DBMS

(9)

18/08/2011

Contents:-

⇒ Introduction & Integrity Constraints.
Referential Integrity Constraints
ER model.

⇒ Schema Refinement (Normalization) : [2+2] marks

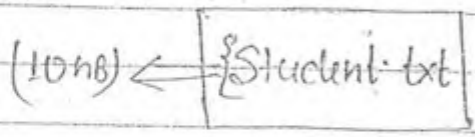
⇒ Transactions & concurrency control. : [2] mark

⇒ Query Lang → Relational Algebra
→ Tuple Relational Calculus : [2+2] mark
→ SQL

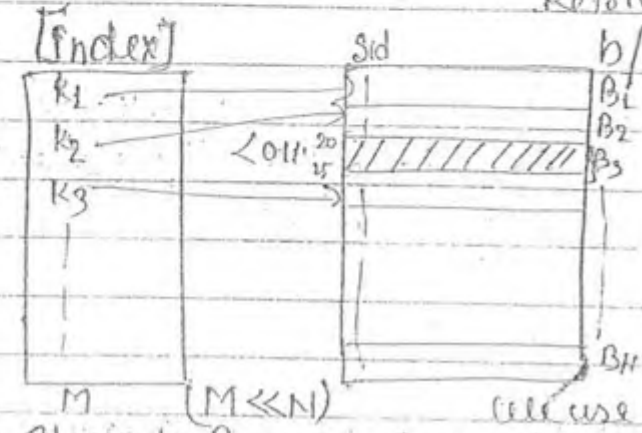
⇒ Indexing & File Organization [6-8] : [2] marks

[2] I/O cost is too huge if data requirement from large sized file.

Example:-



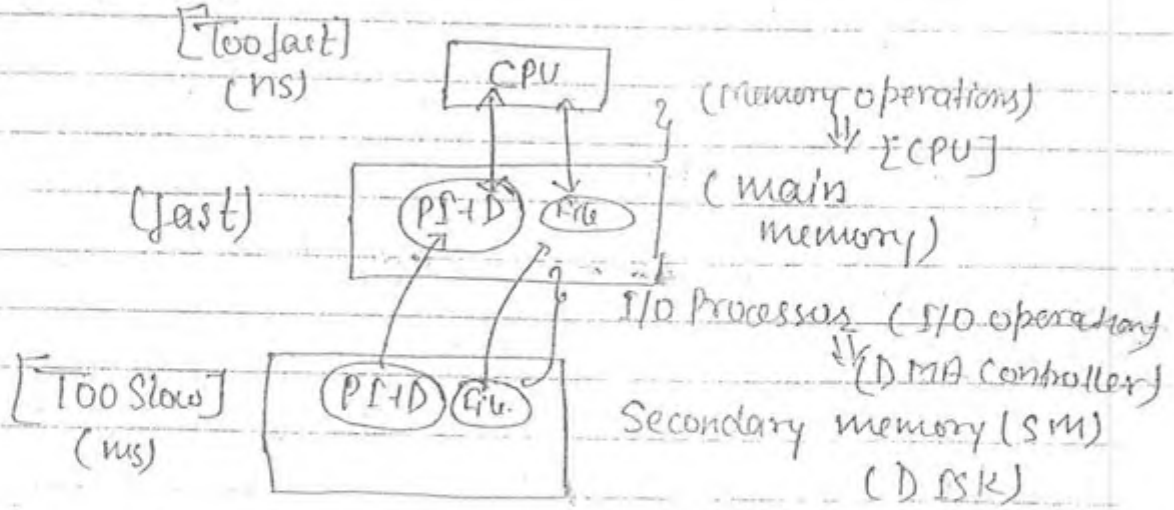
Retrive Students whose RollNo b/w 20 to 25.



We use Stored programming concept in file system as similar as in DB

Stored Programming Concept:-

Progsam + Data Require to execute program should be stored in Highspeed main memory in order to execute by CPU.



I/O Cost:-

No. of Blocks required to transfer from secondary memory (SM) to main memory (MM) in order to access any record.

In Worst Case I/O Cost N blocks.

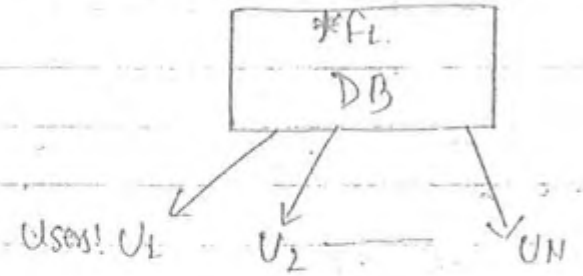
Use of Indexing:-

If we have a book of 500 pages

		I/O cost
Without Index	500	500
30 index pages +	500	30 + 1 (To access the current page)

3) Concurrent Access:- In general when Write ↔ Read/Write, when Read ↔ Write we can't

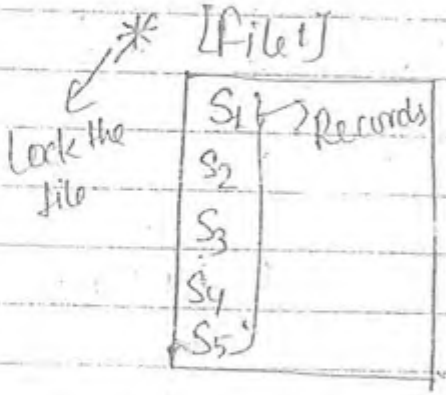
* less concurrency level because of file level locks.



* Operating system also provide concurrency control at the file level:

Operating system lock the entire file at the time of updation.

