GATE = 8
NET = 20

DBMS

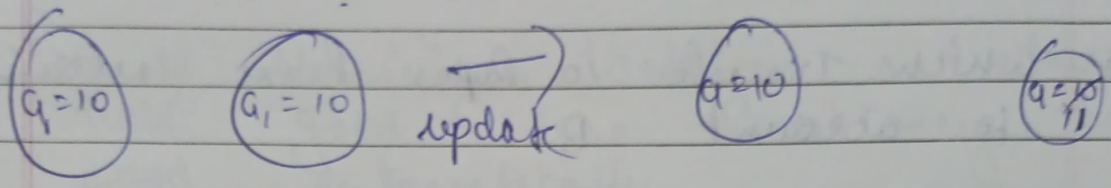
BOOK :- Korth & Navathe

Q. DBMS [Transaction and Concurrency]

Relational

Redundancy Inconsistency

A_1 A_2



MARKS :-

K → (1) E-R diagram (1-2)
N → (2) F.d & Normalisation (2-4)
N → (3) physical struct (B, B+) & Indexing (2-3)
K → (4) Query language (Relation Algebra & Relational Calculus, SQL) (2-3)
Procedural, Non-procedural
K → (5) Transaction & C.Concurrency Control (2-3) marks.

TRANCTION

* Instructions are always atomic in nature that is either a instruction executes Completly or it doesn't execute at all. But it is possible to have partially executed programs which means some instructions are executed But some are not.

Defination :- # According to user Either a work is done or not done Therefore a Transaction is that set of #

Instruction which performs a logical Unit of work according to user.

Ex => A —10→ B

which Transfer 10 Rupees from account A to account B

R(A)
A = A - 10
W(A)
R(B)
B = B + 10
W(B)

Property of Transaction :-

** ACID **

* If we want that our database should be Consistant that we understand that transactions which operates on database must satisfies ACID Properties.

Atomicity :-

* A stands for Atomicity :- It states that either all the instruction participating in a transaction will execute or none.

* Atomicity is Guranted By Transaction management Component.

* Consistency * :- meaning (Consistent behaviour)

* It states that if a database is Consistent before the execution of Transaction then it must Remain Consistance after execution of Transaction.

J.M.P * If Atomicity, Isolation and durability holds good then Consistancy holds good automatically.

* Isolation * :-

* Isolation means if a transaction Run Isolatly or Concurently with other transaction than the Result must be same.

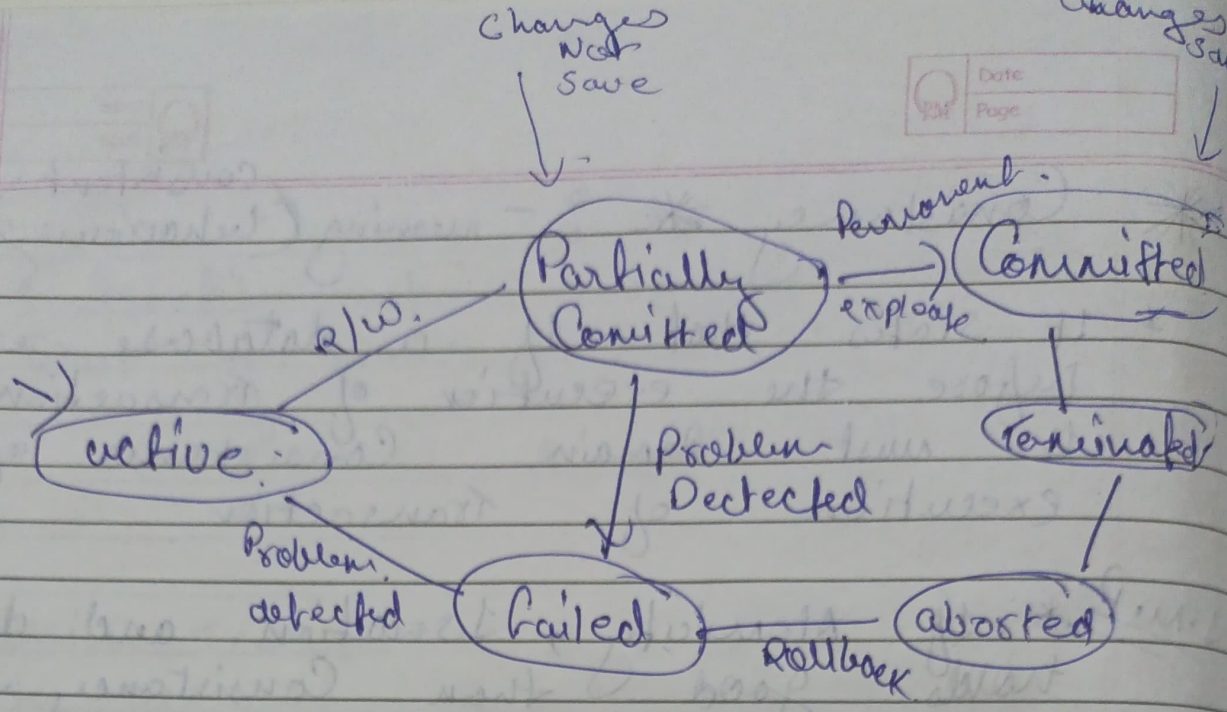
* Concurrency Control Component takes cares of Isolation.

* Durability * :- meaning (storing of data)

* Durability means that a work done by a successful transaction must Remain in system. Even in Case of any Hardware or Software Failure

[redacted] in Recovery management Component take Care of Durability.

→ Transition states :-



Active :- A Transaction is said to be in active state if instructions are said to be in Executing.

Partially Commited :- Partially Commited state means that all the instruction are executed but changes are Temporary And Not updated in Database.

Commited :- when changes are made to be permanent then that state is called Commited

Failed :- If any Problem is Detected either During active state or partially Commited state then transaction Enters in failed states.

aborted :- aborted state means all the changes that were in the local buffer are deleted.

Either we are Committed or aborted database Consistant.

Advantages of Concurrency

1. Reduced waiting time / Response time / And Turnaround time.

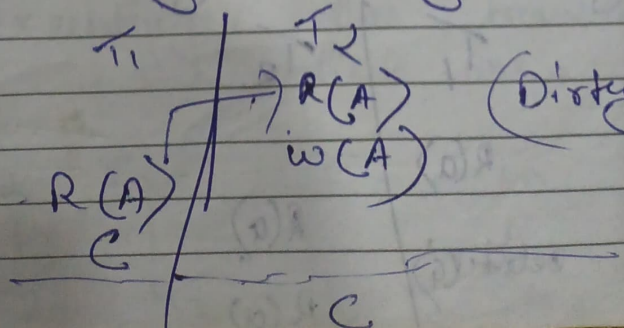
2. Increased throughput or Resource Utilisation.

3. If we run only one transaction at a time than ACID Property is sufficient. But it is Possible that when multiple transaction are executed Concurrently then database may became inconsistance.

Problems because of Concurrency :-
OR (Read-write problem)

(1.) Dirty Read Problem :- If a transaction Reads a Uncommited temporary value written by some other transaction then it is called dirty read problem.

→ some times dirty Read may lead to inconsistan



② Write / write Clash / lost update Problem

T_1	T_2
R(A)	
W(A)	
	W(A) ← Blind write
	C
C	

Here the changes made by transaction T_1 are lost which is updated by a Transaction T_2.

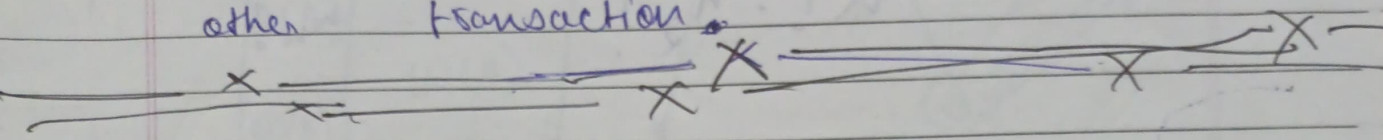
T_1	T_2
10 - R(a)	
	Ex=23
	R(A) - 10
11 W(a)	
	R(A) - 11

(phantom read problem)

③ Unrepeatable Read problem :- Here Transaction T_2 cannot repeat its Read instruction because the value is being updated by transaction T_1. (Ex=23)

T_1	T_2
R(a)	
	R(a)
delete(a)	
	R(a)

4. Infantom Read problem :- a transaction cannot Repeat its Read instruction because that variable is deleted by some other transaction.



Schedule :- A schedule contains two or more transaction executed together or one after other.

1. All the instruction of each individual Transaction will appear in the schedule

2. Context Switching can be done but we cannot change the order of execution.

3. Serial Schedule :- If a schedule Runs only one transaction at a time and can start other Transaction after Completing the first then it is called Serial Schedule.

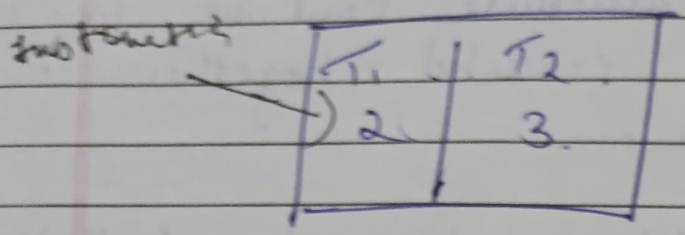
4. If there are n transaction then (n different schedule are possible

NON Serial schedule :- A schedule is set to be non serial if we start executing other transaction before Completing the first one.

T_1 T_2 T_3 . . . T_n

n_1 n_2 n_3 n_4

$$\frac{(n_1 + n_2 + n_3 + \cdots n_n)!}{n_1! \cdot n_2! \cdots n_n!} = N!$$



$2! = 2 \Rightarrow$ serial

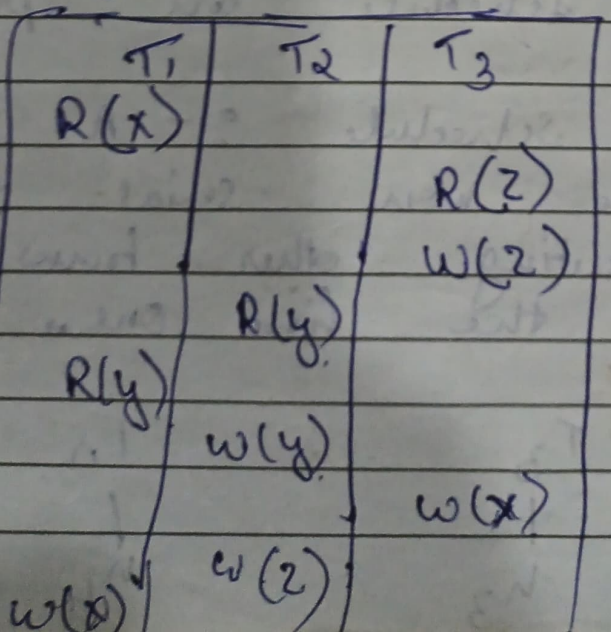
$\frac{(2+3)!}{2!\,3!} = 10!$

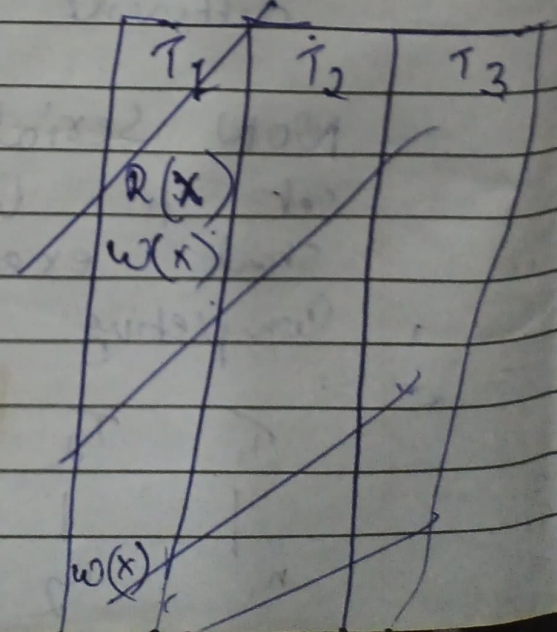
(Total serial) $\frac{5 \times 4 \times 3 \times 2}{2 \times 3 \times 2} = 10!$ (Total serial)

T.S - serial = N.S.

$10 - 2 = 8$ (Non-serial)

Consider a schedule and convert it into a serial schedule by swapping of non conflicting instruction.

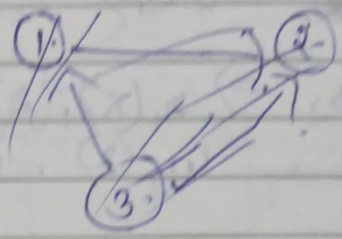


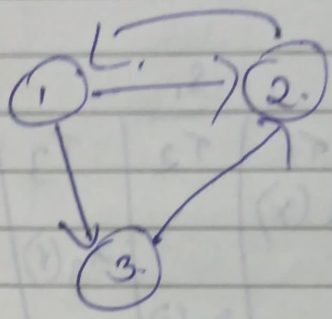


-) Conflict (meaning) :- Collision & disagreement or opposition

-) Non conflict (means) :- Compatible

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

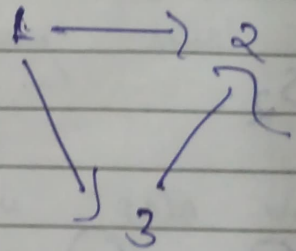




Hence It is not a Conflict Civilisable.

T_1	T_2	T_3
R(x)		
	R(Y)	
		R(Y)
	W(Y)	
W(x)		
		W(x)
	R(x)	
	W(x)	

3 Conditions
1. Different Transaction
2. Same data
3. Write operation



T_1	T_2	T_3
R(x)		
W(x)		
		R(X)
		W(x)
	R(Y)	
	W(Y)	
	R(X)	

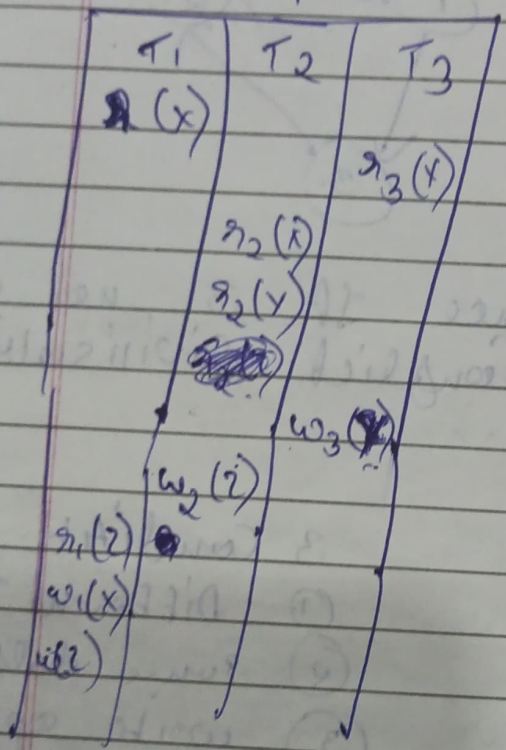
serial schedule

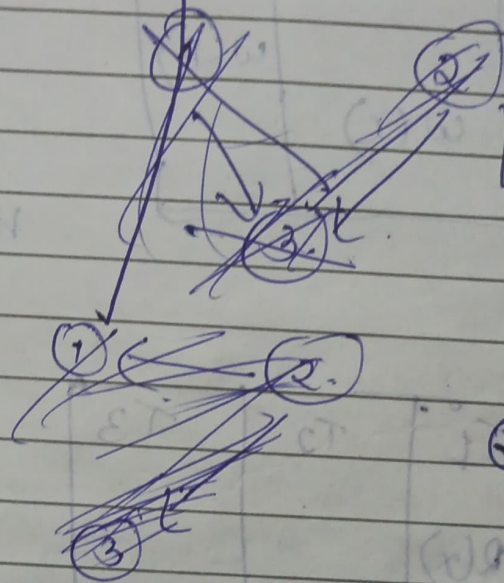
Gate

① $S_1 \Rightarrow r_1(x), r_3(y), r_2(x), r_2(y), r_2(z), w_3(y), w_2(z), r_1(z), w_1(x), w_1(z)$.

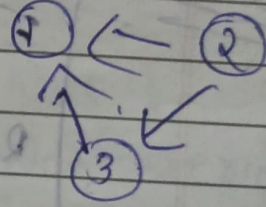
$S_2 \Rightarrow r_1(x), r_3(y), r_3(x), r_1(z), r_2(z), w_3(y), w_2(x), w_2(z), w_1(z)$

If we get cycle then it is not Conflict Serializable.





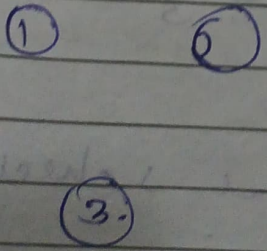
Ans :- 231



It is Conflict Civilizable because It has no cycle. we can translate it in serial

↗ Ans (231)

S_2



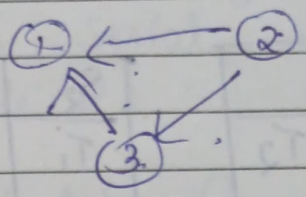
S_1 :- 2RA, 2WA, 3RC, 2WB, 3WA, 3WC, 1RA, 1RB, 1WA, 1WB

S_2 :- 3RC, 2RA, 2WA, 2WB, 3WA, 1RA, 1RB, 1WA, 1WB, 3WC, (231)

S_3 :- 2RA, 3RC, 3WA, 2WA, 2WB, 3WC, 1RA, 1RB, 1WA, 1WB

S_1

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



Serial Schedule will always be Consistance in nature. But a non serial schedule may or may not be.

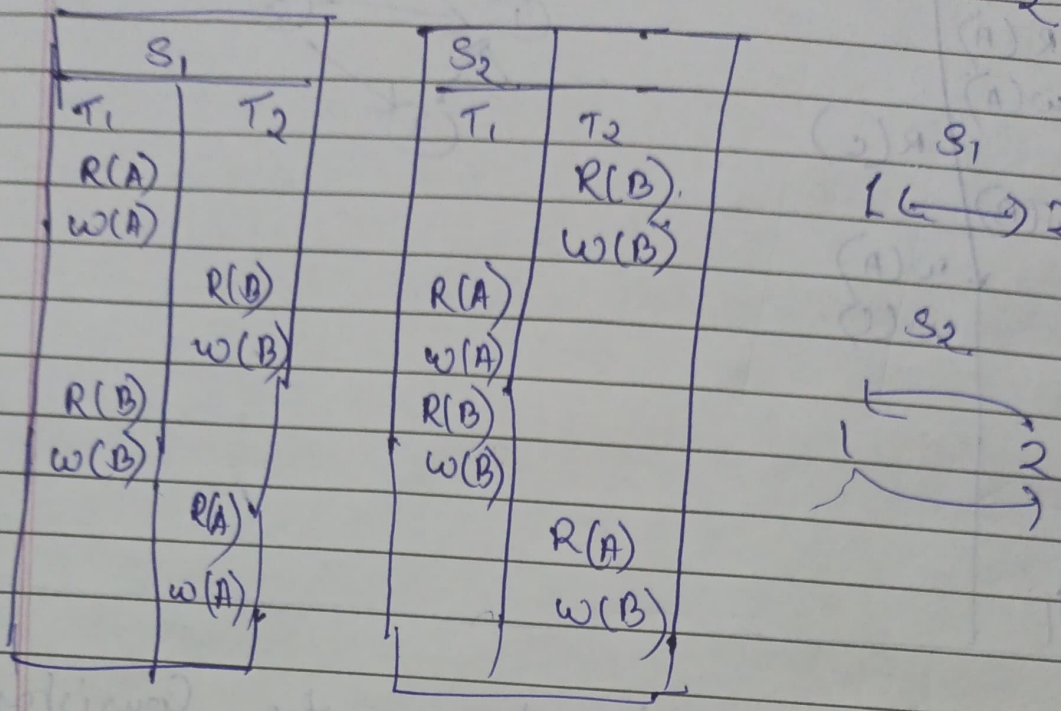
Therfore the Idea is to transform a non-serial schedule into a serial schedule by swapping of non-Conflicting instruction.

Two instruction I_i and I_j are said to be Conflicting if ① they must belong to different transaction.

② Must operate on same data value

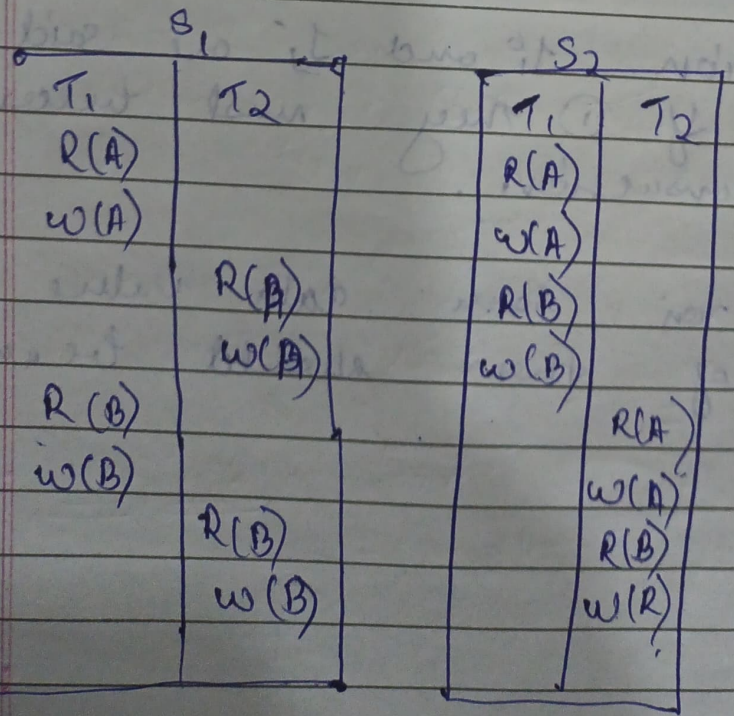
③ Atleast one of them should be a write instruction.

Consider Two schedule S_1 and S_2.



what are Conflict Equivalent schedule :-

Two schedule S_1 and S_2 are said to be Conflict equivalent if it can be transform into another by swaping of non-Conflicting instructions.



Here S_1 and S_2 are Conflict equivalent but because S_2 is Serial Hence S_1 is Conflict serializable.

* If a non-Serial Schedule Can be transformed into a Serial Schedule by swapping of non-Conflicting Instruction then it is called Conflict Civilizable.

