :- A Hash file stores data ~~in this~~ in bucket is considered a unit of storage

Hash Function :- A hash function h is a mapping function that maps all the set of keys address where the actual records are placed.



Search key has only 1 address



i) Static Hashing :-

In static Hashing, when a search key value provided, the Hash function always computes the same address.

mod(x) = 0 → x-1

mod(5) = 0 → 4

10/5 = 0

If we want to generate address for a key using mod(5) hash function it always results in the same (0 → 14)

No. of data buckets in the memory for static hashing remains constant

Dynammic Hashing

i) Limited output in static hashing which may also repeat address location which may cause collission

static Hashing

ii) Does not shrink or expand Dynammically as database expands or shrinks