Drawback of file level concurrency control:-



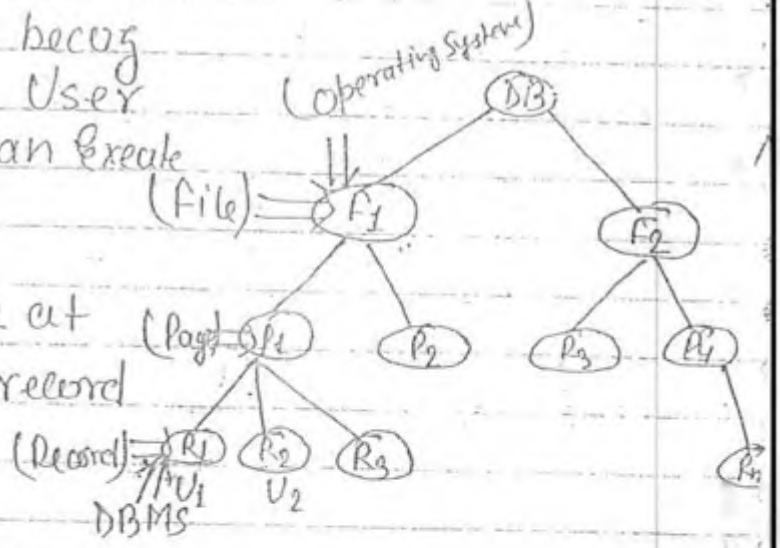
If user U_1: Update S_1
U_2: Read S_3

There is no inconsistency Still file level locks Not allowed to execute U_1 & U_2 operations simultaneously.

Benefit of DBMS in concurrency:-

* Record level locking becoz of low level locking User U_1 & U_2 operation can execute simultaneously.

* DBMS locking is done at low level i.e. at record level.



* High level lock means less concurrency i.e. operating system uses high level locking because of this at file level concurrency decreases.

* low level lock means more concurrency i.e. DBMS uses low level locking because at record level concurrency increases.

4) Security!-

* Security to query file (i.e. password protected) but security requirement is data within file.
* Operating system provide security at only file level

File!-

Sid	Sname	Marks	DOB	Address	Phoneno

Two users

"Faculty" need (Sid, Sname, Marks)

Admin (Sid, Sname, Marks, Addr

O/P system needed ~~two~~ create two file for "Faculty" & "Admin" who contain data item as (Sid, Sname, Masks) & (Sid, Sname, Masks, Address). Using this methods Redundency causes. So, operating system is no applicable for security.

Advantages of DBMSL-

VIEWS
[Virtual Table]

View StudFac (Sid, Sname, Marks)

StudInfoAdmin (Sid, Sname, Middle

File:

Sid	Sname	Marks	DOB	Address	PhoneNo

One of the Data Model!-

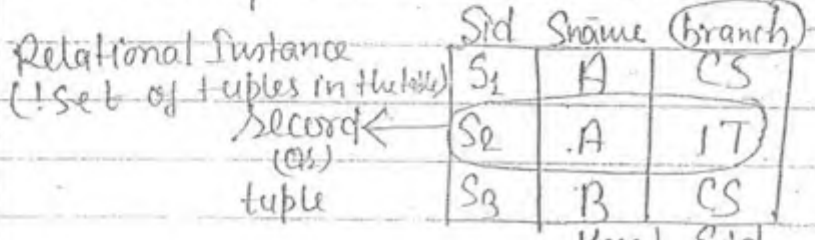
Relational DBMS!-

Every data should be stored in propes Tabulas formate.

Table!-

Collection of rows's and colums.

* DBMS take the data in proper formate formate & every column their should be a unique name.



It is a table with 3 fields or 3 attributes.

Attribute Range!-

possible values accepted by attribute.

Cardinality!- No. of Records of table.

Abstraction of Table (or) Relational Schema } Student (Sid Sname branch)

* If table contains some set of table tuples then we called it Relational Instances.

Codd Rule:-

Every tuple should be differentiate from other tuple.

(Or)

No two tuple of the Relation shou be same in the table.

To preserve above constraints we have to follow some rules:-

Key:-

Min. no. of attributes used to differentiat all the tuples of the Relation.

* Key may contain more than one attributes

Example:-

Sid	Cid	—
S_1	C_1	
S_1	C_2	
S_2	C_1	

(Sid Cid) : Key

⇒ Composite key:-

Key forms with two or more attributes.

⇒ Simple key:-

Key forms with single attrib.

* A relation may contain more then one key.

Example:-

Sid	Sname	PassPno (PPno)	LicenseNo (Lno)	DOB	Fname
S_1		Null		10/10	Y
S_2		P_4		10/10	X
S_3		Null		11/12	X
S_4		P_3		Null	
S_5		Null			

(Unexisted) (Unknown)

Assume Constraints:- No two students with same DOB & Fname.

Keys:- {Sid, PPno, Lno, (DOB Fname)}

* Within the key there should not be a key of proper subset of any key.

i.e. If ABC ! key i.e. if (DOB Fname) is
then { A is not key then
B " " " Alone DOB is not key
C " " " Alone Fname is not key
AB " " " then only if (DOB & Fname
BC " " " together uniquely different
AC " " " the tuple then only they are keys.

Proper subset of ABC is not allowed as key but if ABC is a key

* All the keys collectively called as Candidate Key.

Candidatekey :- {Sid, PPno, Lno, (DOB Fname)}

* Candidatekey is same as key
i.e. Sid is one of the C.key as PPno, LNo, (DOB Fname) &
Of these are C.key

* Prime Attribute:-
Attributes belongs to Candidate key is called Prime attribute set (OR) Key attribute set.

Example:- Sid, PPno, Uno, DOB, Fname are prime attribute

* Otherwise Non-Prime attribute set (OR) Non key attribute set.

Example:- Non Prime attribute Set {Sname}

Primary key:-
One of the Candidate key.

Alternative keys (Secondary key):-
All candidate key except primary key is called alternative key.

Primary key	Alternative keys
⇒ At most one primary key for each relation	⇒ More than one alternative keys allowed for each relation
Let Sid : Primary key	then {PPno, Uno, (DOB Fname)} Alternate key

Candidate key

(Primary key) (Alternative key)

Primary key	Alternative key
⇒ Primary key not allowed Null Value.	⇒ Allowed Null Value
Null : Unknown (or) unexisted Value.	

(integer) ↑ (integer)

Index Sid

Sid	Sname	PPno	Lno	DOB	P
S_1		Null	L_4	10/10	Y
S_2		P_4	L_3	10/10	X
S_3		Null	L_1	11/12	X
S_4		P_3	L_2	Null	

↓ Unexisted Value ↓ Unkn

Constraints of Primary keys:- (DBA allowed)

* Key with No null values.
* If two candidate key not contain the Null value
* default Index is done on the Primary key.

"Key" which is used to access DB more frequently.

* If both two keys are used 50% - 50% then to access the data goes for 3rd constraints i.e. Numerical values gain more priority to design Primary key.
* If all the three constraints are satisfied by two candidate key then go for next constraints i.e Candidate key with less no. of attributes.