OR

* If Two schedules are Conflict equivalent and one of them is Serial Then other is Conflict Serializable.

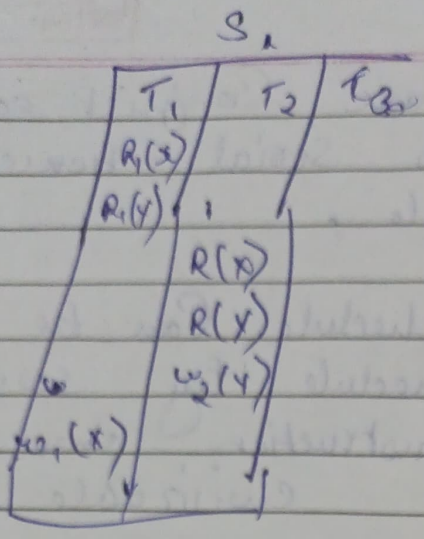
* The problem of finding whether a Schedule is Conflict Civilizable or not is actually the problem of finding whether a graph Contain Cycle or not therefore Cost of ordering n^2. where n is the number of Transistion participating in Schedule.

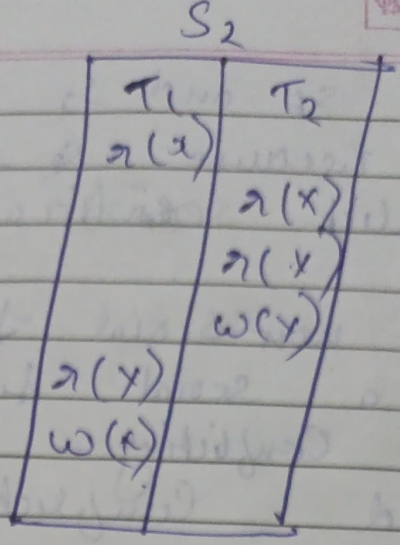
Q. $S_1 = R_1(x), R_1(y), R_2(x), R_2(y), W_2(y), W_1(x)$

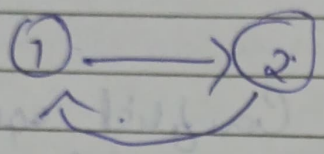
$S_2 = R_1(x), R_2(x), R_2(y), W_2(y), R(y), W_1(y)$

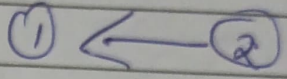
C.S → O(n²)
V.S → NPC

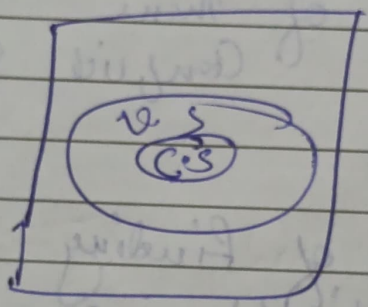
Sol:-











* It is Theoratically not possible to check whether a schedule is Consistent or not. Therefore we apply two filters of Conflict Sirializibility and view Serializibility but even if a schedule is not C.S or view.S. then also it Can be Consistant.

* How to find whether the Schedule is V.S or not?

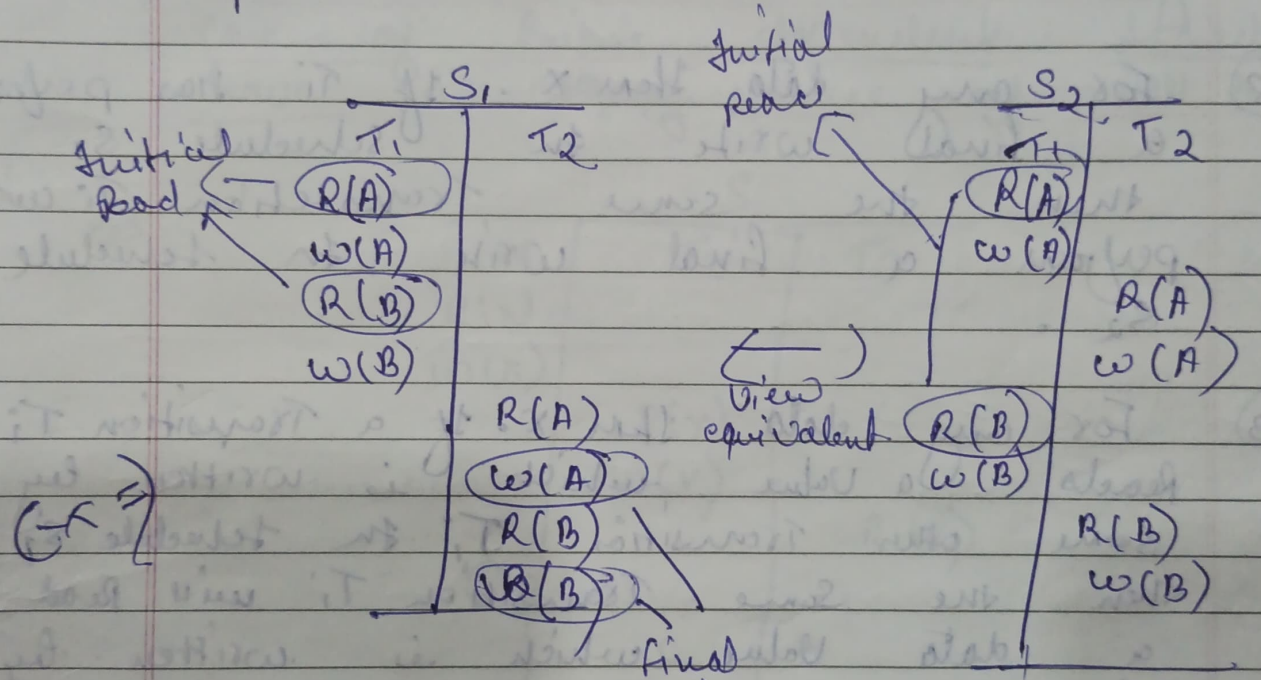
→ Step 1. Find Conflict serializibility if yes then also view serializable otherwise go to step 2.

C.S. :=) By swapping of non-serial to serial.

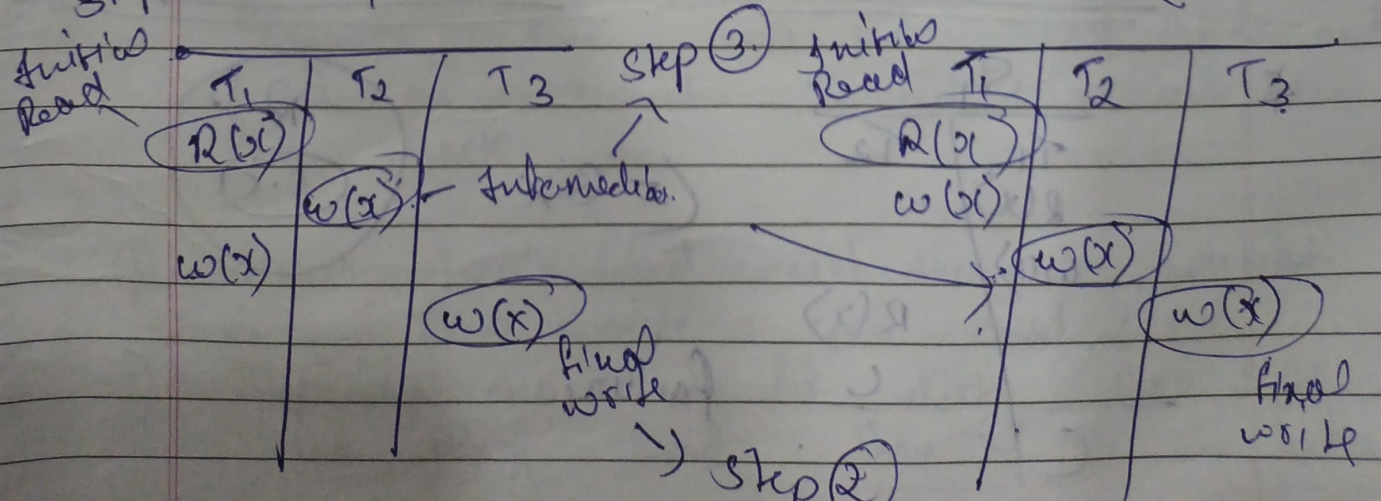
Step 2

check if there exist blind write if yes then goto step - 3. otherwise Not view serializable.

Step 3 :- check the three conditions of view equivalence with every serial schedule possible.



(-X=)



View Serializable

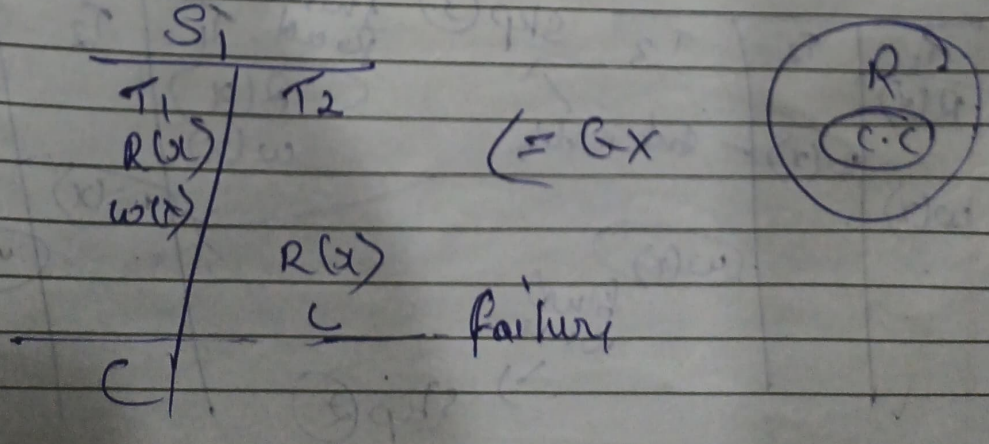
Q what are the three conditions for equivalence?

Sol:- ① for any data item x if a transaction T_i performs a initial Read in schedule S_1 then the same Transaction T_i will perform a initial Read in schedule S_2.

② For any data item x. If T_i has perform a final write in schedule S_1 then the same Transaction T_i will perform a final write in schedule S_2.

③ For any data item x. If a Trasition T_i Reads a value which is written by some other Transition T_j in schedule S_1 then the same Transition T_i will Read a data value which is written by T_j in schedule S_2.

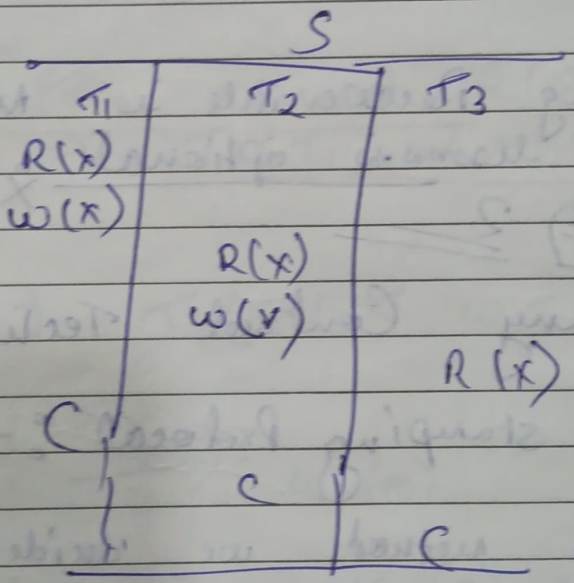
④. finding whether a schedule is View serializable or not it is a NPC problem



* This schedule is Conflict Serializable but if there occurs a Failure before two Commits operations then database will become inconsistant.

* what are Recoverable Schedule ?
→ A schedule is recoverable if in case of failure dependent schedule have a chance of Roll back.

*



This schedule is Recoverable but if failure occurs just before Transition Commits then T_1 will Roll back and based on which T_2 and T_3 will also Roll back.

Q. How to find whether a schedule is Recoverable or not ?
→ Step 1. Search for dirty Read. If no dirty Read Then it is Recoverable

as well as Cascade less.

2. If There Exist a dirty Read Then not Cascade less and goto step 3.

3. If the order in which dirty Read is done is same with the order in which transaction Commits then Recoverable otherwise not.

T.M.P Being Recoverable is mandatory but Cascade lessness is optional. ✗

Part ⇒ 3

Concurrency Control Technique

Time stamping Protocol :-

In this method we decide order before transactions enters into the system here every Transaction T.I. is given a unique Time Stamp T.S.(T.I) which is actually the value of system clock when the Transaction enters into the system.

This Timestamp will Remain Constant until the Transaction is in the system.

will every data item with we associate two data Queue. Read Timestamp of Queue and write Time stamp Que.
R.T.S (Q) & W.T.S (Q)

R.T.S (Q) :- It is the Time Stamp of the transaction who has performed latest Read operation.

If a Transaction T_i Request Read operation on Queue

(1) If T.S. (T_i) request R(Q)

(a) Case I => $Ts(T_i) \geq W.T.S(Q)$ ✓
→ accept

(b) Case II => $Ts(T_i) < W.T.S(Q)$ ✗
Reject.

(2) If $T_i \longrightarrow W(Q)$

(a) If $T.S.(T_i) \geq R.T.S(Q)$ ✓
(b) If $T.S(T_i) \geq W.T.S(Q)$ ✓
(c) If $T.S.(T_i) < R.T.S(Q)$ ✗
(d) If $T.S.(T_i) \leq W.T.S(Q)$ ✗

NOTE :- If Read operation is allowed then Read Time stamp must be updated and If write operation is allowed then If write time stamp also be updated.

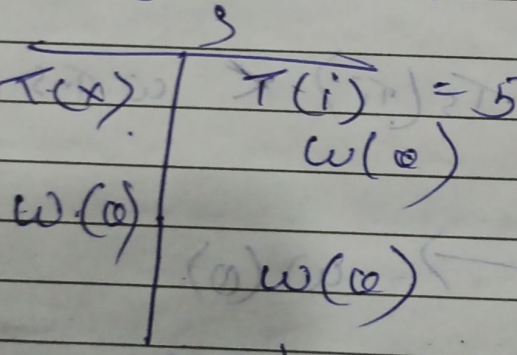
If operation is not allowed then Transaction must enters into the system with a new time stamp.

Conclusion :- Time stamping ensures Conflict serializability & view serializability and independence from deadlock recoverability and Cascade lessness is not gurunteed.

⇒ Thomas write Rule ⇐

If T.S.(T_i) < W-T.S(Q) ×

5 10



In Thomas write Rule we suggest a modification in time stamping protocol where if T_i Request a write Queue

$TS(T_i) < W\text{-}T.S(Q)$

Then instead of Roll Backing a Transaction we Can Just Ignore the write operation.

Conclusion :- If a schedule is generated according to Thomas write then view serializability then

but Conflict
Serializability, Recoverability and Cascadelessness is not Guranteed.

Lock based protocol :- In lock based protocol basic idea is first to acquire a lock before accessing a data item directly. after use should delete that data item.

Conclusion :- Because of this we can avoid clash

here Transaction uses two types of lock shared lock and Exclusive lock.

Shared lock :- If a transaction has acquired shared lock on a data item Q. then it can perform only Read operation.

But if a Transaction has acquired a Exclusive lock then it can perform Read as well as write operation.

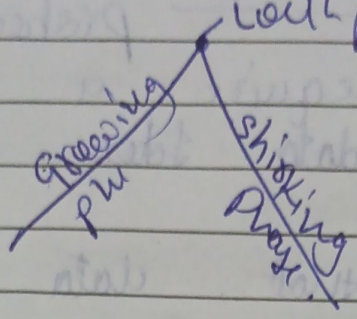
	S	X
S	T	F
X	F	F

T → R → S(Q)

→ W → X(Q)

Two 2 phase locking (2PL) :- In 2PL every Transaction works in two phases in the entire life span.

(1) Growing phase (2) shrinking phase



In growing phase :- only a transaction may acquire lock But Cannot Release any lock.

while in shrinking phase a transaction Can only release lock But Cannot acquire any lock.

growing and shrinking phase Terminate is applicable to transactions not schedule.

- Starvation may be possible.
- Cascading Roll back is Possible.

Property :- (1) Ensure Conflict's and view serializability But fails on Recoverability, Cascadelessness and deadlock.

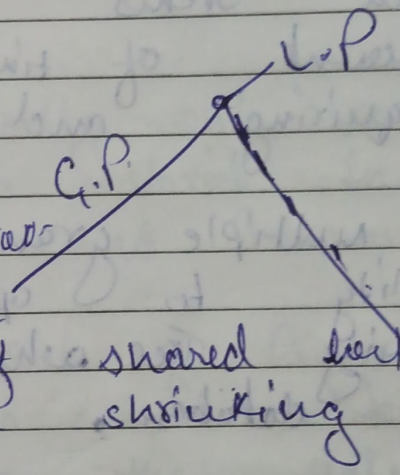
(2) Regrous 2PL :- In Regrous 2PL we Remove shrink phase from the syst. that is a Transaction Can acquire locks But cannot release any lock.

Growing → K L.P.

Because of which dirty Read is not possible. This protocol ensures Conflict serializability, Recoverability and Cascadeness But may suffer from deadlock.

Disadvantage :- (1) Concurrency Reduces

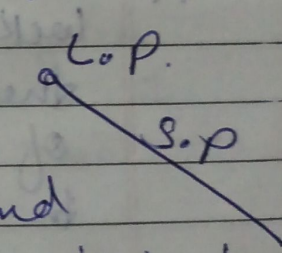
Strict 2PL :-



Strict 2PL is a improvement over Rigorous 2PL where unlocking of shared lock is allowed in the shrinking phase

Property same as Rigorous 2PL.

Conservative 2PL :- In Conservative 2PL we remove ~~static~~ growing phase from the system and every transaction is first required to acquire all the locks first before performing any Read or write.



Property :- It ensures Conflict Serializability view serializability, deadlock Independence, but suffers from Recoverability and cascadeless.

	IS	S	IX	X
IS	T	T	T	F
S	T	T	F	F
IX	T	F	T	F
X	F	F	F	F

Multiple granularity :- When some transaction uses large number of data items then a considerable amount of time is wasted while acquiring and realising a lock.

(2) In multiple granularity we give a facility to apply locks in a locking tree which may vary data to database.

Intention share :- If a node is locked in intension shared mode then it means some subset of that tree or node is locked in shared node.

T.M.P. :-

(1) In Time stamping protocol order of Serialisability is the order in which Transaction enters into the system

(2) In case of two phase locking the order of serialisability is the order in which Transaction acquires the lock point.

DBMS

Raw facts and figure about an object is called data.
Process data is called Information which means it is the data on which we can take some action.

→ Collection of Related data is called data Base.

Here we are working only on Text type of data which is stored in 2D table's divided into Rows and Columns.

→ DBMS is a set of Programs which can store, access and modify data in the data Base, in a secured and effient manner and we must take care that system should be Reliable and Even in case of failure How much time It will take to Recover?

For Ex :- oracle, my sql server. these are Example of DBMS. which are used in almost every filled like Banking, education, telecommunication, transportation etc.

Q/6.

OLAP.	OLTP.
OLAP stand for online Analytical Processing	Online Transaction Processing
① It Deal with Historical data	① It deals with day to day data or Current data
② OLAP Concentrate on decision making, Policy designing etc.	② OLTP It Concentrates on Correct execution of Day to Day operation.
③ OLAP size is of large size. normally Tera Byte and PetaByte.	③ while OLTP is Smaller normally In Mb or GB.
④ OLAP requires only Read operation.	④ while OLTP Requires Read as well as write operation.

→ Advantages of database system over file System.

① Independence from data Redundancy and inconsistency.

② It is easy to design and modify Schema.

③ It is easy to Retrive or access any data using query languages.

(4) It is easy to maintain Integrity & Reliability security etc.

(5) DBMS gives Independence from programming languages and may Connect database to any programming language.

E-R Diagram (1-2 marks)

→ An Object is called Entity if it Can be identified in a group of Object Based on the values of attributes it possess.

→ Entity Can be of two types
(A) tangible :- which Exist physically.
Ex :- Bank locker,

(B) Intangible :- which Exist logically for
ex :- Bank account.

→ Collection of similar type of entities is called Entity set.
In a E-R diagram n Entity set is Represented By Rectangle while in a Relation model By a independent table.

J.M.P :- we Cannot Represent Entity in a E-R diagram as it is a instance or value while E-R diagram is studied for structure or schema

In a Relational model Entity is Represented by a Row or a tupple.

→ Super Key :- A set of Attributes which can uniquely Identify all the other attributes in table is called Super Key.

→ Candidate Key :- Minimal Super Key is called Candidate Key. that is a Super Key whose proper subset is not a Super Key.

→ Primary Key :- is Technically same as Candidate Key But It is that Candidate Key which is selected By database administrator as Primary Source to Identify Tuples (Rows).

→ A Entity Set is said to be weak entity set If it do not have a primary Key.

→ Attributes :- Attributes are the properties on which we can define Entities. In a ER diagram they are Represented By a oval Connected to a Rectangle But in a Relational model Rep they are Represented by Columns.

Types :- —

(A) Simple and Composite :- ① If a attribute can be divided further then it is called Composite otherwise simple.

② In a E-R diagram Composite attribute is Represented when a oval is connected to a oval. But in a Relational model every part of Composite attributes becomes a new Columns.

B) Multivalued :- Attribute is set to be multivalued if it can take more than 1 value at time.

Ex :- ph.no. or address

In a ER diagram Multivalued is represented by a double oval while in a Relational model by Independent table.

C) Stored and derived :- Attribute is said to be stored if its value is permanently saved in database but if the value is derived or Calculated at Runtime then it is called derived attribute.

Ex = Data of Birth while age is a derived attribute

In a E-R diagram derived attribute is represented by dotted ellipse.

→ Relationship :- Relation specify that how a entity of a entity set is Related to the entity of other Entity Set. In a E-R diagram it is represented by a diamond.

Every Relationship will have a Unique name
Every Relation will have a degree which means number of Entity sets participating

minimum degree posible is => 1 [which is called Unary Relationship].
n-ary Relationship. And you can have n- number a associate sets.

NOTE :- Practically Binary Relationship are preferable.

→ Coordinalities Ratio :- Coordinalities ratio are studied on Binary Relationship where every element of the first Entity set is related to atleast one Entity of the other Entity set and Vice versa.

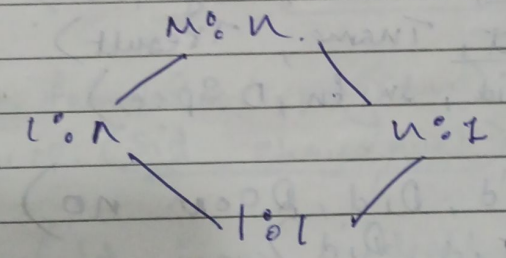
Ex :- Person to adhar Card.

1 to many :- A Relationship is said to 1 to many if every entity of the 1st Entity set can Relate

to any number of Entity of the other Entity set but every Entity of the second Entity set can Relate to atmost 1 Entity of the 1st Entity set.

→ Many to many :—
It is said to be many to many if any Entity of the first Entity set many Relate to any number of Entities of other Entities set and Vice-Versa.
For Ex :- patient to Doctor, student to Teach

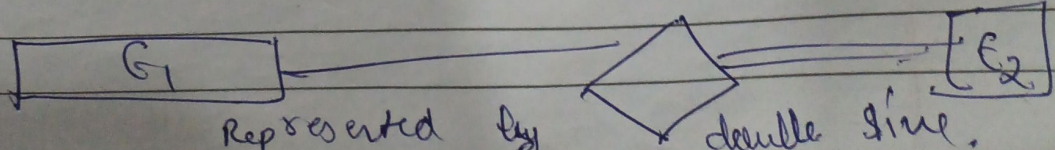
Every 1:1 to



Minimum Coordinality :- For a Entity Set is the minimum degree Possible for of any of its Entity.

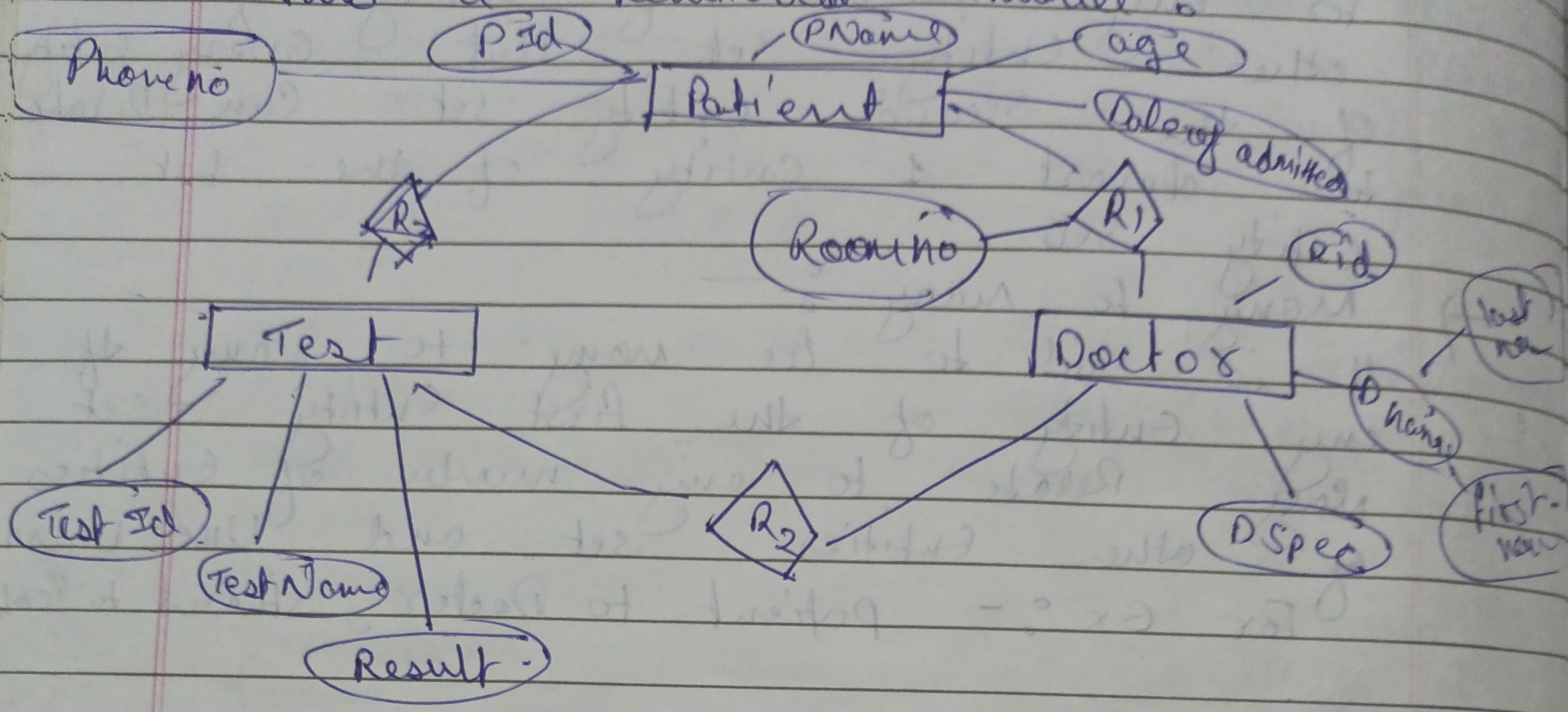
Maximum Coordi :- It is the maximum degree of any Entity for a Entity Set.

→ If minimum Coordinality is 1 or more for a Entity set then it means it has a total participation.



Represented by double line.

Q Consider a E-R diagram and Convert it into a Relational model.

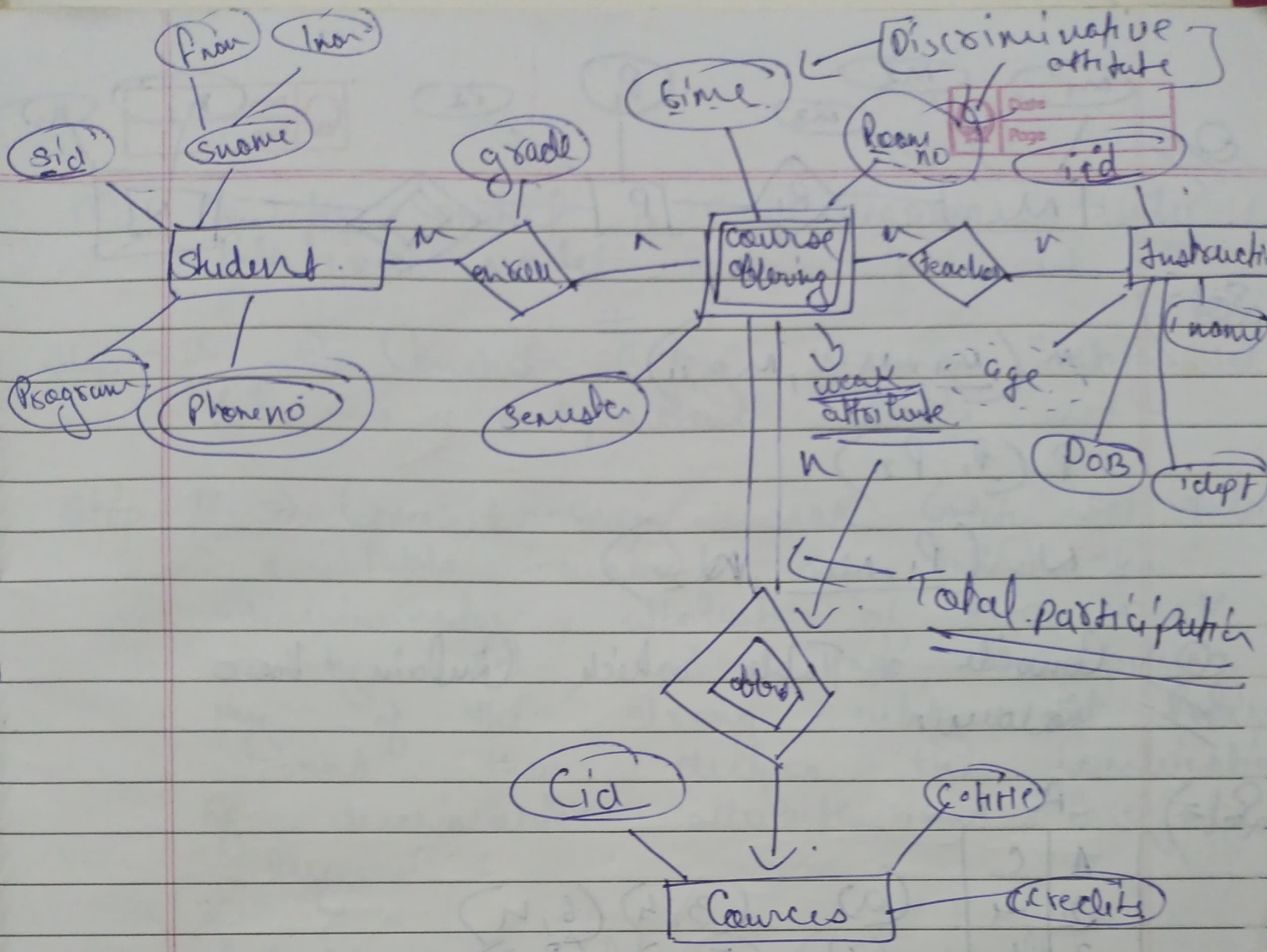


Patient (Pid, Pname, Page, Pda)
Test (Tid, Tname, Tresult)
Doctor (D.id, In, Fn, D Spec)

R_1 (P id, D id, Room no)
R_2 (T id, D id)
R_3 (P id, T id)

Phone no. (P id, Phone no.)

Q Consider a E-R diagram for a University



Student (Sid, fnam, lastnam, program)

student phone no. (Sid, phone numb)

Instructor (id, iname, idept, DOB, age)

Cource (Cid, Ctitle, Credits)

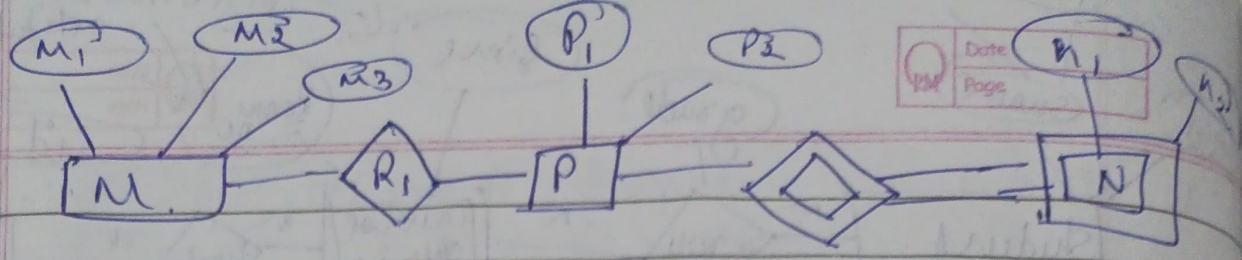
Course offering (time, RoomNo, Cid, semester)

Enroll (Sid, time, RoomNo, Cid, grades)

Teache (Ins id, time, RoomNo, Cid, Inam)

Q

Q//



Sol=)

M (M1, M2, M3, P1)

P (P1, P2)

N. (P1, N1, N2)

Q// Consider a Table which Contain two Columns.

Sol=)

R

A	C
2	4
3	4
4	3
5	2
7	2
9	5
6	4

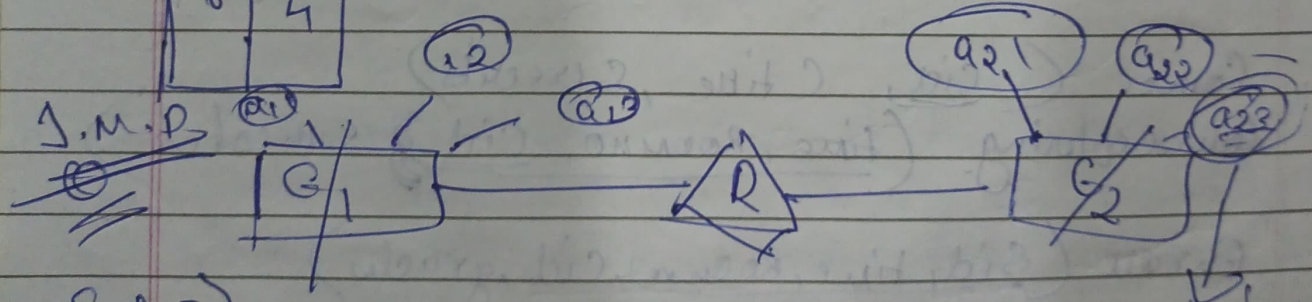
(a) (3,4) (6,4)

(b) (5,2) (7,2)

(c) (5,2) (7,2) (9,5)

(d) none

J.M.P



Sol=)

G1 (a1, a12, a13)

G2 (a21, a22, a23, a11)

G2, a23 (a21, a23)

Q. How to Convert a E-R diagram into Relation model ?

Step - I . ⇒ Convert every strong Entity set into a independent table.

Step - II ⇒ Convert every weak Entity set into a table where we take the discriminator attribute of the weak entity set and take the primary key of the strong entity set from and then declare the Combination of discriminator attribute and foreign key.

If degree is binary (Coordinalties is many to many) or degree is 3 or more then will make a new table for relationship.

Take primary key of all participating Entity set as a foreign key and declare there Combination as ~~foreign~~ primary key. If Relationship have some attributes then they will also become part of the table.

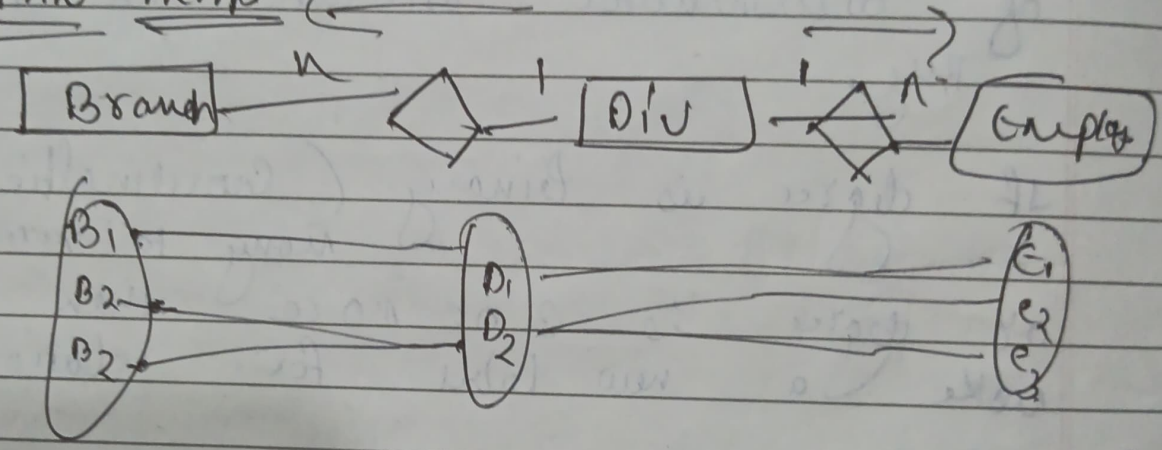
Step - III for binary 1:1 no need of a new table Can modify any side by taking primary key of other side as foreign key.

NOTE : priority must be given to the side having total participation

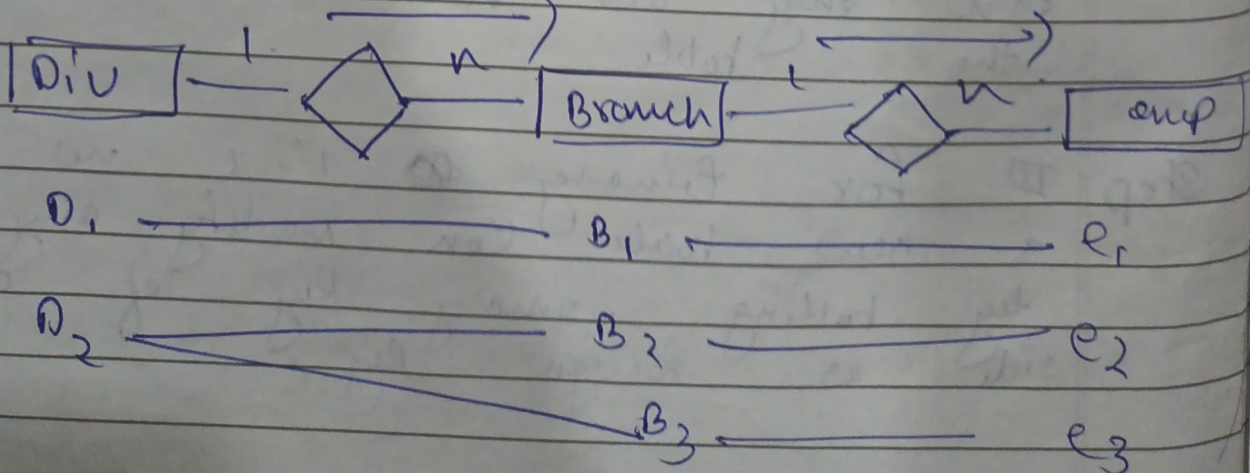
1:n or n:1 :-
here always modify many side.

For every multivalued attribute will make a new table where will take primary key of main table as foreign key and declare foreign key and multivalued attribute.

FAN TRAP



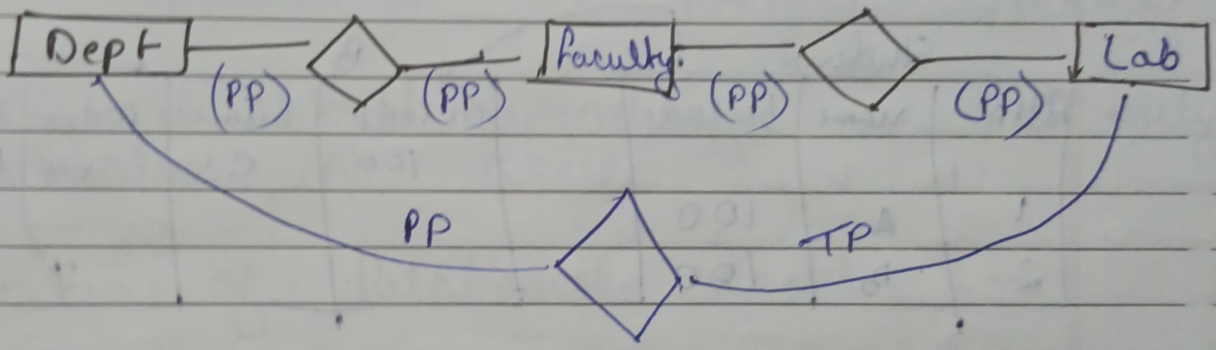
D
↓
B
↓
E



1. when more than one to many Relation goes in opposite direction then this is called fan trap.

2. we can Resolve fan trap By making the direction same

⇒ CHASM TRAP :-



Some time it is possible that two Entity set are related through a third Entity set by Partial Participation even if they are having total participation.

Solution :- we must take a new relationship and connect those Entity sets directly

Prime attribute :-

Normalisation

student

Roll	name	branch	b.name	b Hod name	b Ph.
1	A	100	CS	XYZ	123
2	B	100	CS	XYZ	123
3	C	100	CS	XYZ	123

Student

Rollno	name	br. Code
1	A	100
2	B	100

B.

Br. Code	b name	Hod name	Ph
100	CS	XYZ	123

① Some times it is possible that a single table may Contains a lot of information which leads to a number of problem because we have to repeat the infor for other information.

② Insertion problems :- here with every student we have to repeat the Branch information we Cannot insert the infor of Branch if we do not have a student.

③ modification problem or anal :- Here if we want to change a single piece of infor like Hod - name then it must be done in multiple location.

4. Deletion problem :- Here if we want to delete student information then it will also unintentionally delete branch information.

Conclusion :- Every Table must have a single idea.

5. The method by which we divide table appropriately is called normalisation and the tool used for normalisation is functional dependency.

6. Using functional dependency we can only normalise till BCNF.

* Basics of Relational model :-

① Every Collumn in a table will have a unique name.

② order of Rows and Collumn are insignificant

③ Every cell will Contain single or atomic value.

④ Every table will have a primary key.

NOTE:- The value of primary key cannot be null.

Functional dependency :- Let us Consider a Relational schema R and let x and y be two set of attribute. If there is a functional dependencies if $T_1[x] = T_2[x]$ then T_1 of y should also be equal to $T_2[y]$

$T_1[y] = T_2[y]$.

Here x is the determinant and y is the dependent or x determines y.

Q Consider a Table.

A	B	C	D	G
a	2	3	4	5
2	a	3	4	5
a	2	3	6	5
a	2	3	6	6

(a) $A \longrightarrow BC$

(b) $DE \longrightarrow C$

(c) $C \longrightarrow DG$

(d) $BC \longrightarrow A$

* How to find whether a dependency is valid or invalid ?

→ Step I => $\alpha \longrightarrow \beta$

Check if all value of α are distinct if yes then valid.

Step II =>

Check if all values of β are same if yes then also valid otherwise find atleast 1 value of α on which we are having different values of β.

Q

X	Y	Z
1	4	2
1	5	3
1	6	3
3	2	2

(a) $XY \rightarrow Z$ & $Z \rightarrow Y$
(b) $XZ \rightarrow X$ & $Y \rightarrow Z$
(c) $YZ \rightarrow X$ & $X \rightarrow Z$
(d) $XZ \rightarrow Y$ & $Y \rightarrow X$

Q

A	B	C
1	1	1
1	1	0
2	3	2
2	3	2

(a) $A \rightarrow B$ & $B \rightarrow C$
(b) $A \rightarrow B$ & $B \rightarrow C$
(c) $B \rightarrow C$
(d) $A \rightarrow B$ & $B \rightarrow C$

A functional dependency from α to β is said to be Trivial if β is a subset of α.

① Some functional dependencies are directly visible in Relational Model But some other set of dependency also holds goods which are not directly visible.

② Entire set of functional dependency is called as closure or complete set.

③ Therefore we must know how to calculate closure set of functional dependency before normalisation.

$R(A\ B\ C)$

F_1	F_2	$F_1 + F_2 = F^+$
$A \rightarrow B$, $B \rightarrow C$	$A \rightarrow C$	$(A^+) = ABC$

RAT Rule/RATarion/Armstrong Rule.

6 Inference Rules to Calculate closure Set.

① IR_1 (Reflexivity rule)

If B is subset of ~~alpha~~ α then α tends to B hold good.

If $B \subseteq \alpha$

$\alpha \longrightarrow \beta$.