UNIQUE :- It is a keyword to show alternative key.
* Unique along with not null is same as primary key.

Table Creation Command :-

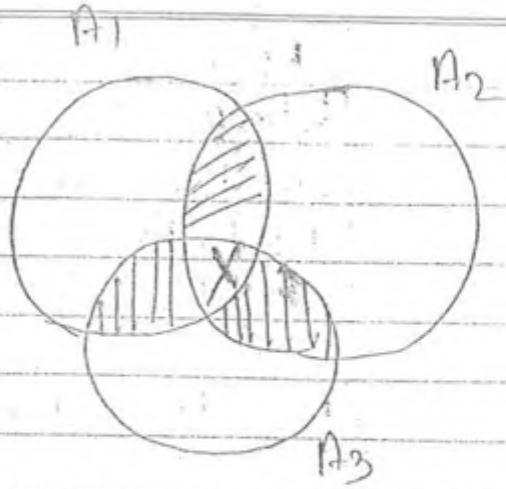
```
CREATE TABLE Student(
    Sid Varchar(10) Primary key,
    Sname Varchar(30) NOT NULL,
    PPno Varchar(15) UNIQUE,
    Lno Varchar(10) UNIQUE NOT NULL,
    DOB date NOT NULL,
    Fname Varchar(30) NOT NULL,
    Unique(DOB, Fname));
```

Superkey :-
Set of attributes used to different ate tuples of the relation.
* Difference between superkey & candidate key is that candidate key should be minimum.

Example :- Sid
Sid Sname
PPno
PPnoLno
} Superkeys

(Sname DOB) Not Super keys Because it is not uniquely determine the tuples of Relation.
* Every candidate key is Super key.
Reverse is not true.

e)



$$\underset{X}{2^{n-1}} + \underset{X}{2^{n-1}} + \underset{X}{2^{n-1}} - \underset{X}{2^{n-2}} - \underset{X}{2^{n-2}} - \underset{X}{2^{n-2}} + \underset{X}{2^{n-3}}$$

Note

Que 2) R is the relation set R(ABCDE) & Candidate key {A, BC}. Then how many super key are possible.

Sol^{n} — Because of A → $2^{5-1} = 2^4 = 16$

" " BC → $2^{5-2} = 2^3 = 8$

Common → $2^{5-3} = 2^2 = 4$

Total :- 16 + 8 − 4

= 20

Que 3) Relational schema with n attributes $(A_1, A_2, A_3, ---, A_n)$ then How many saper key are possible??

* Referencing Relation 1 → (Enrolled):

1) Insertion:- (May causes Referencial integrity violation)

⇒ if violation occurs insertion is prohibited

2) Deletion:- (No violation)

3) Updation:- (Updation of the referencing attribute may cause violation)

⇒ if violation occurs then updation is prohibited

OR

updation is prohibited if integrity violation occurs.

* from the child relation you can't do any modification in the parent relation

* Insertion from the Referenced Relation & Deletion from Referencing relation does not create any violation.

Note

Qu:)

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

A: Primary key

C: Foreign key References A with on delete cascade.
ON DELETE CASCADE.

To preserve referential integrity constraints what are the additional tuples deleted when the tuple (2,4) is deleted.

Sol^{n}:- (5,2), (7,2), (9,5) are deleted.

Pg 30)

Qu:) geq: greater than Or equal to

(x,y) ∈ geq only if y ≥ x

```
CREATE geq
(lb integer not null,
Ub integer not null,
primary key lb,
Foreign key (Ub) References geq on delete cascade);
```

Which of the following is possible if a tuple (x,y) is deleted?

a) a tuple (z,w) with z > y is deleted
b) a tuple (z,w) with z > x is deleted

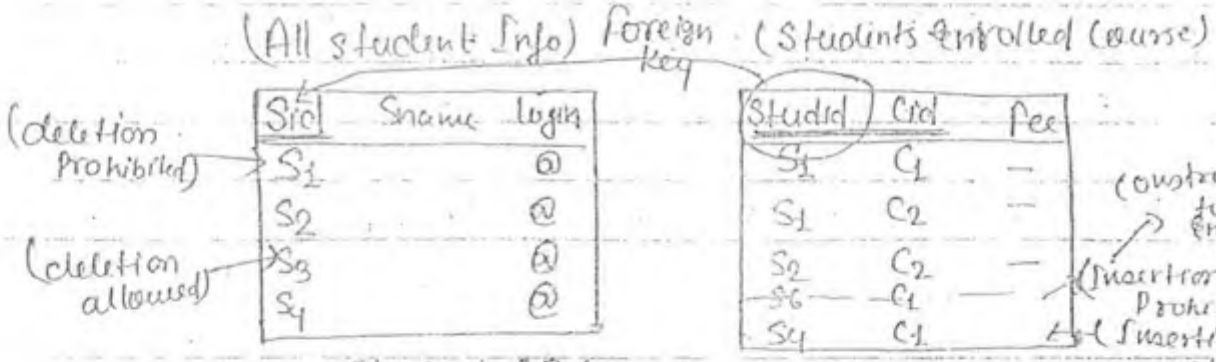
Soln:- if R(ABC)

A
B
C
AB
BC
AC
ABC

} If every attribute is candidate key then no. of super keys

$2^n - 1$ (we substract 1 because ϕ is not a super key)

FOREIGN KEY:-



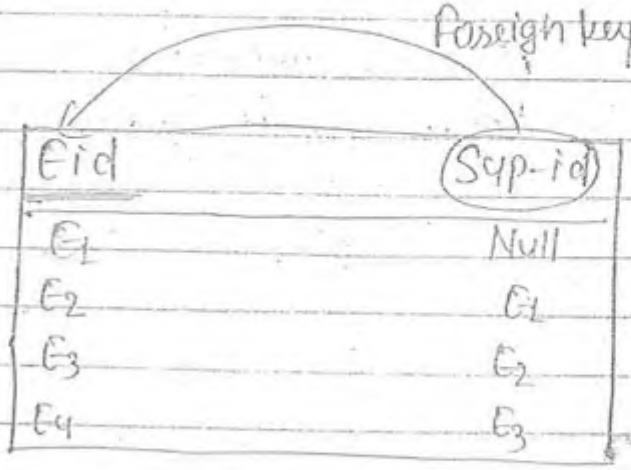
⇒ Foreign key is set of attributes references primary key or alternative key of the same table or some other Table.