A

② IR_2 (Argumentation rule) If $\alpha \longrightarrow \beta$

$\therefore \alpha\gamma \longrightarrow \beta\gamma$

③ IR_3 (Transitive Rule) If $\alpha \longrightarrow B$
$\beta \longrightarrow \gamma$

$\boxed{\alpha \longrightarrow \gamma}$

IR_1, IR_2, IR_3 are functionally Complete known as RAT Rules/RATarions/Armstrong Rule which means only these 3 Rules are sufficient enough to find closure

④ IR_4 - Union Rule :- If $\alpha \longrightarrow \beta$
$\alpha \longrightarrow \gamma$

$\boxed{\alpha \longrightarrow \beta\gamma}$

⑤ IR_4 - Decomposition Rule :-

If $(\alpha \longrightarrow \beta\gamma)$

$\alpha \longrightarrow \beta$

$\alpha \longrightarrow \gamma$

⑥ IR_6 (Pseudo Transitive) If $\alpha \longrightarrow \beta$

$\beta\gamma \longrightarrow \delta$

$\boxed{\alpha\gamma \longrightarrow \delta}$

Q. Consider Some Relational schema and Identify a closure set for each object

R (ABCDEFG)

A → B

BC → DE

AEG → G.

$(AC)^+ = AC$?
ABC
ABCDE

R (ABCDE)

A → BC

CD → E

B → D

E → A

$(B)^+ = B$?
BD

R (ABCDEF)

AB → C

BC → AD

D → E

CF → B

$(AB)^+ = AB$?
ABC
ABCD
ABCDE

R (ABCDEFGH)

A → BC

CD → E

E → C

D → AEH

ABH → BD

DH → BC

BCD → H

$(BCD)^+ = BCD$
BCDE
ABCDEH

Comparision b/w two set of functional depend

Consider a Relational schema which contains (A C D E H).

R (A C D E H)

F	G
A → C	A → CD
AC → D	E → AH
E → AD	
E → H	

(a) $F \subseteq G$
(b) $F \geq G$
(c) $F = G$
(d) $F \neq G$

F (c) G

$(A)^+ \rightarrow ACD$ | $(A)^+ = ACD$
$(AC)^+ \rightarrow ACD$
$(E)^+ \rightarrow ACDEH$ | $(E)^+ \rightarrow EADHC$.

$G \subseteq F$.

R (A B C D E)

P:	Q:
B → CD	B → CDE
AD → E	A → BC
B → A	AD → G

(a) $P \subseteq Q$
(b) $P \geq Q$
(c) $P = Q$
(d) $P \neq Q$

F

$(B)^+ \Rightarrow BCDE$ | $(B)^+ \rightarrow BCDAG$
$(AD)^+ \Rightarrow ADECB$ | $(A)^+ \rightarrow ADE$
$(B)^+ \Rightarrow BCDE$ | $(AD)^+ \rightarrow ADG$

Minimal Cover

Date
Page

* Minimization for a set of functional Dependency

Q Consider a Relation

R(A B C D)

A → B
C → B
D → A B C
A C → D

A → B	(A^+) = A B	A
C → B	$(C)^+$ → C B	C
D → A	$(D)^+$ → D A B C	D
D → B	D^+ → D B	D
D → C	$(D)^+$ → C D	C
A C → D	A C → A C D B	A C

$(AC)^+$ = A C D B
$(A)^+$ = A B
$(C)^+$ = C B

A → B
C → B
D → A C
A C → D.

— X —

Q R(W X Y Z)

X → W
W Z → X Y
Y → W X Z

X → W	$(X)^+$ = X W.	X
W Z → X	$(WZ)^+$ = X W Z Y	X W Z Y / Z
W Z → Y		
Y → W	$(Y)^+$ → Y W.	Y
Y → X	$(Y)^+$ = X Y W	X Y W Z.
Y → Z	$(Y)^+$ = Y Z X W.	

$(X)^+$ = X W
$(WZ)^+$ = W Z X Y
$(WZ)^+$ = W Z Y X

X → W
W Z → Y
Y → X
Y → Z

Q How to find Candidate Keys in a Relational Schema are :-

Q Find the no. of Candidate Key in each Schema.

① R(ABCDEFGHIJ)

AB → C

A → DE

B → F

F → GH

D → IJ

② R(ABCDEFGHIJ)

AB → C

BD → EF

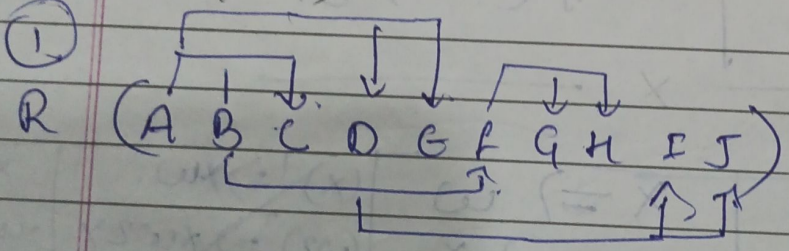
AD → GH

A → I

H → J

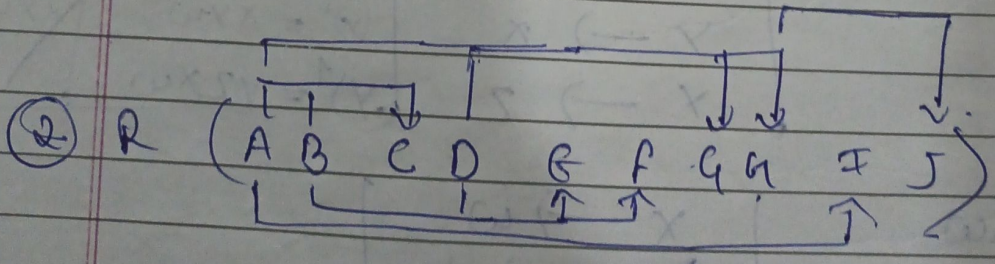
~~(R) R(ABCDE)~~

①



$(AB)^+ = R$ ✓

②



$(ABD)^+ = R$ ✓

2NF

(2). R(ABCPE)

A → B

BC → E

DE → A

1NF

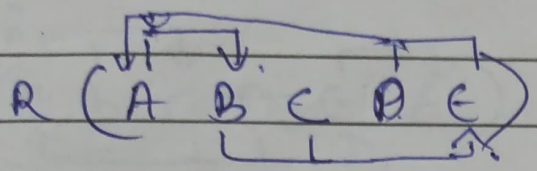
(4) R(ABCDE)

CE → D

D → B

C → A

(3).



(CD)$^+$ = X

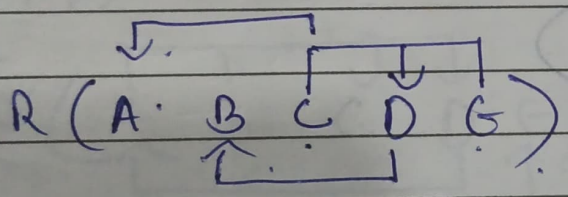
(ACD)$^+$ = R ✓

(BCD)$^+$ = R ✓

(CDE)$^+$ = R ✓

3NF

(4.



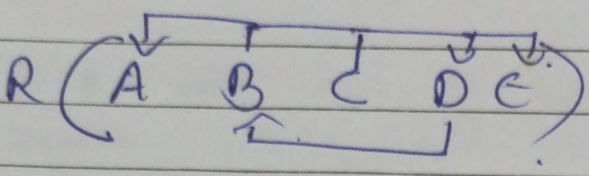
1NF

(CE)$^+$ = R ✓

(5) R (A B C D E)
BC → ADG
D → B.

(6) R (A B C D E)
BD → E
A → C

(5) R (A B C D E)



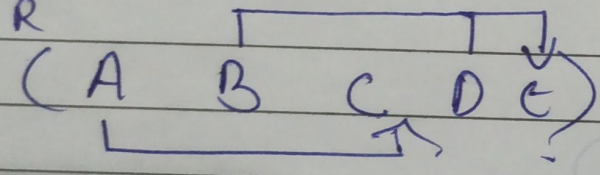
(C)⁺ = X.
AC = X
BC = R —
CD = R —
CE = X

(AE)⁺ = X

BC
CD

3NF

(6) R (A B C D E)



(ABD)⁺ = R —

1NF

~~ABCD~~
~~ABDE~~

(7) R(ABCDEF)

AB → C

DC → AE

E → F.

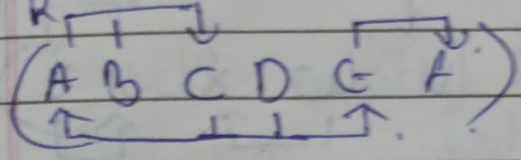
(8) R(ABCDE)

AB → CD

D → A

BC → DE

(7)



(DB)$^+$ = X.

(ABD)$^+$ = ✓

(BCD)$^+$ = ✓

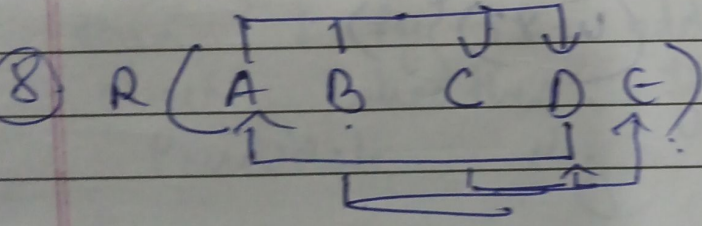
(BDE)$^+$ = X.

(BDF)$^+$ = X

BDEA ⇒ X.

1NF

(8)



3NF

(B)$^+$

AB = R ✓

BC = R ✓

BD = R ✓

BE = X

(9) R(WXYZ)

Z → W

Y → XZ

XW → Y

(9)

R (W X Y Z)

(W)+ ⇒ X

(X)+ ⇒ X

(Y)+ ⇒ R

(Z)+ ⇒ X

(WX)+ = R

(XZ)+ = R

(WZ)+ = X

3NF

(10) R(VWXYZ)

Z → V

Y → Z

X → YV

W ⇒ X

R (V W X Y Z)

(W)+ = X

(VW)+ ⇒

(WX)+ ⇒ R

(WY)+ ⇒ R

(WZ)+ ⇒

(WYZ)+ ⇒

2NF

(11) R(ABCDEF)

ABC → D

ABD → G

CD → F

CDF → B

BF → D

R (A B C D G F)

(AC)+

1NF

How to find a Candidate Key in Relational Model

Step - I

Identify those attributes which do not have a incoming edge known as essential attribute.

check whether Essential attribute are Candidate Key or not If yes then it means only 1 Candidate Key is there.

Step => II

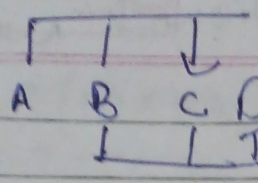
If there is no Essential attribute or Essential attribute are not Candidate Key then I try I add 1 attribute and check whether If the combination is Candidate Key or not.

If a set is Candidate Key then no need to check whether its Super set is a candidate key or not.

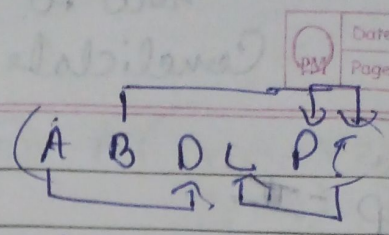
Q Consider following schema and Identify Candidate Key for each of them.

(1) R(ABCD)
AB → C
BC → D

(2) R(ABDLPT)
B → PT
T → L
A → D



① $(AB)^+ = R$.



$(AB)^+ = R$.

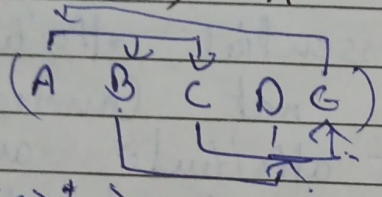
② R(ABCDE)

$A \rightarrow BC$

$CD \rightarrow E$

$B \rightarrow D$

$E \rightarrow A$



Key.

$(A)^+ \Rightarrow R$ — $(BC)^+ \Rightarrow R$ ✓

$(B)^+ \Rightarrow X$ — $(CD) \Rightarrow R$ ✓

$(C)^+ \Rightarrow X$ — $(BD) \Rightarrow X$

$(D)^+ \Rightarrow X$

$(E)^+ \Rightarrow$ ✓

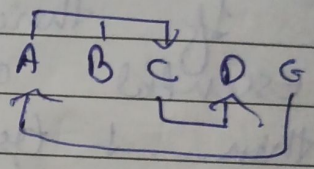
④ R(ABCDE)

$AB \rightarrow C$

$C \rightarrow D$

$D \rightarrow C$

$E \rightarrow A$



⑤ R(ABCD)

$AB \rightarrow CD$

$D \rightarrow A$

$\Rightarrow$

AB

BC

BD

⑥ R(ABCDEG)

$AB \rightarrow C$

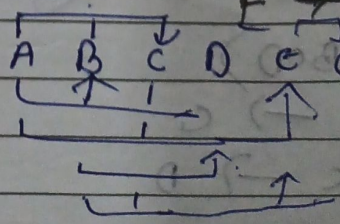
$AC \rightarrow B$

$AD \rightarrow E$

$B \rightarrow D$

$BC \rightarrow E$

$E \rightarrow G$

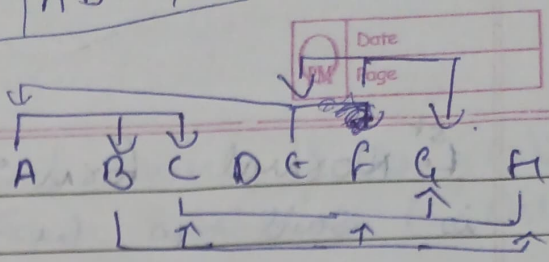


③ AB

AC

BC

AB → A Trivial



2013 (2)

R(ABCDEFGH)

CH → G
A → BC
B → CFH
E → A
F → EG

(AD) ⇒ R
(BD) ⇒ R
(CD) ⇒ X
(ED) ⇒ R
(FD) ⇒
(GD) ⇒
(HD) ⇒

(a) = 3
(b) ⇒ 4
(c) ⇒ 5
(d) 6

2015 R(PQRSTU)

2016

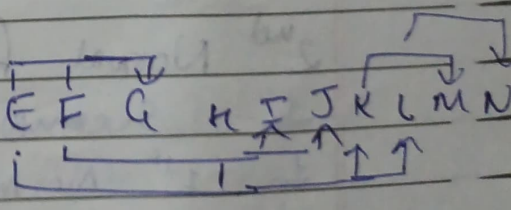
R(VWXYZ)

BK → UY

(a) VXYZ
(b) VWXZ
(c) VWXY
(d) VWXYZ

2014

R(EFGHIJKLMN)



EF → G
F → IJ
EH → KL
K → M
L → N

(a) EF
(b) EFH
(c) EFHKL
(d) E

EFH

cell

Date
Page

1st Normal form :- A relational schema R is said to be in first normal form if every cell in the table contains atomic value.

It means

1. 1st Normal form do not allow composite or multivalued attributes.

2. A Table in 1st Normal form will always have exactly 1 primary key. (Null value is not allowed in P.K.)

Student.

Roll no	Name	Course
1	A	OS, DBMS
2	B	CN, CA
3	C	DSA

→

Student

Rollno	Name	Course
1	A	OS
1	A	DBMS
2	B	CN
2	B	CA
3	C	DSA

NOTE :- It is always assumed that if we have a proper Relational table then it will always be in first normal form.

2nd Normal form :- A Table is said to be in Second Normal form if 1st Case :- It is in 1st Normal form.
2nd Case :- There must not exist any partial dependency.

Q) what is partial dependency ?

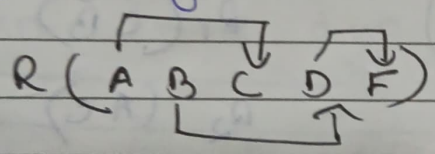
A) If a non prime attribute instead of depending on the Entire Candidate key

depends on its subset then. It is called partial dependency. [From NON prime to NON Prime]

Prime :- A attribute is called prime attribute if it is the part of any Candidate Key.

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_3	c_3
a_4	b_3	c_3
a_5	b_3	c_3

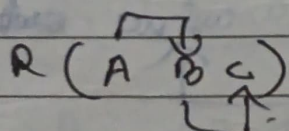
1.) If a table is not in 2nd Normal form then Construct 1st Table for the Candidate Key. only those attributes will come which are totally dependent on key and then for every partial dependency we can have a seperate table.



R_1 (AB)
R_2 (AC)
R_3 (BD F)

* 3rd Normal form :- A Relational table is said to be in 3rd Normal form *.

1. It is in 1st Normal form.
2. There exist No Transitive dependency.



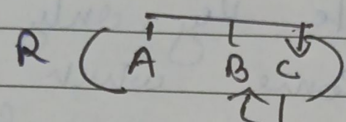
B ⟶ C

Here A is prime attribute and B and C are NON-prime attribute Here dependency from B to C is transitive as 1 - NON prime is finding other NON-prime. Here if B get a Null Value in Runtime then we cannot determine C.

(2) P.D. => (Prime) P ⟶ N.P. (NON Prime)
(3) T.D. => N.P. ⟶ N.P.

A Relational schema are said to be in 3rd Normal form :- If from every dependency from $\alpha \to \beta$ either α is a super key or β is a Prime attribute.

BCNF :- a relational table is said to be in BCNF if for every dependency a from $\alpha \to \beta$, α must be super key



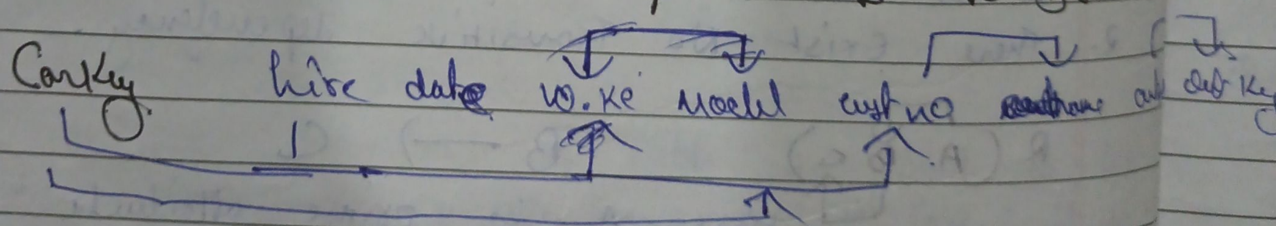
Keys :- AB, AC

AB → C
C → B

R_1 (C B)
R_2 (A C)

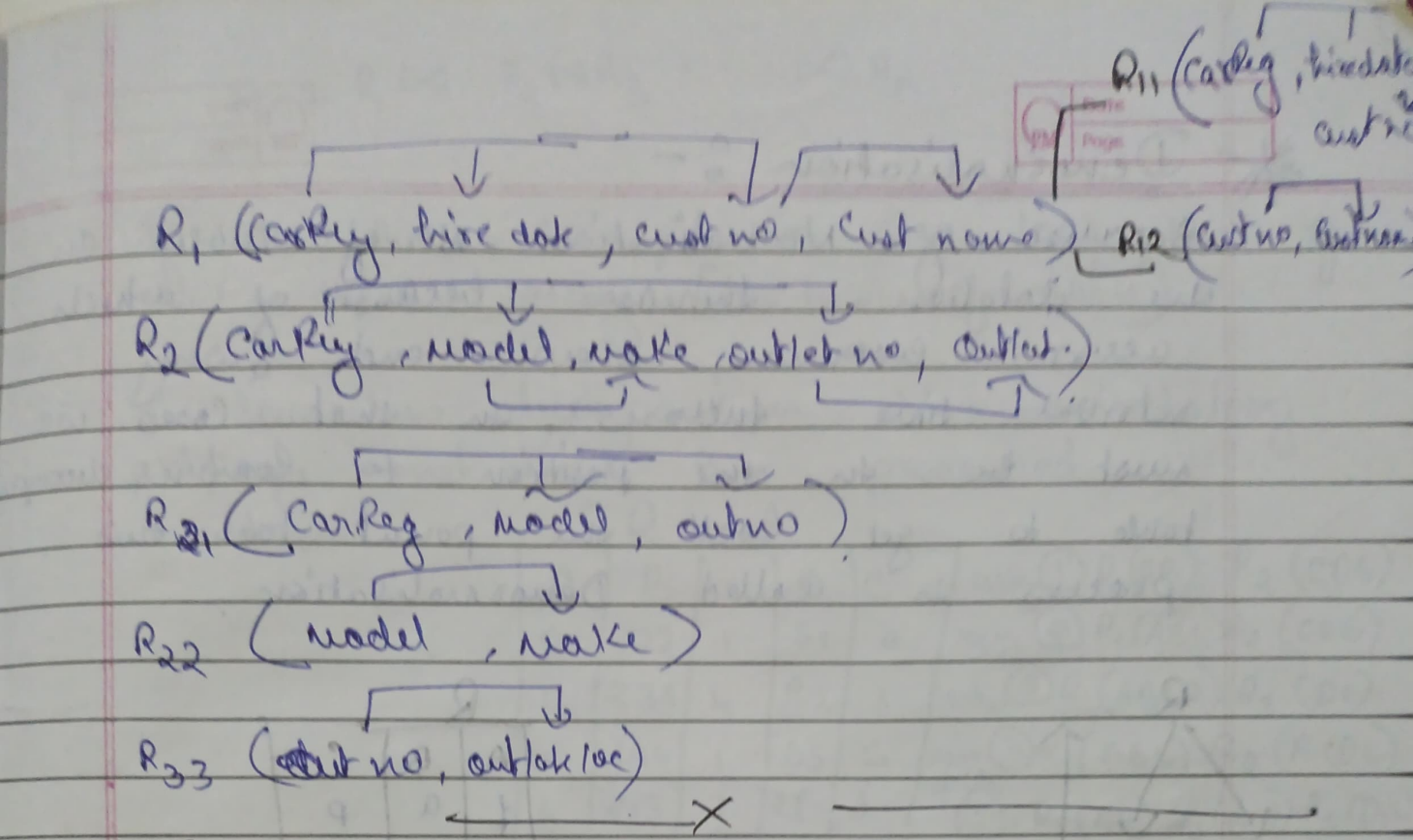
———— X ————

Q/ Consider a Relation Schema of a taxi Company. Identify the Candidate key and Normal form. If Not in BCNF than Decompose it to BCNF.

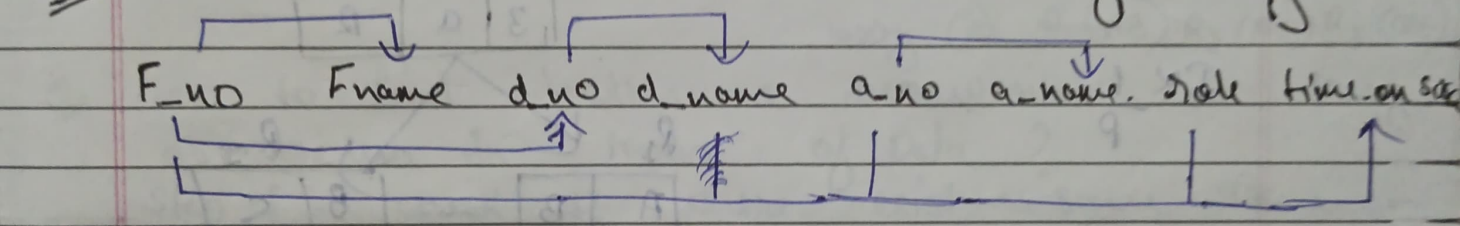


A B C D G F G H I

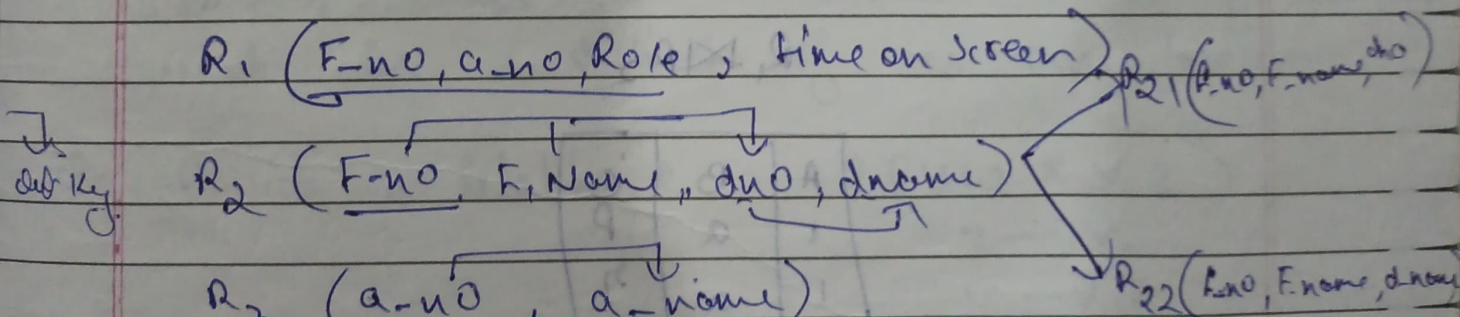
ABC DG F GH I



Q Consider a Relational schema of Bollywood



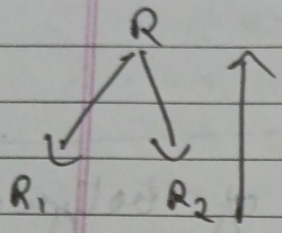
(F_no, a_no, Role) = R — (Candidate Key)



Maximum keys in each leaf node :-
Block Pointer + x (Key Size) + x (Record pointer) < Block

* Denormalisation :-

Because of Normalisation No. of Tables in the database increases because of which access becomes slow and query retrival time increases. In that cases we must be in the position to combine decomp table to get back the parent table this process is called Denormalisation



$I \propto \frac{1}{P}$

R

A	B	C
1	a	p
2	b	q
3	a	R

R_1

A	B
1	a
2	b
3	a

R_2

B	C
a	p
b	q
a	R

$R_1 \bowtie R_2$

A	B	C
1	a	p
1	a	R
2	b	q
3	a	p
3	a	R

If a Table R is decomposed into n number of Tables then after Taking natural join From R_1 to R_n we must get back the same Table

$$R = R_1 \bowtie R_2 \bowtie R_3 \cdots \bowtie R_n$$

If yes then decomposition is lossless, If Not then lossy. [In terms of data]

a lossless decomposition is a mandatory property and must be preserved.

R

A	B	C	D	E
a	122	1	S_1	a
e	236	4	e_2	6
a	199	1	65	G
b	213	2	28	d

lossy (1) $R_1(AB)\ R_2(CDE)$

lossy (2) $R_1(ABC)\ R_2(CDE)$

lossless (3) $R_1(ABCD)\ R_3(DE)$

lossy lossless (4) $R_1(ABC)\ R_2(ACDE)$

(5) $R_1(AB)\ R_2(BCD)\ R_3(DE)$

lossy (6) $R_1(AC)\ , R_2(DE)$

lossy (7) $R_1(AB)\ R_2(AC)\ R_3(CDE)$

Q// How to find whether a decomposition is lossy or lossless in terms of Data ?

Step 1°:- check if there exist some common attribute or not, if not then lossy, if yes go to step 2.

Step 2°:- This common attribute or set of attribute must be distinct, if yes then lossless.

Q// Consider a Relation R :- R(V W X Y Z)

$Z \rightarrow Y$

$Y \rightarrow Z$

$X \rightarrow YV$

$VW \rightarrow X$

$R_1(VWX)\ \ R_2(XYZ)$

$R_1(V\ W\ X)$ — 1

VW

XW

=

$R_2(X\ Y\ Z)$ — 1

X

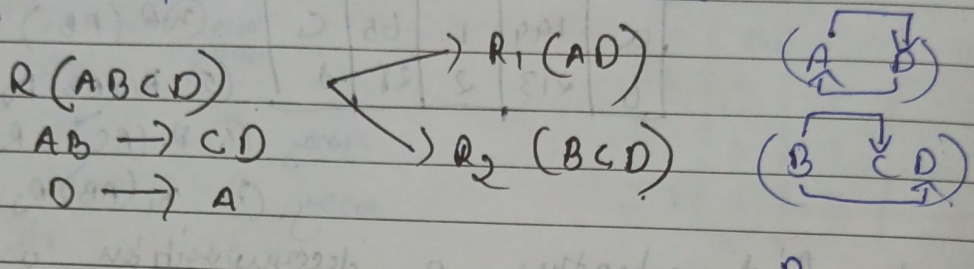
=

how to find decomposition lossy and lossless using functional dependency?

Step no. 1:-

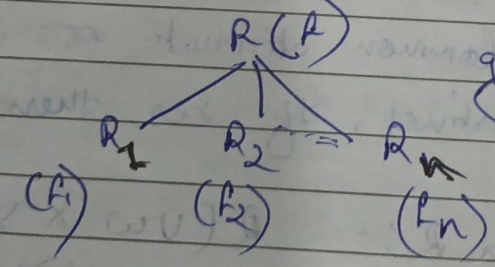
Find a common attribute if not then lossy if yes then go to step no. 2.

Common attribute must be a candidate key either in Table 1 or Table 2 or both. If yes then decomposition is lossless.



B
A

Dependency preserving decomposition:- If a table R is decomposed into $R_1, R_2 \ldots R_n$ such that $F^+ = (F_1 \cup F_2 \ldots \cup F_n)^+$ If this equation holds good then decomposition is dependency preserving otherwise not.



a decomposition should be dependency preserving is a optional property. It is nice to have but not necessary.

R(ABC)

A → B

B → C

C → A

$A^+ = ABC$	$R_1(AB)$	$R_2(BC)$
$B^+ = BCA$	A → B	B → C
$C^+ = CAB$	B → A	C → B
		$C^+ = CBA$

Consider a following schemas and identify which of them are dependency preserving as well as lossless.

① R(A B C D)

A → B
B → C
C → D
D → A

(A)⁺ = ABCD
(B)⁺ = BCDA
(C)⁺ = CDAB
(D)⁺ = ABCD

R1(AB)	(BC)	CD
A → B ✓	B → C ✓	C → D ✓
B → A ✓	C → B ✓	D → C ✓

ABC

lossless, as well as Dependency preserving

② R(ABCDEG)

AB → C
AC → B
AD → E
B → D
BC → A
B → G

R1(AB)	R2(BC)	R3(ABDE)	R4(EG)
A → B ✗	B → C ✗	A → B ✗	E → G ✓
B → A ✗	C → B ✗	B → A ✗	G → E ✗
		B → D ✓	
		D → B ✗	
		E → D ✓	
		D → E ✗	
		A → BDE ✗	
		B → ADE ✓	
		D → ABG ✗	
		E → ABD ✗	
		B → D ✓	
		AB → DC ✓	
		AD → E ✓	
		BE → D ✗	
		ABD → E ✗	
		ADE ✓	
		BDE ✓	

A⁺ ⇒ A
B⁺ ⇒ BDG
C⁺ ⇒ C
D ⇒ D
AB ⇒ ABCDEG
AC ⇒ ABCDGED
AD ⇒ ADE
BC ⇒ ABCDGE

③ R(ABCDE)

AB → C

C → D

D → E

E → A

$R_1(BC)\ R_2(CD)\ R_3(EA)\ R_4(DE)$

B C D

E A D

Lossless

But not D.P.

$R_1(BC)$	$R_2(CD)$	(EA)	(DE)
✗ B → C	C → D	E → A	D → E
✗ C → B	✗ D → C	A → E ✗	E → D ✗

$(AB)^+ = AB$

④ : R(PQRS)

PQ → RS

S → P

(L-L/DP)

$R_1(PS)$

P → S

S → P

$R_2(QRS)$

✱ Multivalued functional dependency :-

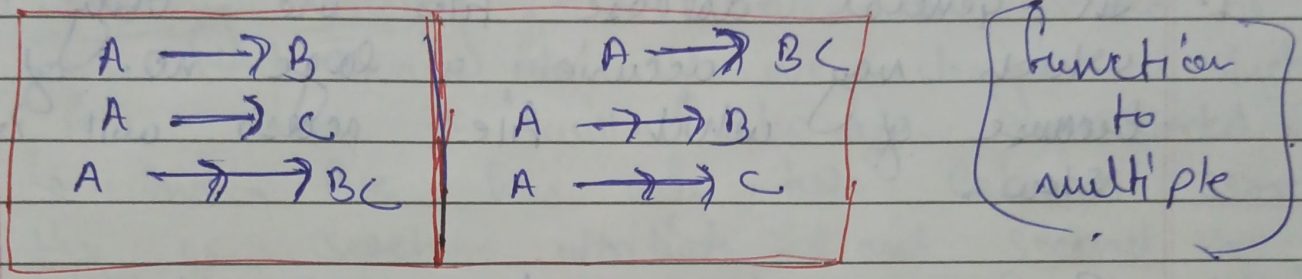
A multivalued function dependency is denoted like this

A →→ B

A →→ B

which means or a single value of A is possible to have multiple value of B.

2. If A functionally determines B then it also multivaluedly determines B But vice-versa is not True



Lcid	P.id	Add
L1	P-1	A-1
L1	P-1	A-2
L2	P-2	A-3
L3	P-2	A-3

If there exist more than 1 multivalued dependency in a table which are not related to each other like a person can have different loans and Addresses which are not Related to each othe therefore 1 information will be repeated for other.

Conclusion :- A table is said to be in forth normal form 1st if it is in BCNF and 2nd there Exist No-multivalued function dependency.

Date 10/2/18.

* A Draw Back of Normalisation is :-

No. of table increases in database and for a single query sometimes 4-5 Tables are required which make system slow.

1. In General database files are very large which may accurie a large no. of tables Because of which file access will be slow.

2. There are two method of organising a file in database

1. Sorted file :- Here the records of a file are sorted according to some Colomn or field. [It means file is still unsorted with respect to other fields or Colomn]

Advantage :- Access is very fast as it supports binary search disadvantage maintainence is costly as insertion and deletion Records a number of Swap.

2- Unsorted file organisation :- Here the Records are order Randomly or Generally in the order in which they are inserted

Advantage :- Maintainence is easy.
Dis Advantage :- access is slow as it requires liner search

UNIT => 3

* Indexing

1. Indexing is a secondary or alternative method to access the file in an efficient manner.
2. Indexing can be classified either on sorted or unsorted or single level or multilevel index or sparse or dense indexing.

General

Primary Indexing :- 1) Either for sorted or unsorted main file. If we design a index file then it will always be sorted.
2) A index file has only two column search key or search attribute and second is pointer [Either Block pointer or Record pointer]

Imp.
1. Main file sorted with respected to primary key.
2. Indexing will be done on primary key.
3. No. of entries in index file is the No. of Blocks acquired by main file.

Q. Suppose there is a database file which contains 30,000 records and the size of each records is 100 Bytes. If memory contains Block's of 1 KB then find the no. of Block's required to store the main file? If we maintain a primary indexing file on this main file such that each record is 15 byte long find

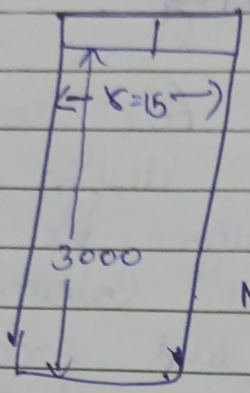
$$30,000 \times 100 \Rightarrow 1024$$

- Blocking factor = $\frac{\text{Block Size}}{\text{record size}}$
- no. of Block = $\frac{\text{total Blocks}}{\text{Blocking factor}}$

1 Block me kitne record aate hai

1) Blocking factor of index file.
2) No. of Blocks occupied by the I.F.
3) How many block access are Required if we access main file through index file.

Ans:)



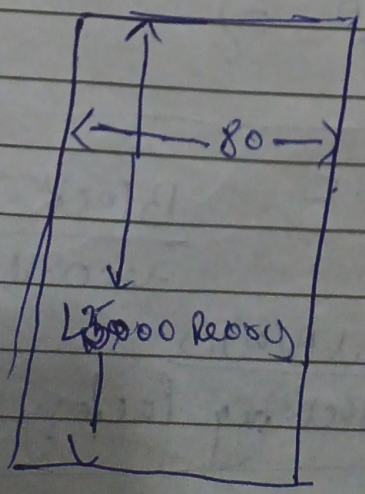
$B.f. = \left\lfloor \frac{1024}{15} \right\rfloor \Rightarrow 68$ → lower Bound

$\text{No. of Blocks} = \left\lceil \frac{3000}{68} \right\rceil$ ← upper Bound $\Rightarrow \lceil 44.11 \rceil$

$\Rightarrow 45$

Block access :- $\lceil \log_2(45) \rceil = 6+1$

Q/ Consider a file which Contain 45,000 records and each records is of 80 Byte If System Block is of 512 byte and Index file Contains 2 Columns Search Key of 12 byte and pointer of 8 byte Find the Improvement of using primary Index file Irrespect to main file in terms of Block access.

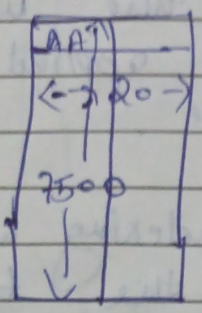


Block size = $\frac{512}{80} = 6$

Blocking factor = 6

No. of Block = $\frac{45000}{6} = 7500$

Block access : $\lceil \log_2(7500) \rceil \Rightarrow 13$



$\Rightarrow \frac{512}{20} \Rightarrow 25$

$\frac{7500}{25} = 300$

$\lceil \log_2 300 \rceil \Rightarrow 9$

$= 10$

Improvement = 3.

Primary indexing is an example of sparse indexing

* Clustered indexing :- In main file sorted, indexing is on Non-key attribute.

2. Avg no. of block access = $\log_2(B_i) + 1$

3. No. of entries in index file is the no. of unique values of the Non-key attribute in the main file.

4. It is a example of sparse as well as dense indexing.

* Sparse indexing :- when all records of the main file do not gets a entry in index file then it is called sparse indexing. [No. of entries in main file ≠ No. of entries in index file]

Ex :- Primary indexing.

* Dense indexing :- when every value of the attribute on which indexing is done get a entry in index file then it is called dense indexing.

Sparse ⇒ Every Record is not given entry.
Dense ⇒ Every value is given entry.

Ex :- Clustered Indexing.

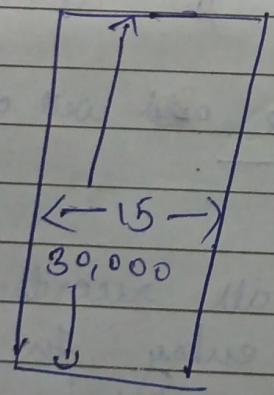
* Secondary Indexing :- 1. here main file is unsorted Can be applied on a Key or NON Key attribute.

2. It is a example of dense Indexing as every record gets a entry in the Index file.

3. No. of entries in Index file is same as of No. of entries in the main file.

Avg Block Access = $\frac{B}{2}$ | For Index File = $\log_2(B_i)+1$

Q On the previous Question if we Consider that a main file is unsorted and we apply a secondary Index with the Record size of 15 Byte find the Block access required using Index file ?



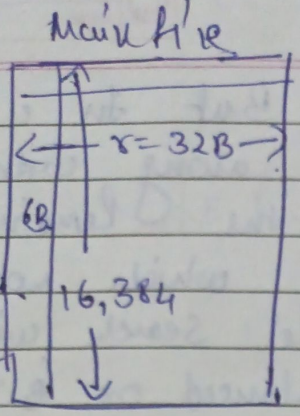
Blocking factor :- $\frac{512}{15} \Rightarrow 68$

No. of Block = $\frac{30,000}{68} \Rightarrow 442$

$q+1 \Rightarrow 10$

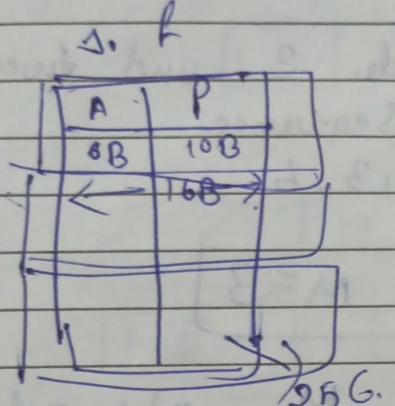
Primary Indexing = 1

Q Consider a file with 16384 Records each Record size length = 32 Byte and Key filed is 6 Byte .. Block size 1024 Byte and Block pointer is 10 Bytes.



Block Factor = $\frac{1024}{16} = 64$

No. of Block = $\frac{16384}{64} = 256$



P.I = $\frac{256}{64} = 4$

	I	II
(a)	8	0
(b)	128	6
(c)	256	4
(d)	512	5

In Case of clustered indexing main file is closly sorted or NON-decreasing.

Physical structure

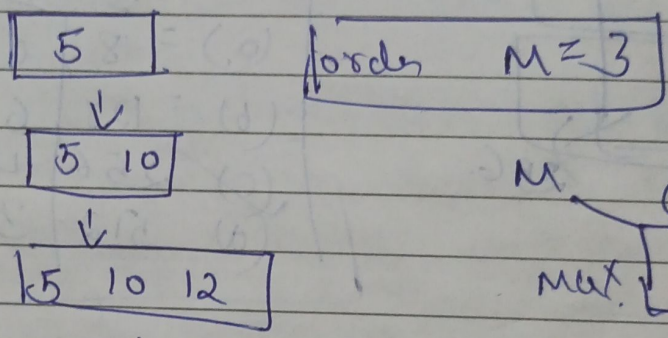
As we understood that index file's are sorted structure then we must have proper datastructres to take advantage of index file

2. array work's well with sorted data but maintain that is insertion and deletion is costly for array.
3. Binary search Tree are good with sorted data as well as they are flexible but worst Case search time Complexity for BST is order of n.
4. AVL Tree is Roughly sorted and give worst

case search time ($\log_2 n$) But in a node they store only a data along with two pointer which will waste the remaining database block. because of which more blocks will be required and search will be slow. solution is a Balanced or B Tree.

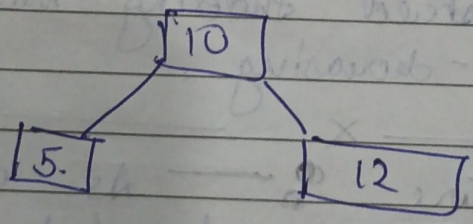
Q Consider a B Tree of order 3 and insert the following data in sequence

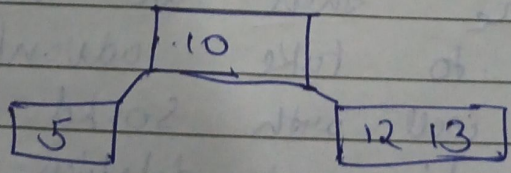
5, 10, 12, 13, 14, 1, 2, 3, 4.

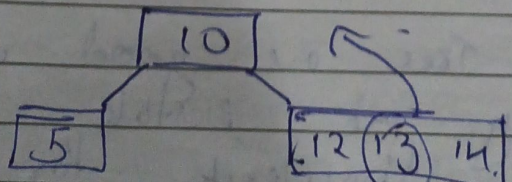


M — Child Data

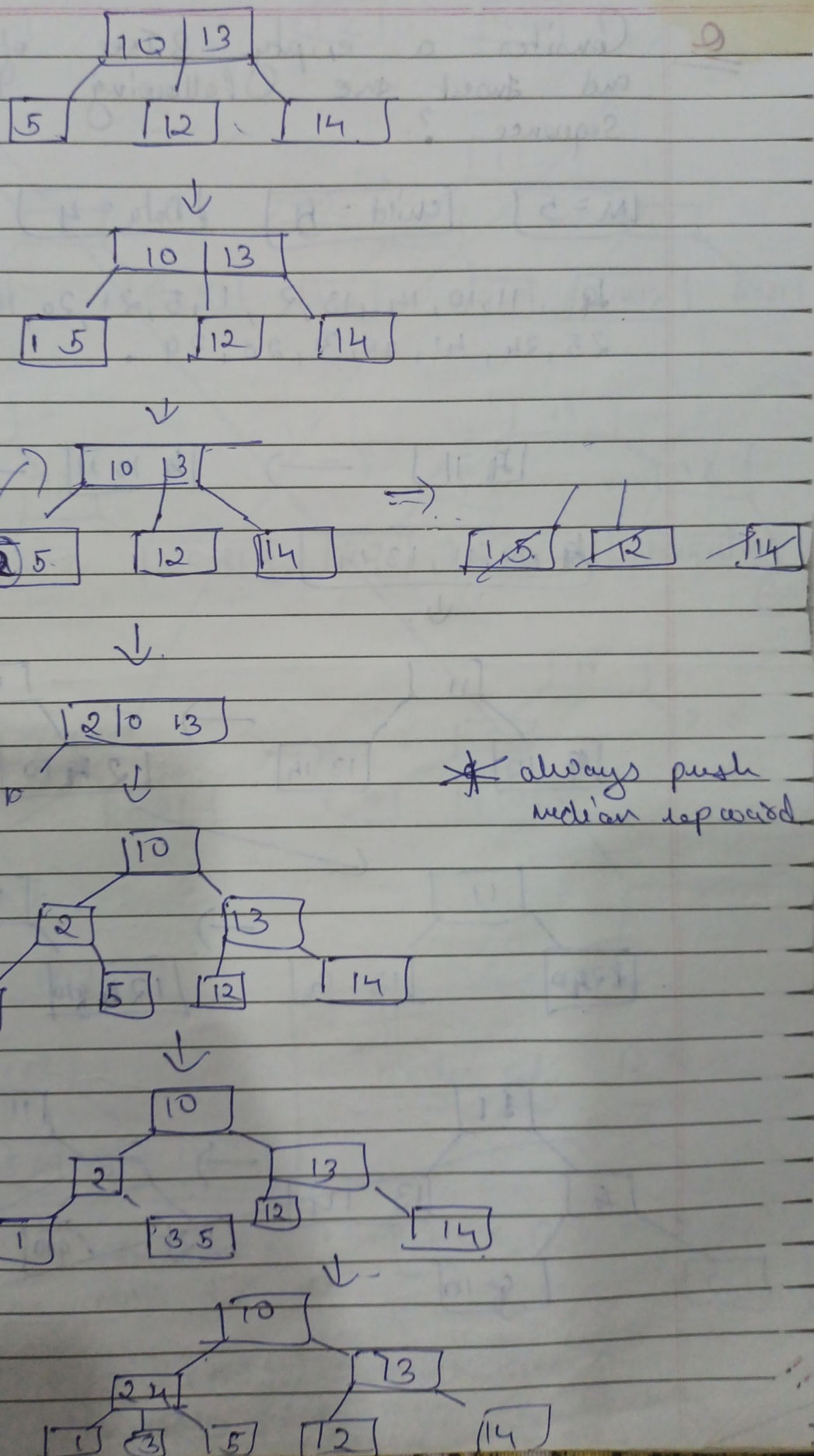
Max | 3 | 2







Push median upward.