Creation of Enrolled Table:-

```
CREATE(TABLE ENROLLED)
(Studid varchar (10),
cid varchar (10),
fee integer,
primary key (studid, cid),
Foreign key (Studid) References Stu
(Sid
```

ON DELETE NO ACTION

Example for foreign key reference the same table:-



Here Sup_id refrene the Eid means supid must belong to Eid i.e. no one member not belongs to Eid is in Supid. i.e. every numb of Sup-id must belongs to Eid.

* Refrential key is also called foreign key.

Referential Integsity Constraints:-

* Referenced Relation :- (Student)

1) Insertion :- No violation.

2) Deletion :- (May cause referential integrity violation)

* Referenced Table is also called Parent Table & Referencing Table is also called Child Table.

* Referential Integrity Constraints:-

DELETE

a) ON ~~Deletion~~ NO ACTION (default)
(deletion from Referenced Relation is restricted if Referencial integrity violation occurs.

b) ON DELETE CASCADE
(deletion happens from both tables).

* ON DELETE CASCADE never prohibited deletion

If foreign key is on the same table then

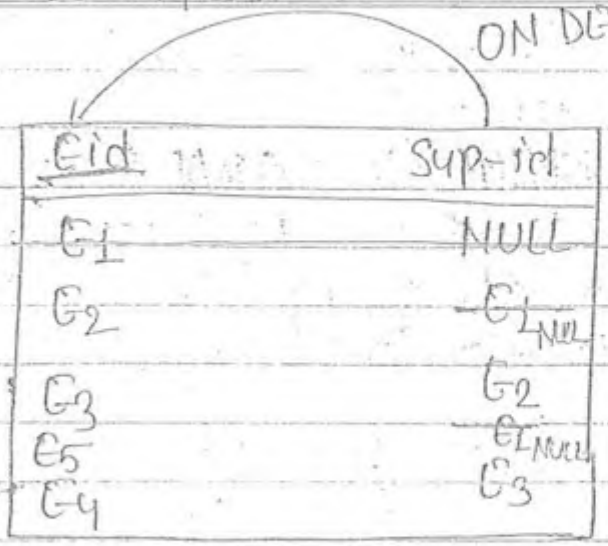
Eid	Sup-id
E1	Null
E2	E1
E3	E2
E4	E3

If we delete E1 using ON DELETE CASCADE it will delete more of the information from the table.

c) ON DELETE SET NULL
firstly try to set Null Value in the Referencing Relation foreign key attribute if success then allowed to delete from Referenced Relation. Otherwise not allowed).

* Referenced table ... referencing relation that putting null value in that relation is allowed if allowed then after delete the tuples from Referenced Relation otherwise not possible. Foreign key

Example:-



* If Foreign Key is not null attribute or primary key then ON DELETE SET NULL is equal to ON DELETE NO ACTION.

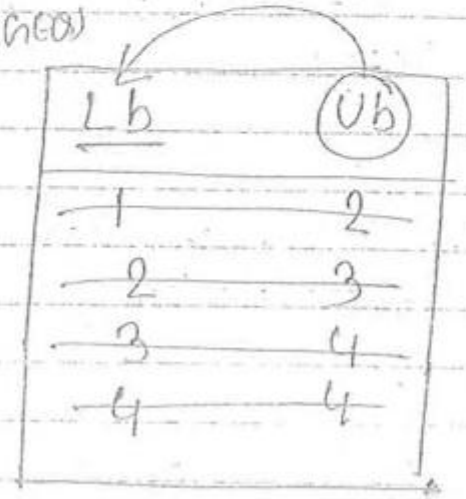
3) Updation:- (Referenced Attribute Set Updation) may cause Referencial integrity violation.

a) ON UPDATE NO ACTION (default) } Mostly used updation
b) ON UPDATE CASCADE
c) ON UPDATE SET NULL } → less used updation.

c) a tuple (z, w) with w<x is deleted.

d) Deletion of (x, y) is prohibited.

Sol^n 1-



(X Y) (Z, W)

*) (4,4) (3,4)
(2,3)
(1,2)

Then possibility is that w<z

- Requirements
- Design "UML" Model i.e. Diagramatic Representation of Entire S/W flow.

ER Model:- (Entity Relationship Model)

* High level DB Design..
* It is Diagramatic Representation of Entire DB Design.

Entity:-
An object which can be differentiate from other objects.

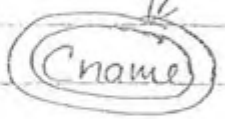
Entity Set:-
Set of Similar Entities.

* In ER Diagram Entity Set is represented by

Attribute:- 

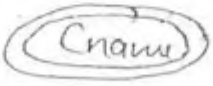
Key Attribute:- 

Multivalued Attribute:- MVA

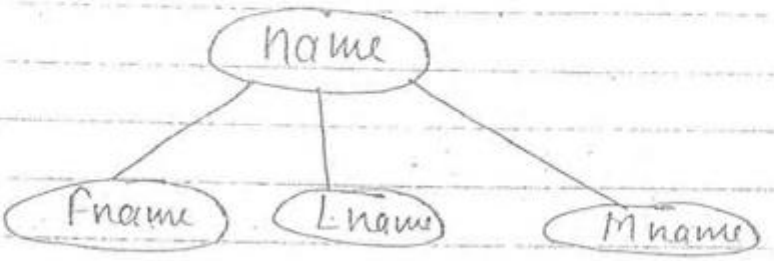


Example:-

Sid	Sname	Cname
S_1	A	C/C++
S_2	A	C++/Java

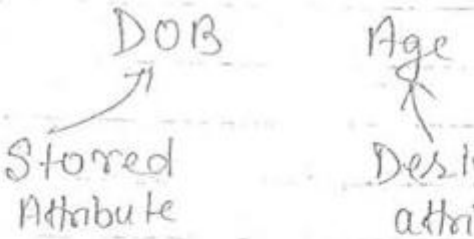
represented as 

Compound attribute (Composite Attribute)- Further subdivided in two or more attributes. Such kind of attribute is called compound attributes.



Desived Attribute-

Value of desived attribute desived from other stored attribute.



Age = Current Date – DOB

It is represented as

* ⟶ One

—— Many

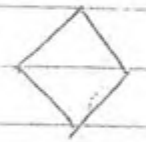
═══ Total participation

19/08/2011

Relationship Set:-

Used to relate two or more entity sets.