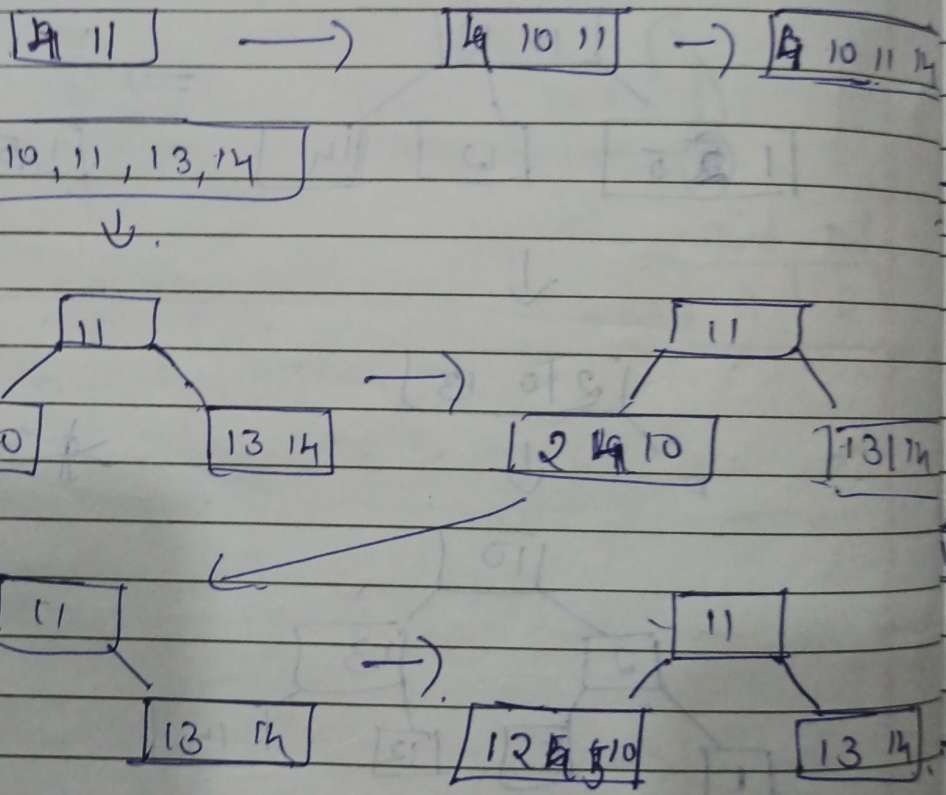

10 13
5 12 14
10 13
15 12 14
10 13
1 2 5 12 14
1 5 12 14
2 10 13
always push median upward
10
2 13
1 5 12 14
10
2 13
1 3 5 12 14
10
2 4 13
1 3 5 12 14

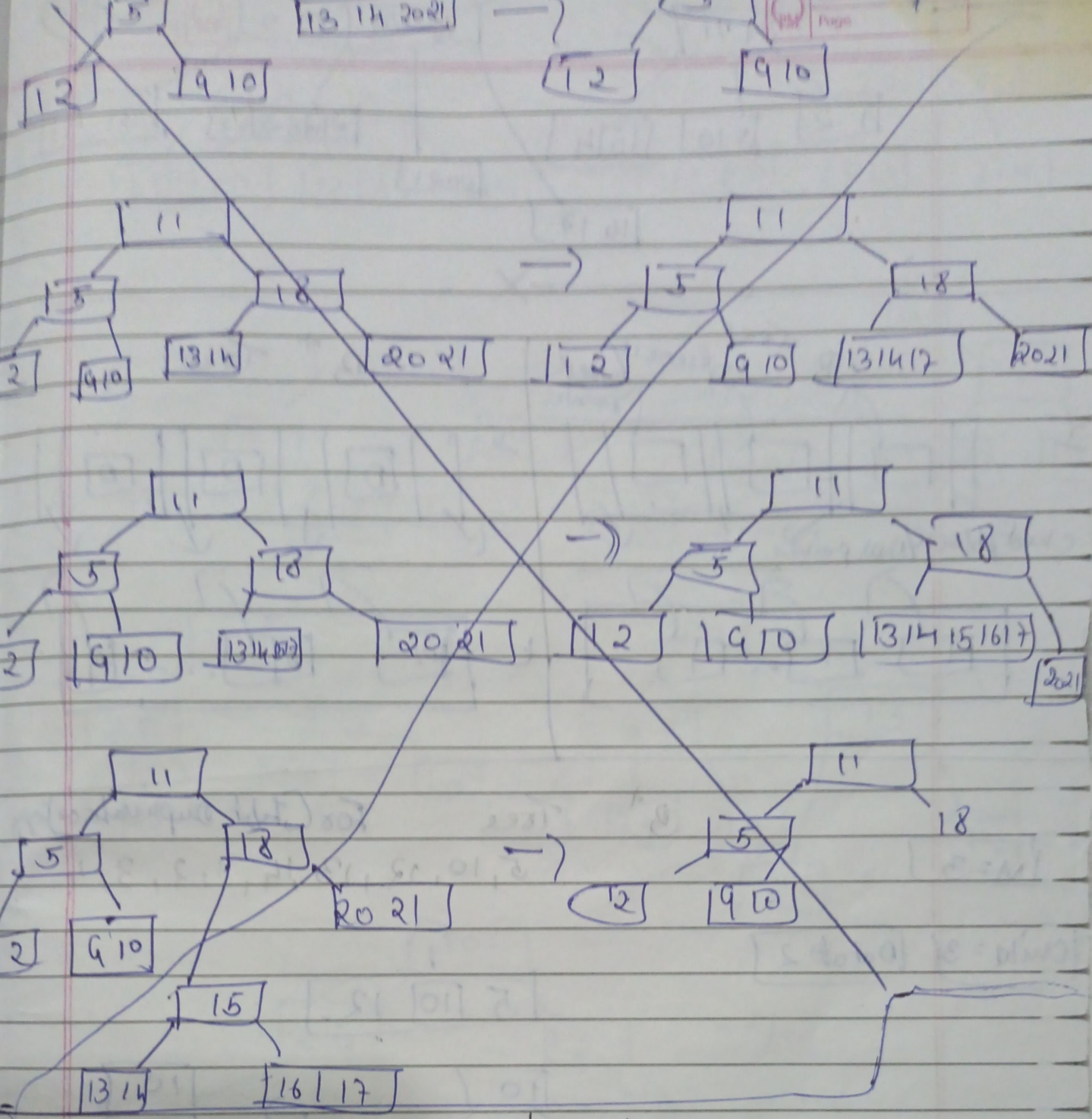
B Tree

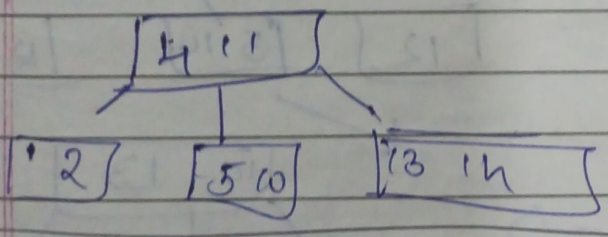
Q. Consider a empty B Tree of order = 5 and insert the following nodes in sequence ?

M = 5 | child = M | Data = M

4, 11, 10, 14, 13, 2, 1, 5, 21, 20, 18, 12, 15, 16, 23, 25, 24, 41, 30, 31, 25, 29.







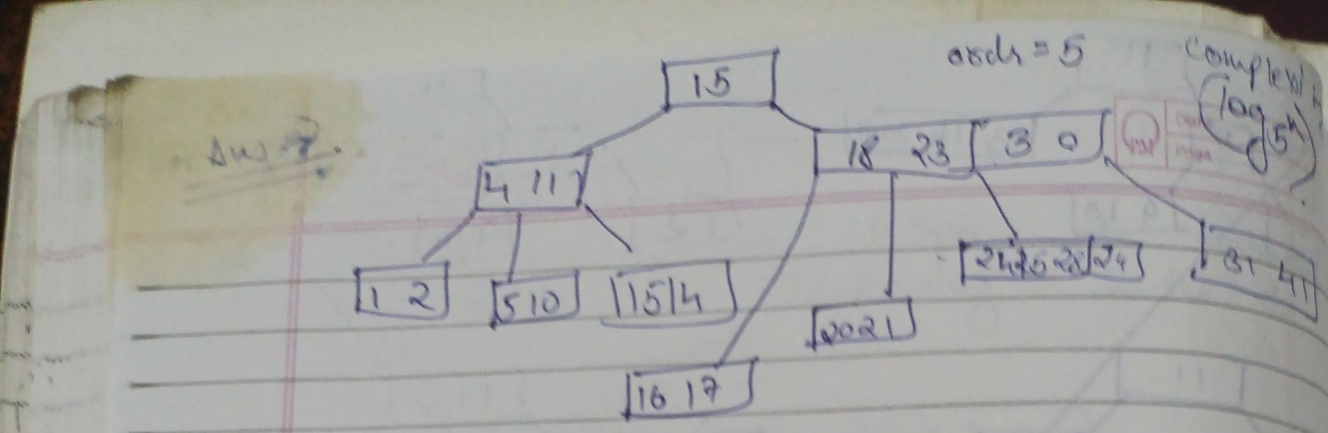
order of B Tree

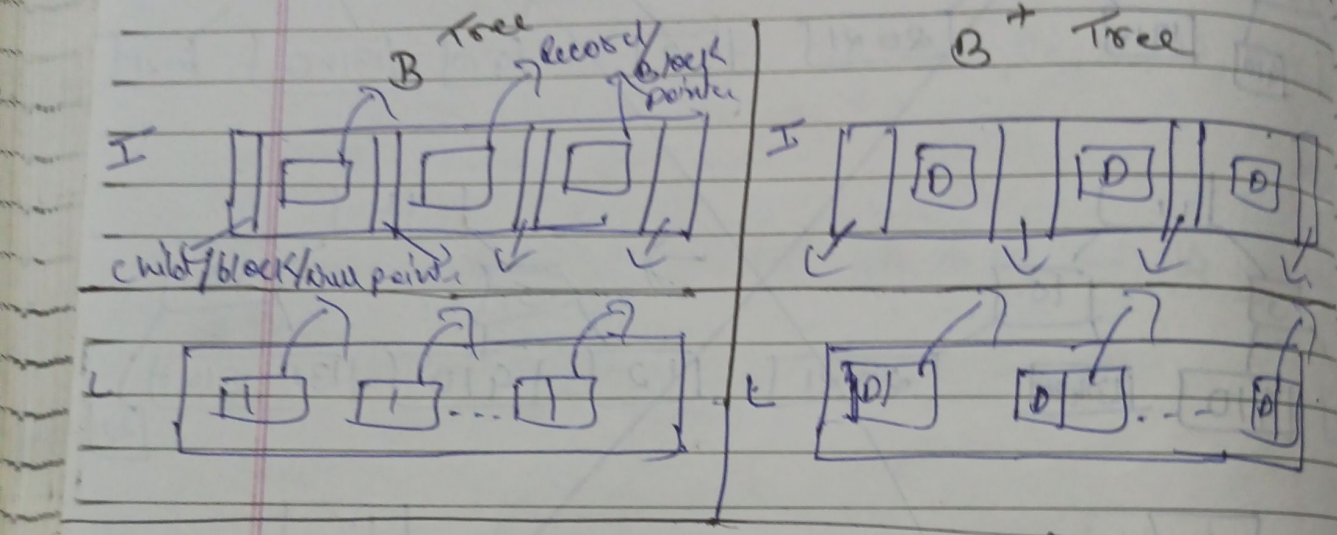
B Tree structure :- $[P<K_1 R_1> P<K_2 R_3> \ldots P]$

Let the order be

$$[O \times P + (O-1)(K + P_R) \leq \text{Block size}]$$

where P : is block pointer size K : search key size
P_R : record pointer size.



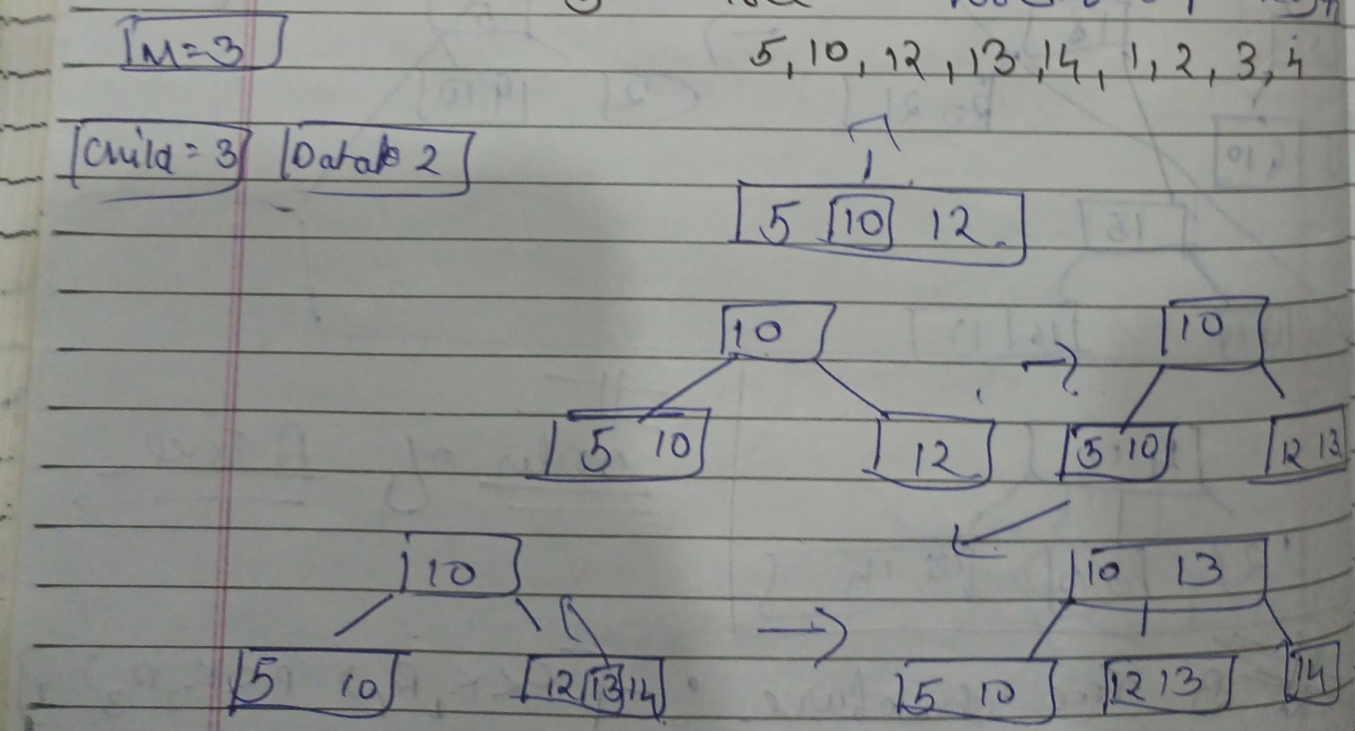


B+ Tree For Eliminate Duplicate entry

M=3

Child = 3 Data = 2

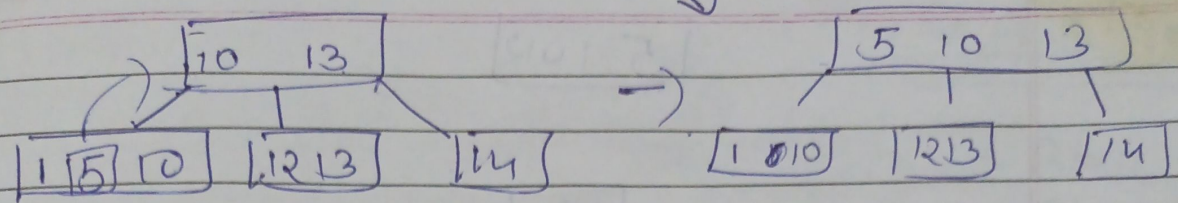
5, 10, 12, 13, 14, 1, 2, 3, 4

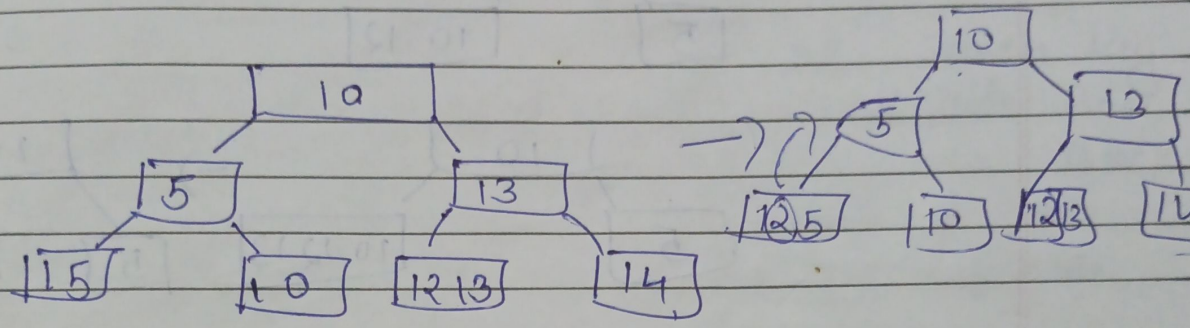


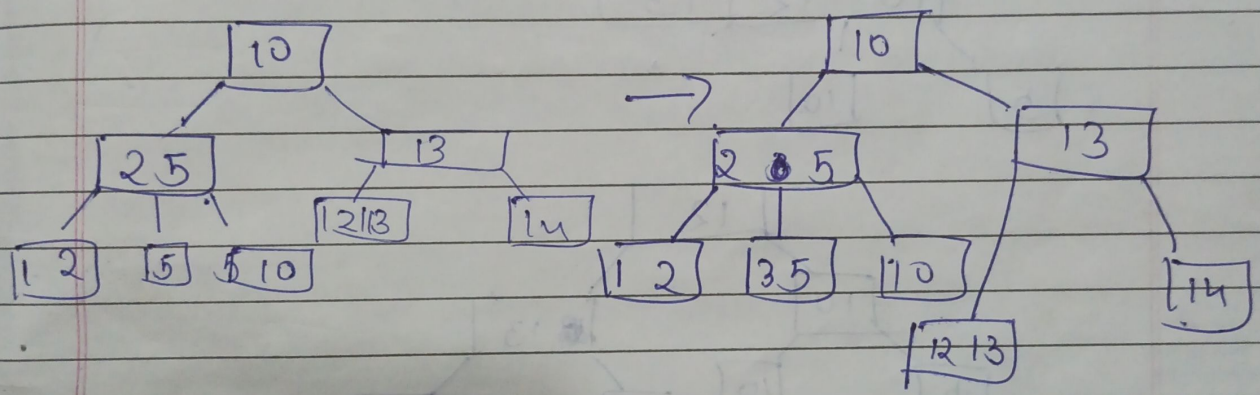
Maximum no. of Record that can be indexed if 'n' is degree and 'h' is height.