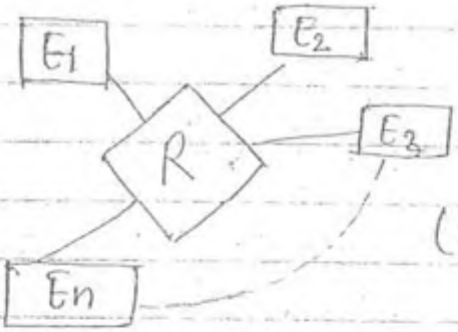
Symbol:-



represents the Relationship

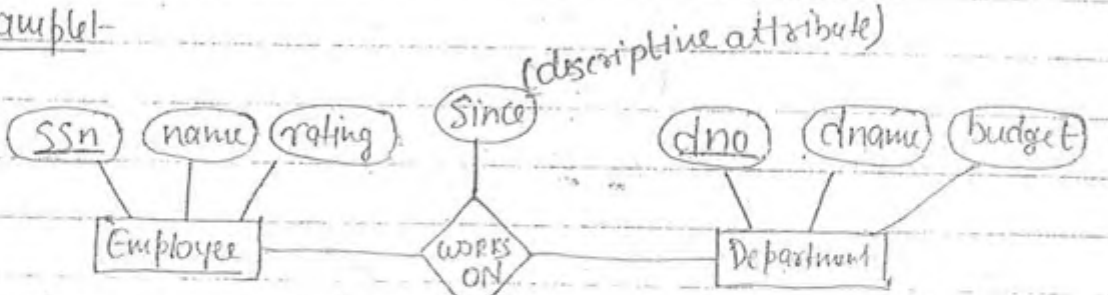
* Let R Relationship Set Relating $E_1, E_2, E_3, \ldots, E_n$ Entity Sets and $A_1, A_2, \ldots, A_n$ primary keys of $E_1, E_2, \ldots, E_n$ Entity sets Respectively.

Then Attributes of R is $A_1 \cup A_2 \cup \ldots A_n$.
i.e. primary keys of entity set relating to the relationship are attribute of Relationship set.



(n-ary Relationship set)

Example:-



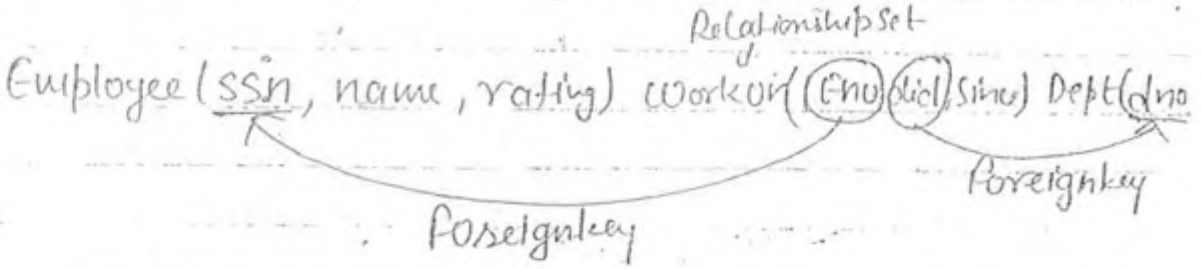
Employee(SSn, name, rating) worksON(Eno, did, since) Dept(dno, dname, budg

Constraints of Entity Set

- Primary key
- Alternative key
- NOT NULL / NULL

Relationship Set Constraints

- Primary key
- Alternative key
- NULL / NOT NULL
- FOREIGN KEY Constraints



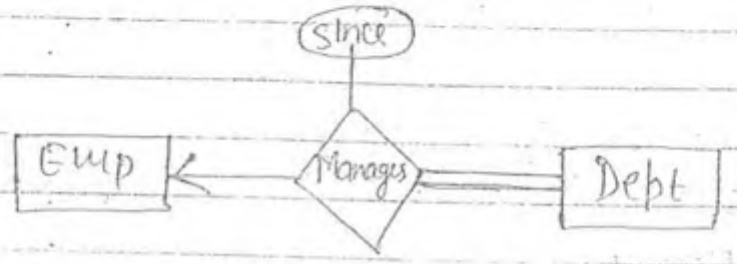
Creation of Table for WORKS_ON Relation :-

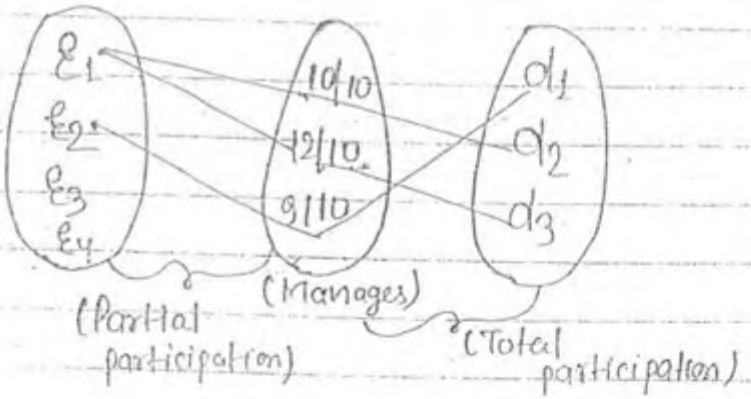
```
CREATE TABLE WORKS_ON
( Eno varchar(10),
  did varchar(15),
  Since date,
  primary key (_____ )
  FOREIGN KEY (ENO) Reference Employee ON DELETE CASCADE
  FOREIGN KEY (DID) Reference Dept ON DELETE CASCADE).
```

* Difference between relationship set & entity set is that relationship set requires foreign key constraints but Entity Set does not require foreign key constraints.

* Relationship Set Constraints:-

1) Participation:-





Partial Participation:- not every entity of the entity set is relating with Relationship Set is called Partial Participation.

Total Participation:- Every Entity of the entity set is Related with relationship Set is called Total Participation....

* Foreign key doesnot depends on cardinality only primary key does only depends on cardinality.

2) Cardinality :-

How one Entity Set Entities relat with other Entity set.

These are fours types Cardinality :-

a) One to One :-

We donot allow Eno & did as dupli i.e. both the entity should be key for One to One cardinality.

Eno	did
e_1	d_1
e_2	d_3

Candidate keys :
A, B

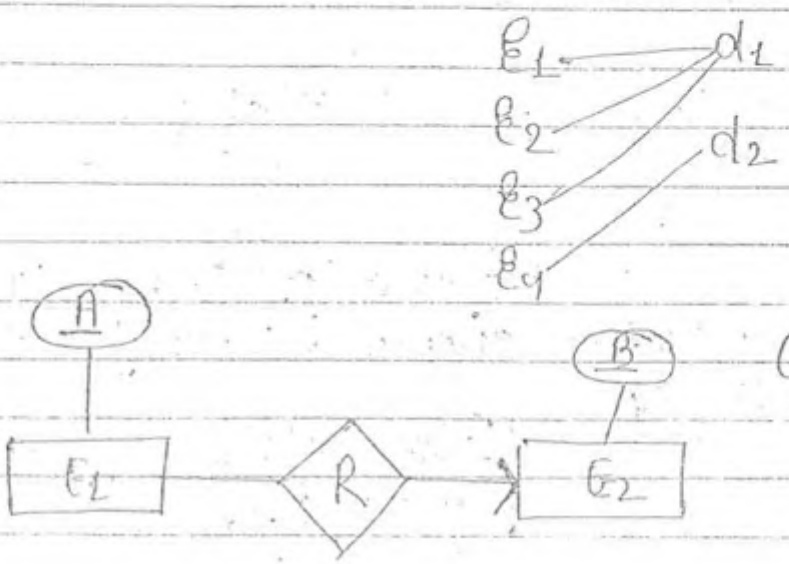
b) One to Many :-

In one to many cardinality "did" should be primary key.

Eno	did
e_1	d_2
e_2	d_3
e_3	d_1

Candidate key of R
B

c) Many to One:- In Many to One Eno should be primary key.



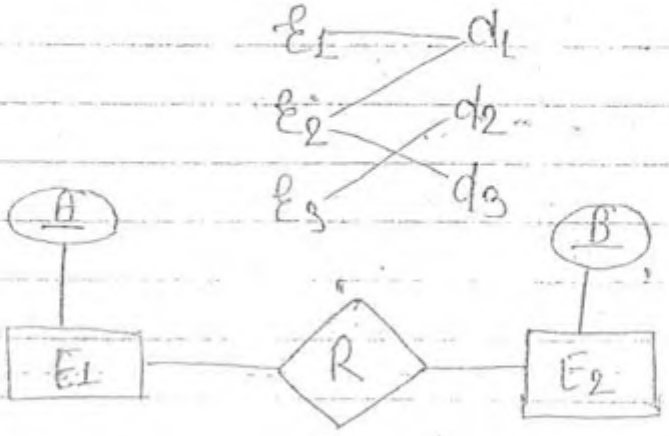
Candidate key

A

d) Many to Many:- (1:M && N:1)

It is combination of One to Many and Many to One.

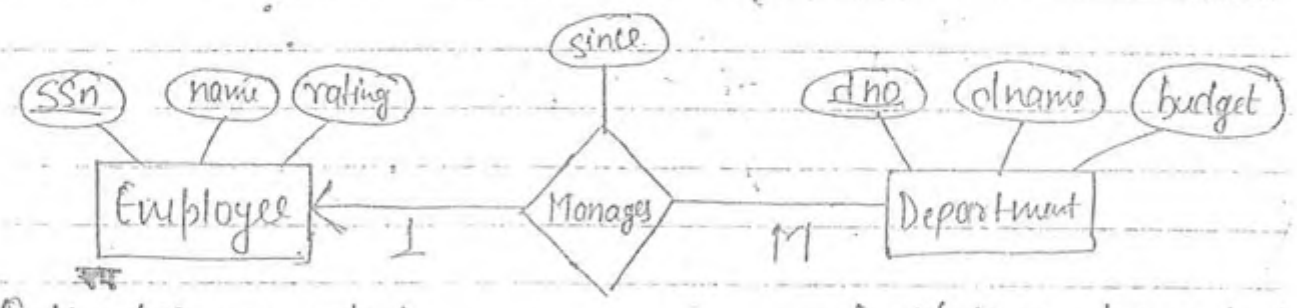
Here in both Eno & d duplication is possible.



Candidate key

AB

Example:-



Employee(SSn, name, rating)

Dept(dno, dname, budg

Manages(SSn, dno, since)

Primary key : dno

Foreign key : ssn
dno

After merging the manages table & Department Table:

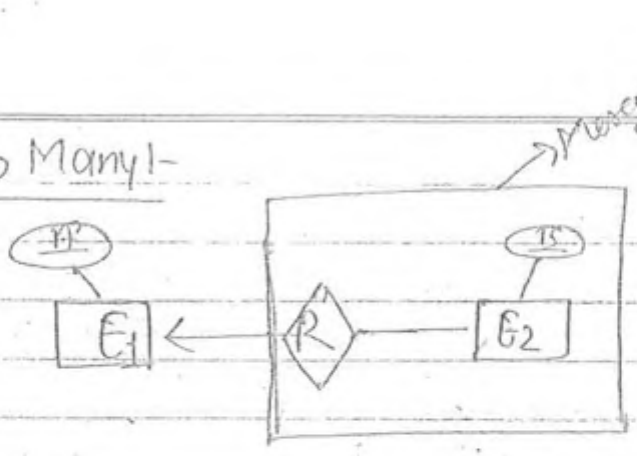
Dept Manages(dno, dname, budget, ssn, Sinc

foreign key (ssn) Reference Emp.

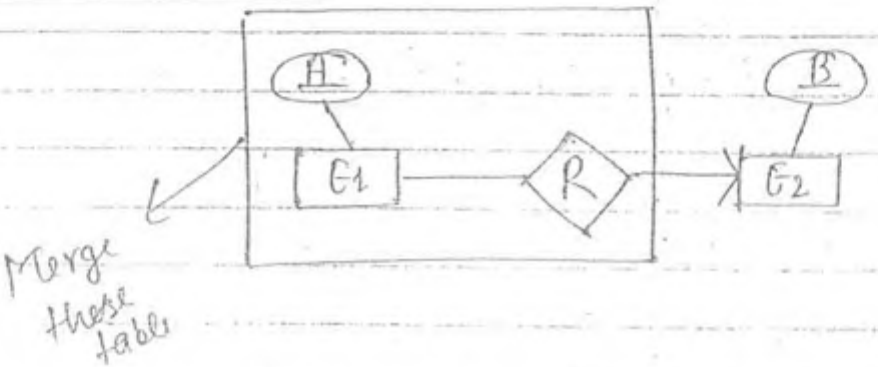
Minimization of ER Model:-

If foreign key attribut
set of Relationship set R is also primary key of
R and references Entity set E. Then R & E can be
Merge into Single Table.