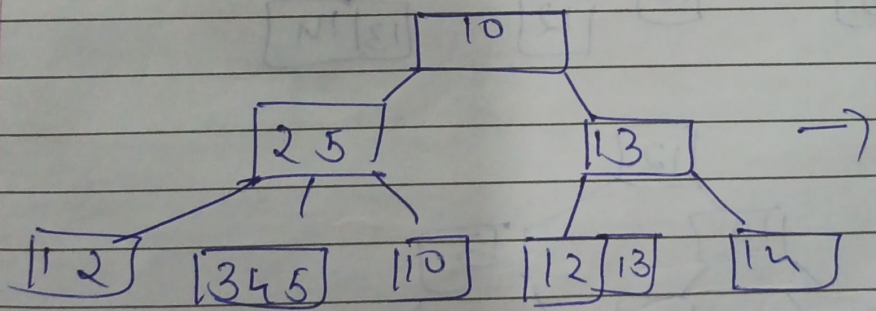
$n^{h+1} - 1$ Ans

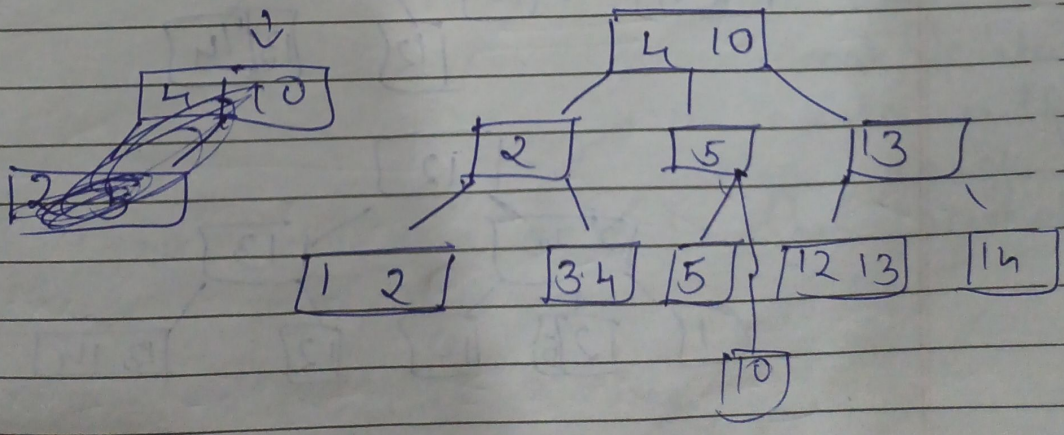
Pehle insertion phir overflow check karte hai



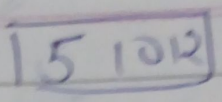


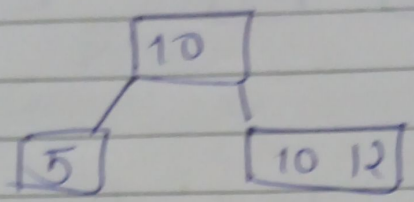


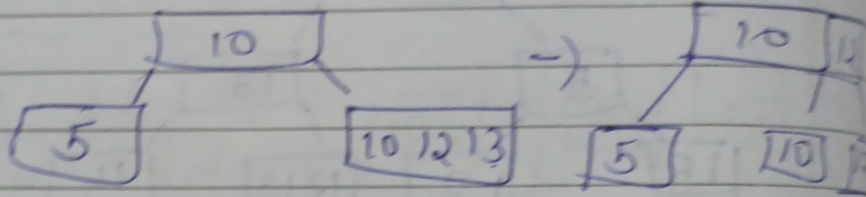


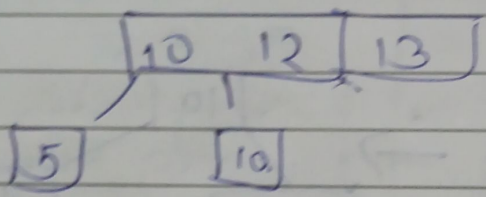


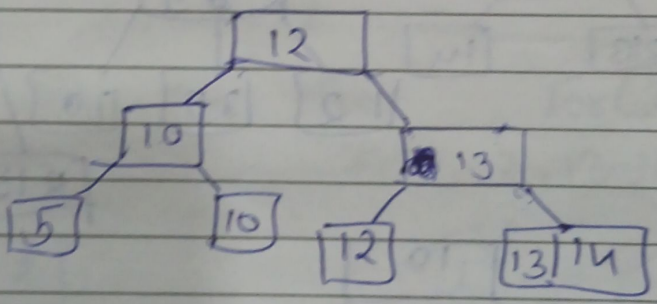
Ex 3 : [Right Copy] 5, 10, 12, 13, 14, 12, 3, 4

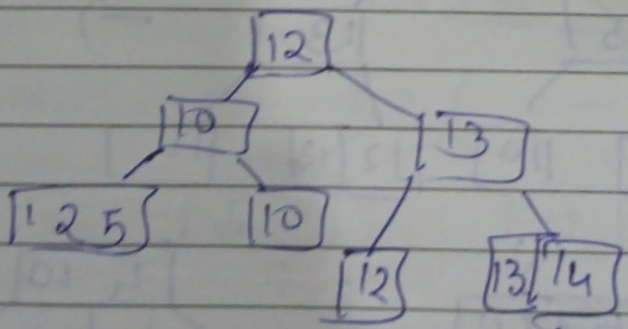


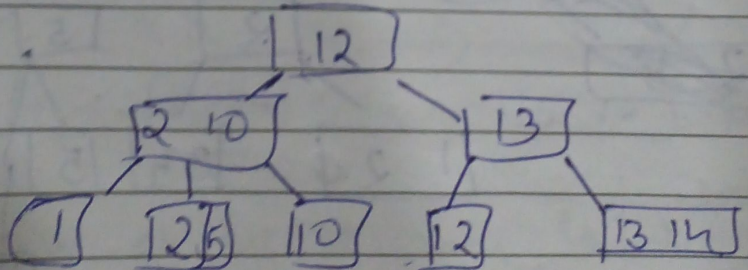


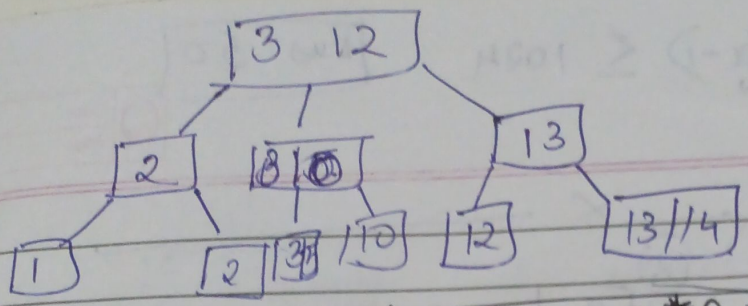












order of B+ Tree

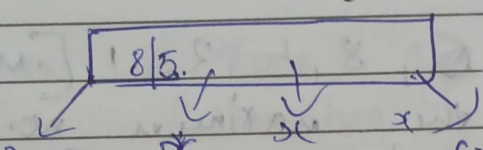
non leaf $\rightarrow *P + (P-1)K \leq$ Block Size

leaf $\rightarrow$ leaf $* (P_b + K) + P \leq$ Block size

Q Consider a B+ Tree and if the search key value is 8 Byte long and Data Base Block is 512 Byte Block, the Block pointer is of 2 Byte. what is the maximum order of B+ Tree posible.

13

B.



≤ 512

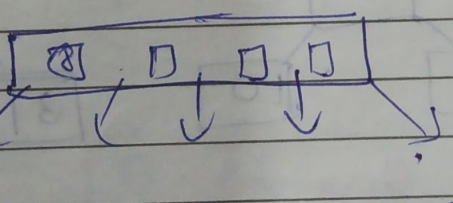
$2x + 10(x-1) \leq 512$

$2x + 10x - 10 \leq 512$

$12x \leq 512$

$x = \frac{512}{12} = \underline{\underline{45}}$

B+



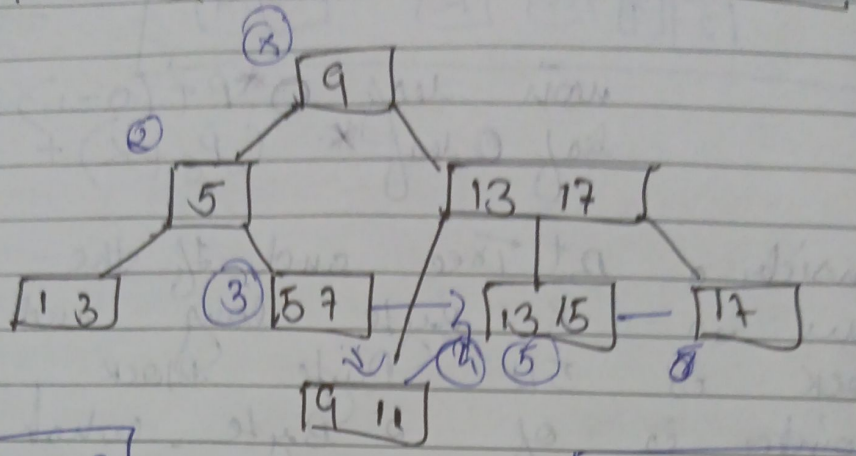
$2x + 8(x-1) \leq 512$

Q Consider a B+ tree where search key 12 Byte Block size 1024 Byte. Record pointer is 10 Byte long and Block pointer is 8 Byte long. what is the maximum no. of key's can a non leaf that can accomodate a non-leaf Node.

$8x + 12(x+1) \leq 1024$

$12x + 8(x-1) \leq 1024$

Ans = 50

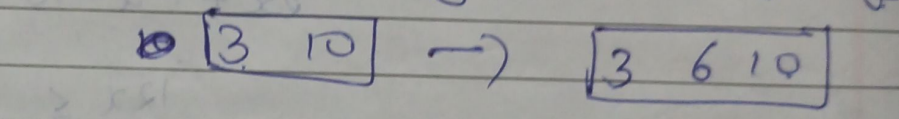


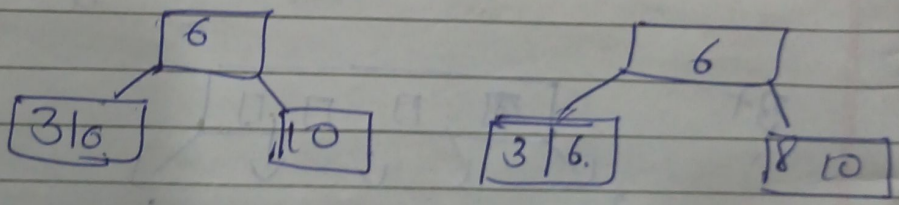
Ans = 5

$7 \leq x \leq 15$

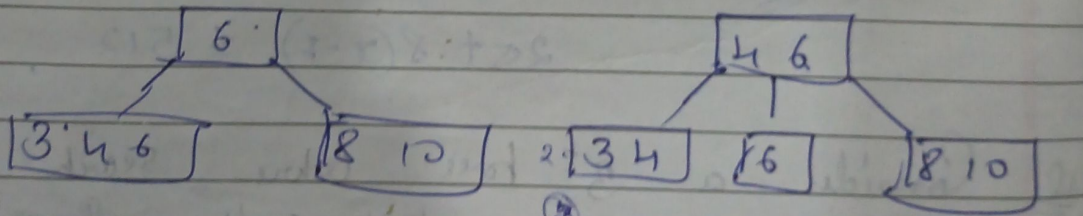
Q. 10, 3, 6, 8, 4, 2, 1 [M = 3]
what is the maximum no. of times leaf node get spilled. Try Left and Right Copy

2
3
4
5.

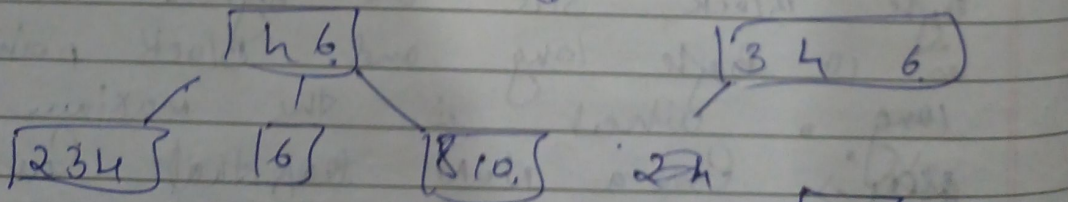




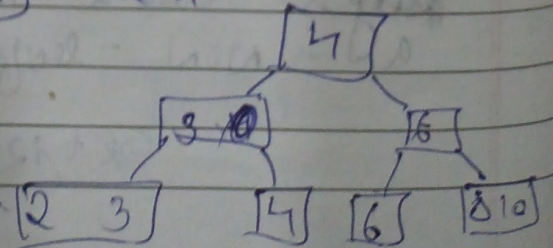
①

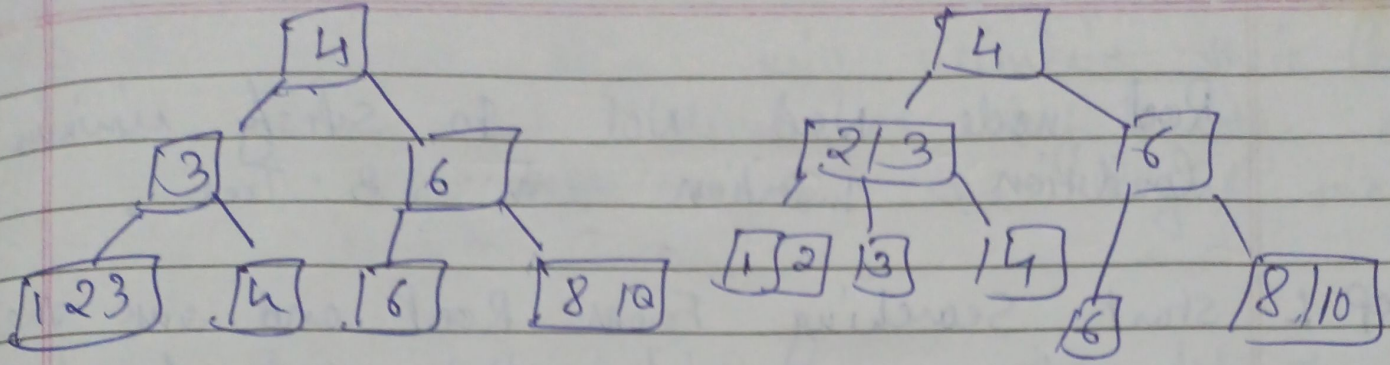


②



2 4





In left Copy. Total = 4

* B Tree Deletion :

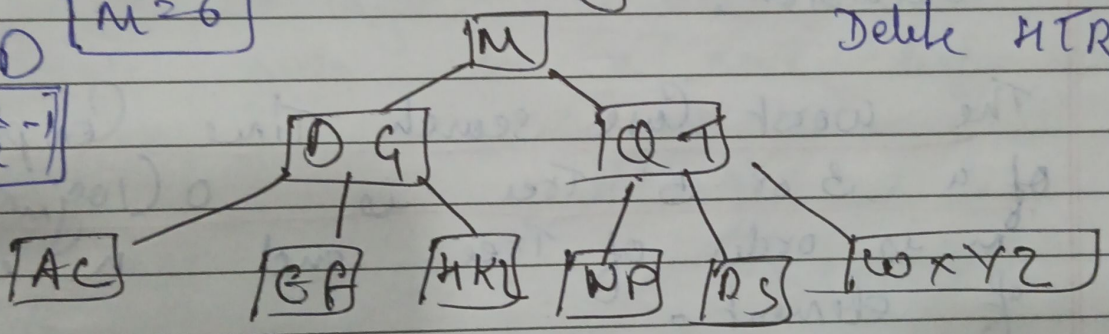
Q Consider a B Tree of order 6 and Delete the following nodes in sequence

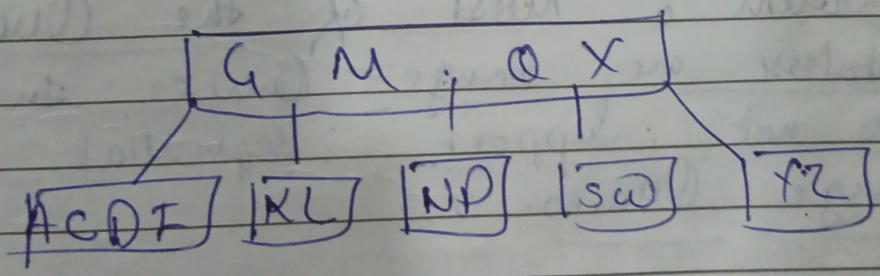
Delete HTREACS

M=6

C D

min $\left\lceil \frac{M}{2} \right\rceil \mid \left\lceil \frac{M}{2} \right\rceil - 1$





B Tree :- B stands for Balance Tree Every B Tree is defined with order [M] which specify the maximum number of children a node can have

min $\left\lceil \frac{M}{2} \right\rceil \mid \left\lceil \frac{M}{2} \right\rceil - 1$

max $M \mid M-1$

M=6

Root node Need Not to satisfy minimum Condition Insertion in B-Tree.

Step 1 Start Searching From Root and sink the key to appropriate leaf and insert it into the leaf.

Step 2. If there is a overflow then pick the median and push it into parent. Same process must be repeated Recursively. [If order is even then clash can be resolved either with left median or right. conclusion.

The worst Case search Time Complexity of a B or B+ Tree is O(log$_m$ n) where m is order of Tree and n is number of element.

Problem :- Most of the Queries in Database are Range Queries in simple B+ Do not support sequential search for Range Queries.

Solution :- B+ Tree are same as B Tree and all deletion and insertion protocol are same except two changes.

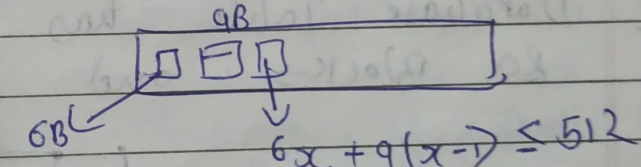
1. Each leaf Contains a Tree pointer in the Extreme Right which points to its Right sibling which points to its Right

2. All internal keys will maintain their copy on the last level or leaf either on the left or on the Right that is inorder successor.

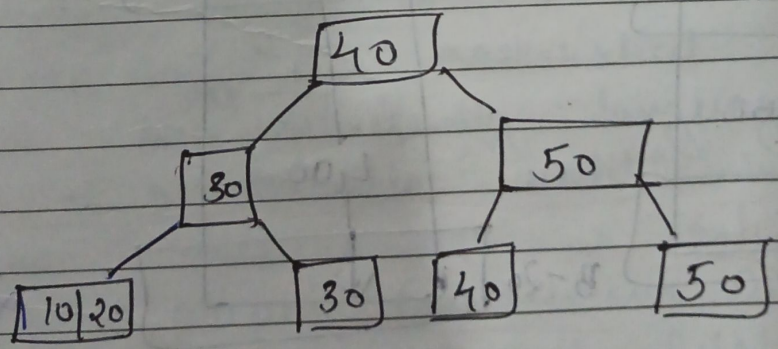
NOTE :- A modification is suggested in a B+ Tree when all the internal Nodes instead of holding data and record pointer pair we holds only data.

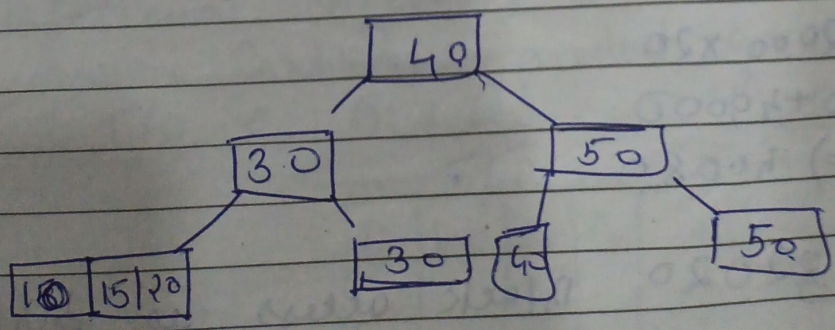
Q In a indexing system search key filed is 9 byte long block size 512 Record pointer 7 byte long and Block pointer 6 Byte what is the largest possible order on a non leaf node on B^+ Tree ?

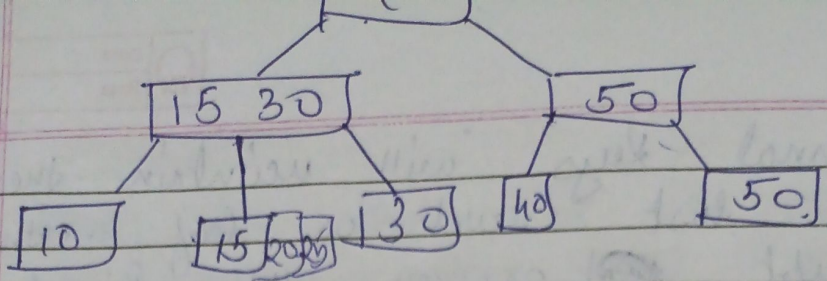
(a) 23
(b) 24
(c) 34
(d) 44

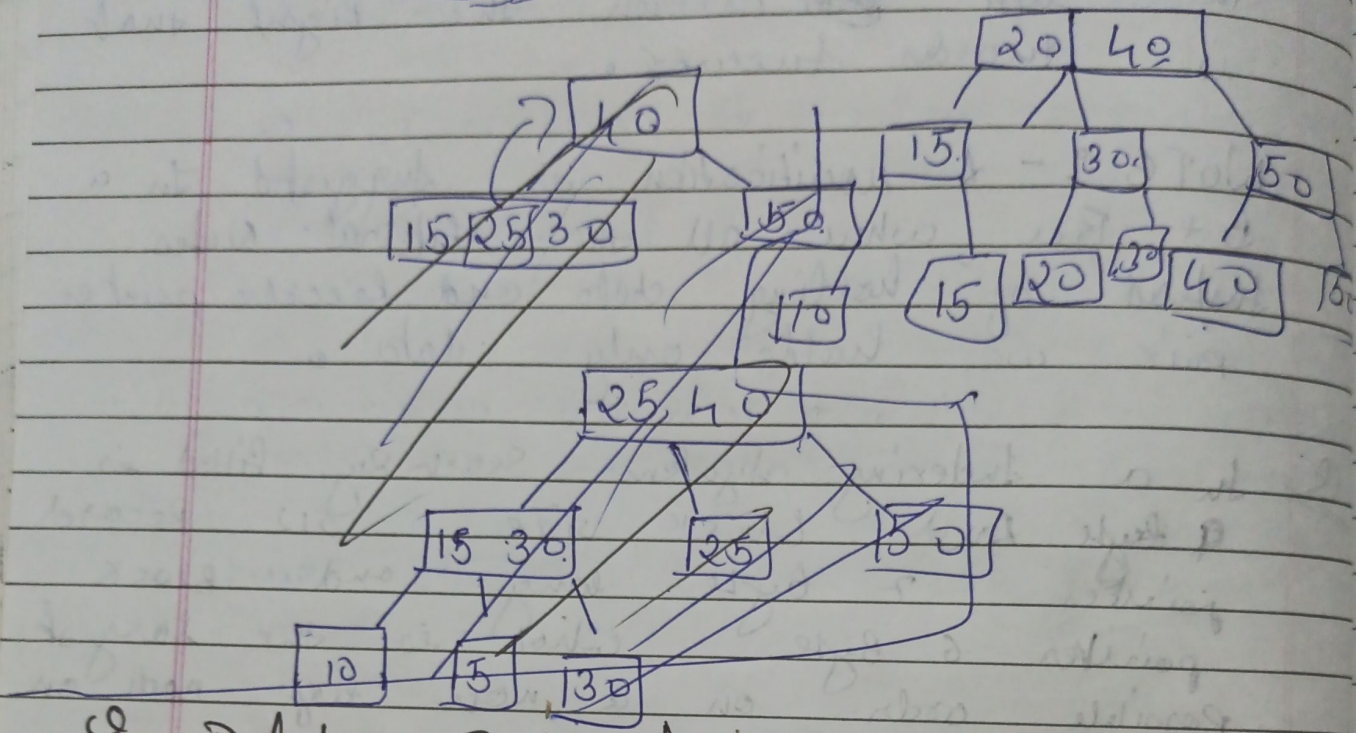


Q

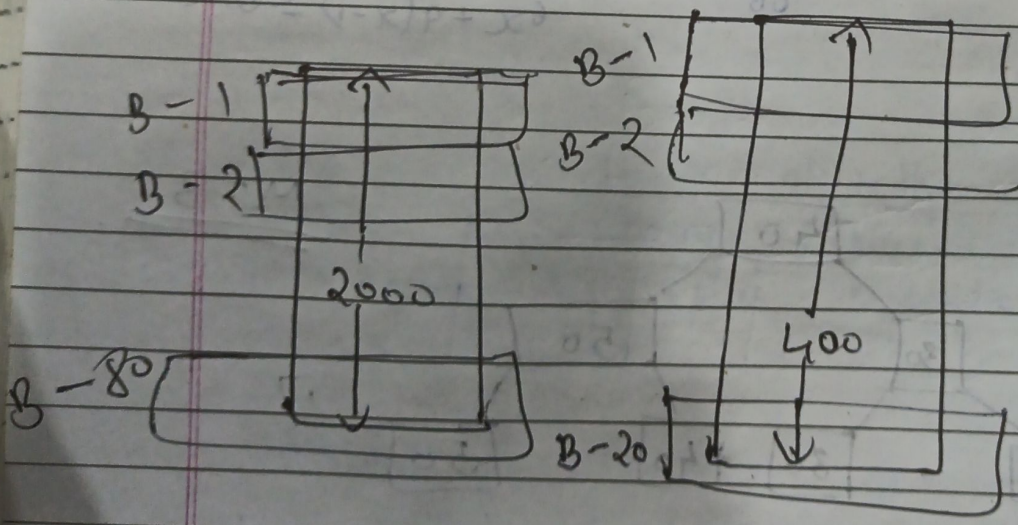








Q Database Table has 20000 Records and 80 Blocks and



$80 + 2000 \times 20$

$80 + 40000$

$\Rightarrow 40080$

$20 + 400 \times 80$

32000

$32,020$

32020 Block access is sufficient

* Query language (2-3) marks

Query language :- 1. Query language is which a user request information from the DataBase.