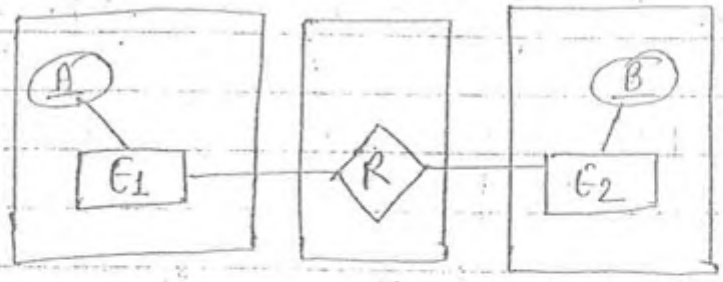
In One to Many!-



In Many to One!-



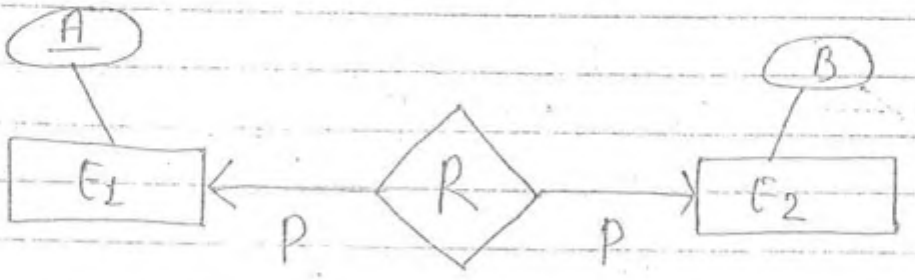
Many to Many!-



* In many to many we always need seperate table for Relationship Set R.

* Participation does not matter while minimizaing the one to many, many to one or many to many but it take matter while minimizaing one to one.

One to One :-



$E_1(\underline{A}, A_1, A_2)$ $R(A, B)$ $E_2(\underline{B}, B_1, B_2)$

A	A_1	A_2
1	a_1	a_2
2	a_1	a_2
3	a_3	a_4
4	a_4	a_4

Partial

A	B
1	13
2	12

Partial

B	B_1	B_2
11	b_1	b_1
12	b_1	b_2
13	b_1	b_2

$(E_1\ R)_1$

A	A_1	A_2	B
1	–	–	13
2	–	–	12
3	–	–	Null
4	–	–	Null

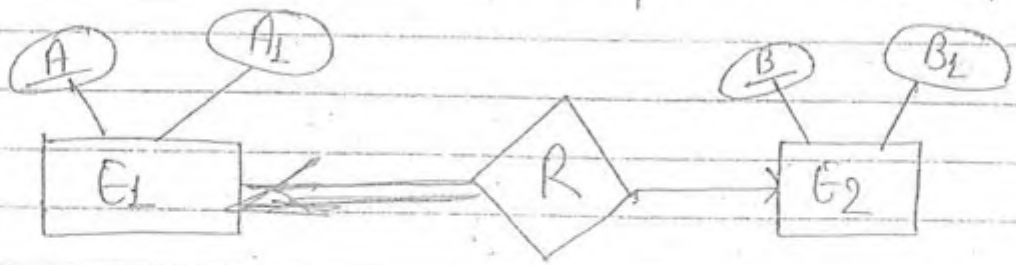
A: Primary key B: alternate key

$R\ E_2$

B	A	B_1	B_2
11	Null		
12	2		
13	1		

* If partial participation exist then we can minimize it either in $(E_1\ R)$ or $(R\ E_2)$.

Assume:- Total participation Exists b/w E1 & R.



E1

A	A1
1	a_1
2	a
3	a_2

R

A	B
3	13
2	11
1	12

E2

B	B1
11	b_1
12	b_1
13	b_2
14	b_2

A	A1	B	B1
1	a_1	12	b_1
2	a_1	11	b_1
3	a_2	13	b_2
Null	Null	14	b_2

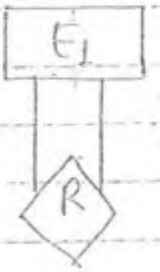
B: Primary key
A: Alternate key

* If at least one total participation is exist then it is required only one table ~~is required~~ for minimization i.e. if at least one or more then one total participation is exist between one to one cardinality then only one table is required after minimization.