2. It can be divided into two types procedural and NON-procedural.

1) Procedural :- Here user instruct the system to perform a sequence of operations to do a desire task.

For Ex :- Relational algebra is a mathematical model which is procedural.

2) NON-Procedural :- Here user only describes the desired information without describing what is the procedure to obtain it.

Ex :- Relational Calculus.

NOTE :- Sql uses the features of Both Relational algebra and Relational Calculus.

* Relational algebra :- 1. It is not user friendly but it design a mathematical frame work for a user friendly language called sql.

2. Algebra is a theoretical model which do not run on any machine it is only used for Identity purpose (Research and Development).

3- In Relational Algebra, A Relation is a subset of the Cartision product of all the domain

$T/R \subseteq D_1 \times D_2 \cdots D_n$

T/R

D_1	D_2	..	D_n

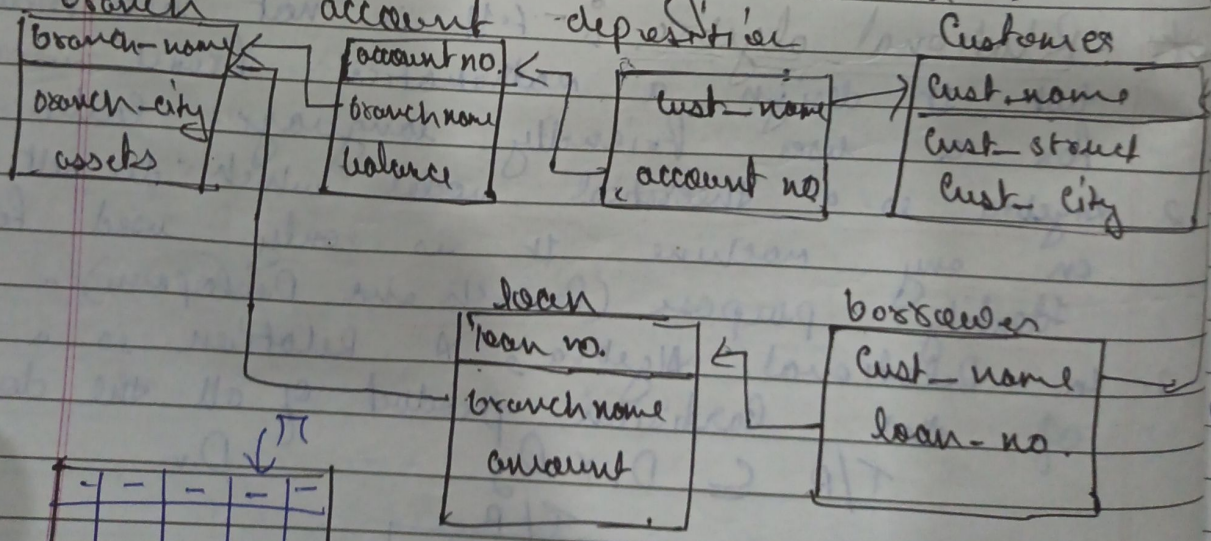
For any Relational algebra Query Input output are always in term of Relation Table where no. of Input is min ⇒ 1 and Max ⇒ 2. But as a output we exactly 1 table or Relation without name.

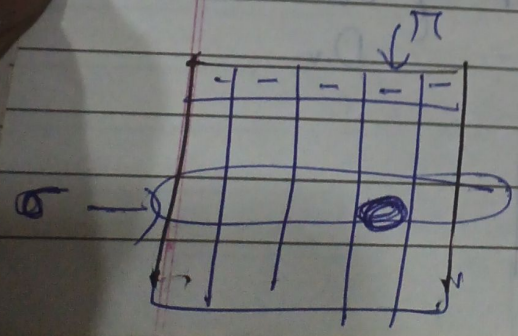
For Input either we provide name to the Entire table.

Following are fundamental operator of Relational algebra :-

1. Select (σ)
2. Project (π)
3. Union ($\cup$)
4. Set difference ($-$)
5. Cartision product ($\times$)
6. Rename (ρ)

Q Consider a Banking Schema :-





Q. Find all the information of the account where branch is GKM

Format => σ (condition (table name))

σ branch name = GKM (account)

Find Information about those loans where where branch is C.P and amount is greater then 10000.

σ amount > 10,000 (σ branchname = C.P (loan))

σ branchname = C.P. (σ amount > 10,000 (loan))

σ branchname = C.P $\wedge$ amount > 10,000 (loan)

* Projection :- Projection is also a unary operater which can select any set of Columns.

Format π Column name (table name)

Find all the Customers name.

π Cus_name (Customer)

Find those account numbers which are in Kanpur branch ? where ~~balance~~ is greater than ~~5000~~.

σ branchname = Kanpur (π accountno (account ~~...~~)) $\wedge$

π account_no (σ branchname = Kanpur (accoun))

Q. Find all the information of the account where branch is GKM

Format => σ (condition (table name))

σ branch name = GKM (account)

Find Information about those loans where where branch is C.P and amount is greater then 10000.

σ amount > 10,000 (σ branchname = C.P (loan))

σ branchname = C.P. (σ amount > 10,000 (loan))

σ branchname = C.P ∧ amount > 10,000 (loan)

* Projection :- Projection is also a unary operater which can select any set of Colounm.

Format π Colounm name (table name)

Find all the Customer name.

π Cus_name (Customer)

Find those account number which are in Kanpur branch ? where ~~balance~~ is greater than ~~500~~.

σ branchname = Kanpur (π account_no (account ...)) ∧

π account_no (σ branchname = Kanpur (account))

Q) Find those loan no. which are from gwalior branch and having balance amount less than 5,000 ?

$\Pi_{loan\ no}(\sigma_{gwalior\ bran \wedge amount < 5000}(loan))$

Q) Find those Customer name who have either a loan or a account or both?

$\Pi_{cust\_name}(depositior) \cup \Pi_{cust\_name}(borrower)$

NOTE :-

Sql supports duplicacy But algebra doesn't supports.

Q) Find those Customer name who have account in gwalior branch.

$\Pi_{Cust\_name}(\sigma_{account.acct\,no = depos.acc\,no.}(account \times deposition) \wedge branch\,name = gwalior)$

Q) Find those account_no. along with their balance which are in a branch which is situated in gwalior and their asset $\leq 50,00,000$.

$\Pi_{account\,no,\ balance}(\sigma_{branch\_bs.name = accou.bs\_name}(branch \times account) \wedge branch\,city = gwalios \wedge asset < 50.Cs$

Q) Find those Customer name which have a loan from a branch which is situated in gwalior

Date
Page

(branch x loan x borrower)

customer name (σ bs.bname = loan.bname
∧ loan.loanno = borrow.loanno.
∧ branchcity = gwalior

Natural Join ⋈ :- Natural Join is also a binary operator which is a improved version of Cartisian product where it consider those rows which have a common value of the common attribute.

R_1

A	B
a	1
b	2
c	3

R_2

B	C
2	p
3	q
4	r

$R_1 \times R_2$

A	R_1.B	R_2.B	C
a	1	2	p
a	1	3	q
a	1	4	r
b	2	2	p
b	2	3	q
b	2	4	r
c	3	2	p
c	3	3	q
c	3	4	r

$R_1 \bowtie R_2$

A	B	C
b	2	p
c	3	q

$R_1 \bowtie R_2$

A	B	C
a	1	null
b	2	p
c	3	q

$R_1 \bowtie R_2$

b		
b	2	p
c	3	q
null	4	r

$R_1 \bowtie R_2$

A	B	C
a	1	null
b	2	p
c	3	q
null	4	r

The Problem with Cartisian Product is that it Generates a number of extra tuples which is solved by natural Join which consider only Common Value but 1 draw back is some information is

only either in Table 1 or 2 then it will get lost.

Solution :- Modified Natural Join :-
① Left Natural Join .
② Right " " .
③ Complete " " .

① Left Outer Join :- Left outer Join gives priority to left table or first operand.

② Right Outer Join :- Right outer Join gives priority to Right table or 2nd table.

③ Complete Outer Join :- C.O.J give priority to both.

NOTE :- Left and Right are Not Commutative but Complete is Commutative.

Dealing with Null Value :- If any arithmetic equation or statement Contains a null then the Result will also be always be null.

Any Comparison if Involves null then the Results will also be null.

Exception with logical operator :-

T ∪ null ⇒ T
T ∧ null ⇒ null
F ∪ null ⇒ null
F ∧ null ⇒ F

Q Find the largest account balance in bank.

$\Pi_{balance(account)} - (\Pi_{B.balance}(\sigma_{A.balance < B.balance}(\rho_A(account) \times \rho_B(account))))$

A	B
100	100
100	200
100	300
200	100
200	200
200	300
300	100
300	200
300	300

A.balance
100
200
300

B.bal
100
200
300

Q Find all the Cust_name who have account in all the branch located in Gwalior.

$\Pi_{cust\_name, br.name}(account \bowtie depos)(branch)$

$\div$

$\Pi_{br.name}(\sigma_{br.city = gwalior}(branch))$

* Division Operator ÷ - is also a non fundamental operator which can be used when 1 information is repeated for all the other information.

Q. $R_1(A,B)$ $\quad$ $R_2(C,D)$

(a) $\pi_B(r_1) - \pi_C(r_2) \neq \phi$

(b) $\pi_B(r_1) - \pi_C(r_2)$

(c) $\pi_C(r_2) - \pi_B(r_2) = \phi$

(d) $\pi_B(r_1) - \pi_C(r_2) = \phi$

Q.

CR

S.N	CN
SA	CA
SB	CB
SA	CC
SB	CB
SB	CC
SC	CA
SC	CB
SC	CC
SD	CA
SD	CB
SD	CC
SD	CD
SE	CD
SE	CA
SE	CB
SF	CA
Sf	CB
SF	CC

$T_1 \leftarrow \pi_{cname}(\sigma_{S.N=SA}(CR))$

$T_2 \leftarrow (R \div T_1)$

SA
SC
SD
SF

T_2 AWS=4

Q.

R_1

A	B
1	5
3	7

R_2

A	C
1	7
4	9

$R = R_1 \bowtie R_2$

	A	B	C
1.	1	5	null
2.	3	null	7
3.	4	7	1
→ 4.	1	5	7
→	3	7	1
→	4	4	9

$R = R_1 \times R_2$

B	A	C
5	1	7
7	3	4
4	4	9

cl employee (emp-id, empname, emp-age)
dependent (dep-id, emp-id, dep-name, dep_age)

$\Pi_{empid}(emp) - \Pi_{empid}(emp \bowtie_{(empid = eid \wedge emp\,age < dep\,age)})$

(a) Some dependents.
(b) all
(c) Some his/her''
(d) all his/her''

emp

id	name	age
1	A	22
2	B	23
3	C	24

id	eid	dn	dage
1	1	x	19
2	1	y	22
3	1	z	23
4	2	p	16
5	2	q	12

IN SQL,

* UNION, EXCEPT, INTERSECT (Removes/Eliminates Duplicates)
* UNION ALL, EXCEPT ALL, INTERSECT ALL (Does not eliminate Duplicate)

SQL

1. SQL stands for structured Query language which is the most influential and commonly used Query language in industry.

2. It has a capability to define data definition, data manipulation, views definition, embedding it will other programming language, Transaction, authorisation, integrity.

3. SQL suffer from duplication but it provide a operator distinct using which we can diminate duplicacy.

4. The Reason sql consider it optional to Remove duplicacy that is if user Schema do not contain duplicacy or duplicacy is irrelevant to user then use of distinct keyword not required then speed of processing will be high. otherwise we can use distinct keyword.

Q Find all the information about account.

(account)

In Sql the standard query has 3 clause
Select (Π) $A_1, A_2 \ldots A_n$ (Column name)
From (,) T (Table name)
where (σ) C
Here For the sake of flexibility select and from are Treated as a mandatory clause.

Q. Find all the account number where balance is less than 5000.

```
Select   account no
From     account
where    amount ≤ 5000
```

NOTE :- In sql Select clause has capability to perform any numeric operation.

Q. Find those loan numbers where amount is ≥500 but ≤1000.

$500 \leq amount \leq 100$.

```
Select   loan no
From     loan
where    amount ≥ 500 and amount ≤ 1000
         amount between 500 and 1000
```

here between keyword include boundry keyword.

Q. Find those Customer names, who have account, or a loan or both.

```
Select   Cust_name
From     depositior
[redacted]
Union
Select   Cust_name
From     [redacted] borrower.
```

all the set operates can be executed by their names.

Q Find those customer_name who have a loan ₹5000.

```
Select    Cust_name
From      loan, borrower
where     amount ≤ 5000
          and loan no. = borrower loan no.
```

```
Select    Cust_name
From      loan NATORAL JOIN
where     amount ≤ 5000
```

* aggregate function :- to min, max and count are also applicable on character where a is considered as minimum and z is consider as maximum.

to (In all 5 aggregate function only count can work on null and min, max, sum and average simply ignores null)

* string matching :-

Q Find all the details of the account where branch name is gwalior.

Select => *

From => account.

Where branch name = 'gwalior'

1. SQL is not Case Sensitive but string matching in SQL is Case Sensitive.

2. We have 'like' operator which is much more stronger than simple Comparision along with two operators (_, %) where '_' is used to Count characters and '%' is used for ~~Count~~ prefix, suffix ~~or~~ subscript.

Q Find average balance of bank.

```
select avg(balance)
From account
```

Q Find average balance of every branch in Bank.

```
Select branch name, avg(balan)
From account.
group by branch name.
```

bon	b.
A	10
A	20
B	30
C	40
C	50

A - 15
B - 30
C - 45

Q Find the Branch name whose average balance is greater than 1500 and must be situated in gwalior ?

Select Branch-name
From branch ⋈ account
(Table) → where branch city = gwalior
group by branch-name
(group) → having avg (salary) > 40[redacted]

having :- is a Conditional operator which is applicable on groups while where can work only on the entire table but Cost of implementation in terms of time For having is more than of where therfore if possible then we should where and only necessary and essential situation we should use having.

Ordering OF the Rows according to a Column is also posible using order by clause

Q//

A

id	name	age
12	An	60
15	Sh	24
99	RO	11

B

id	name	age
15	Sh	24
25	Hr	40
98	RO	20
99	RO	11

C

id	phone	ag
10	2200	02
99	2100	01

(a) 7 $(A \cup B) \bowtie (Aid \rightarrow 40) \vee (Cid < 15)$
(b) 4
(c) 5
(d) 9

Q/ select A, id
from A
where A age > all (select B age
from B
where B name = 'As')

(a) 4
(b) 3
(c) 0
(d) 1

Q/

Borrower	Bank Manager	loan amount
Ra	Sun	10
Sur	RAM	50
Ma	SON	70

select Count (x)
from (select Bor, BM
From LR as s
natural join
Select Bm, L as
From LR as T

a) 3
b) 9
c) 5
d) 6

Q/ Select title
From Book as B
where (select Count (*))
From Book as T
where T.price > B.price) < 5

B

T	P
B1	1
B2	2
B3	3
B4	4
B5	5
B6	6
B7	7

T

T	P
B1	1
B2	2
B3	3
B4	4
B5	5
B6	6
B7	7

Ans = 5

Relational Calculus

TRC — DRC

Q

R.A: $\sigma_{br = 'C.S'}(student)$ — Student (Rollno, name, br)

SQL :-
```
Select x
From Student
Where br = 'C.S'
```

TRC :- $\{x \mid x \in student \wedge x.br = 'C.S'\}$

DRC :- $\{(Rollno, name, br) \mid student(Rollno, name, br) \wedge br = 'C.S'\}$

O/P — × — I/P

Q Find the name of student of Electronics branch

R.A :- $\Pi_{name}(\sigma_{name = 'EC'}(Student))$

SQL :-
```
select Name
From Student
Where br = 'EC'
```

TRC :- $\{x.Name \mid x \in Student \wedge x.br = 'E.C.'\}$

DRC :- $\{Name \mid Student(Rollno, name, br) \wedge br = 'E.C.'\}$

O/P — × — I/P

Q Consider the following Queries on TRC and find out the meaning. On Banking schema

① $\{t \mid t \in loan \wedge t[amount] > 1200\}$ — Projection

② $\{t \mid \exists s \in loan\ (t[loan\ no] = s[loan\ no] \wedge s[amount] > 1200\}$

③ $\{t \mid \exists s \in borrower\ (t[customer\ name] = s[cust\_name]\}$
$\wedge (\exists u \in loan\ (u[loan\ no.] = s[loan\ no.]$
$\wedge (u[br\ name] = \text{'noida'}))\}$ — returned bank table

④ $\{t \mid \exists s \in borrower\ (t[cust\ name] = s[cust\_name]$
$\vee \exists u \in deposition\ [t[cust\ nos] = u[cust\ nos\_name]\}$

Find those loan no. where amount is Greater then 1200.

Find the name of the Customer who either have a loan account or both.

Find the name of the Customer who have account in Noida.

Q Consider the following Queries in DRC and Identify their meaning:

Q Find the details of the loan where amount is greater than 1200.

⇒ $\{\langle l, b, a \rangle \mid \langle l, b, a \rangle \in loan \wedge a > 1200\}$
O/P I/P

Q Find the loan no. where amount is greater the 1200

⇒ $\{\langle l \rangle \mid \exists b, a\ (\langle l, b, a \rangle \in loan \wedge a > 1200)\}$
output Not op I/p

Q Find the Customer name and along with the account who have a loan from noida branch.

⇒ $\{<C,a> \mid \exists l (<C,l> \in \underbrace{borrower}_{JTP} \wedge \exists b (<l,b,a> \in loan \wedge b = 'noida'))\}$

Q Find the name of the Customer who have either a loan or a account or both from Noida branch.

✱ TRC :- In Relational Calculus there are two methods.

1. Tuple Relational Calculus
2. Domain Relational Calculus

TRC :- In TRC the
$\{t \mid Condition\ on\ t\}$
where t is a variable and Condition helps to identify or filter the required tuples.

DRC :-

$\{(D_1, D_2, D_3 \mid Condition\ (D_1, D_2, D_3 \dots D_n, D_{n+1})\}$

In DRC in the first clause we identify the required Domains or Columns and then we write Conditions which

are applicable to all domains.

Student (Roll no, name)
Course (Cno, Cname)
Reg (Roll no., cno, percent)

Find the distinct name of all the students who scores more than 90% in Course number 107.

SQL :- Select S.name
From Student as S, Reg as R
where Roll no. = S.roll no.
and R.College no = 107
and R.percent ≥ 90.

R.A ⇒ $\pi_{name}(\sigma_{Courseno = 107 \wedge percent > 90}(Reg \bowtie Student))$

TRC ⇒ $\{T \mid \exists S \in Student, \exists R \in Reg (S.roll no. = R.roll no. \wedge R.CourseNO = 107 \wedge R.percent > 90.$

DRC ⇒ $\{\underbrace{\langle S_N \rangle}_{O/P} \mid \underbrace{\exists S_R \exists R_p}_{No\ of\ O/P} (\underbrace{\langle S_p, S_N \rangle \in Student \wedge \langle S_p, 107, R_p \rangle \in Reg \wedge R_p > 90}_{I/P})\}$.

2016 All In Blanks.

```
with total(name, capacity) as
select d_name, sum(capacity)
From water scheme
group by d_n
with total_avg (capacity) as
        select avg (capacity)
        from total.
select name
from total_avg
where total capacity ≥ total avg – capacity.
```

water scheme

S no	d_n	Capacity
1	Aj	20
1	Bi	10
2	Bi	10
3	Bi	20
1	Ch	10
2	Ch	20
1	Ou	10

total

name	Cap.
Aj	20
Bi	40
Ch	30
Ou	10

~~Find the address of the~~ ~~maximum capacity.~~

emp (e_id, e_name, ...)
Cost (e_id, c_name, ...)

```
Select e_name
From emp as e
where not exist (select c_id
                 From cust as c
                 where c_sales_id = e_id
                 and c_satis <> 'good')
```

---

we want to print last name & hire date of all the new hires in the respective department in the location 1700 (number)