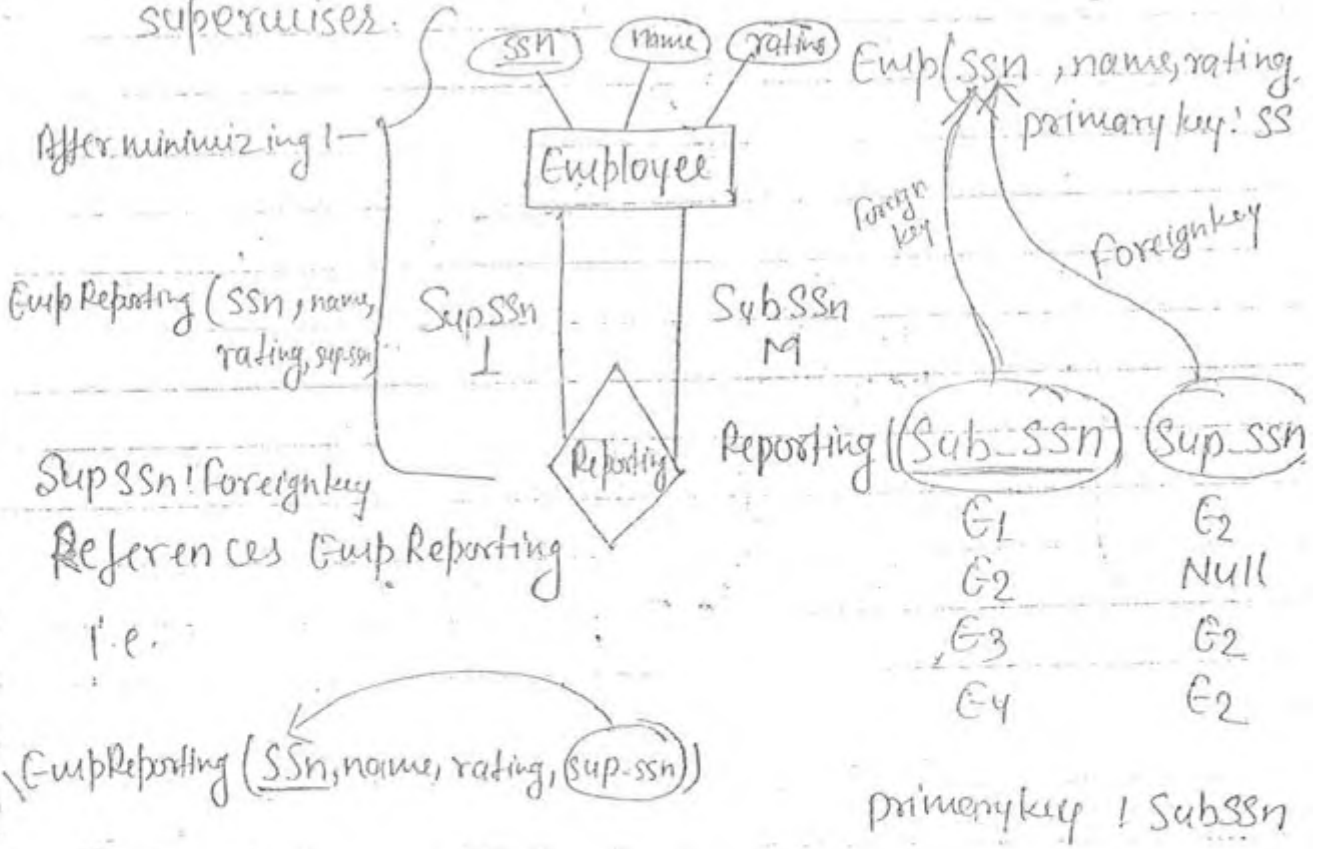
*

Self Referential Relationship Set:-

If relationship set reference same entity set then that set is called Self Referential Relationship Set.



Eg:- An Employee can supervise multiple sub-ordinates & each Sub-ordinate reporting to single supervisor.

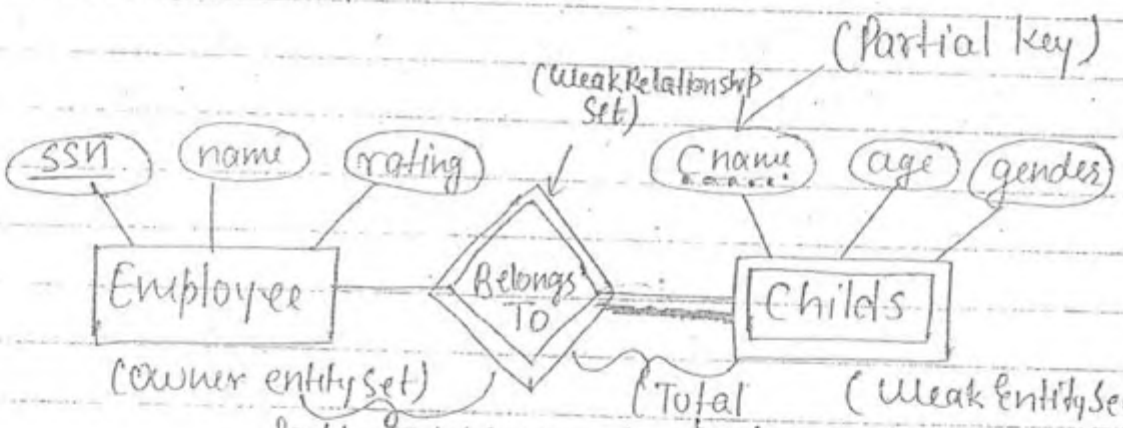


Weak Entity Set :-

1) Entity Set with no proper candidate key.

It is represented by :- 

2) Tuples in the weak entity set may not possible to differentiate using their's own attribute Set.



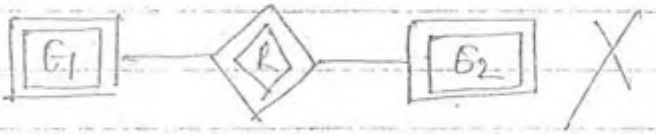
SSN
e_1
e_2
e_3
e_4

SSN	Cname
e_1	A
e_2	A
e_3	B

Cname	Age	gender
A	20	M
A	25	M
B	20	F

3) For any weak entity set there should be corresponding corresponding OWNER ENTITY SET.

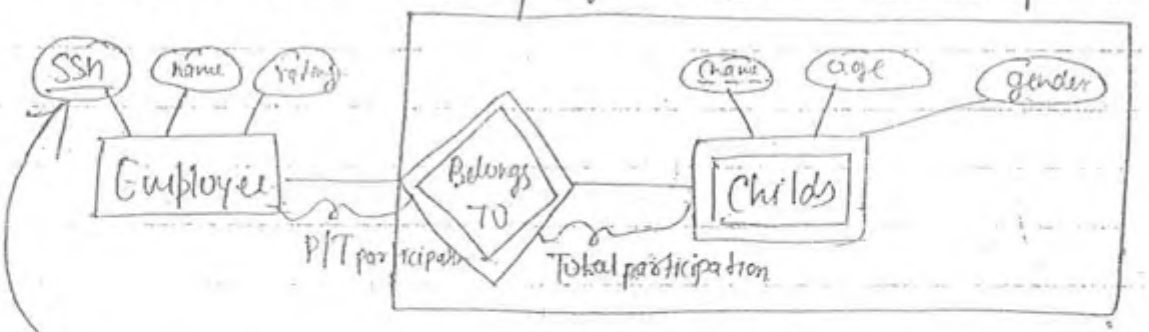
4) Two weak Entity not possible to relate with Relationshipset.



5) Participation b/w weak Entity set with Relationship set always Total.

6) Cardinality b/w owner entity & weak entity can be one to many or many to many.

7) DataBase table should be always combines with weak entity & weak relationship Set.



Childbelongs

SSn	Cname	age	gender
E_1	A	20	M
E_2	A	25	M
E_1	B	20	F

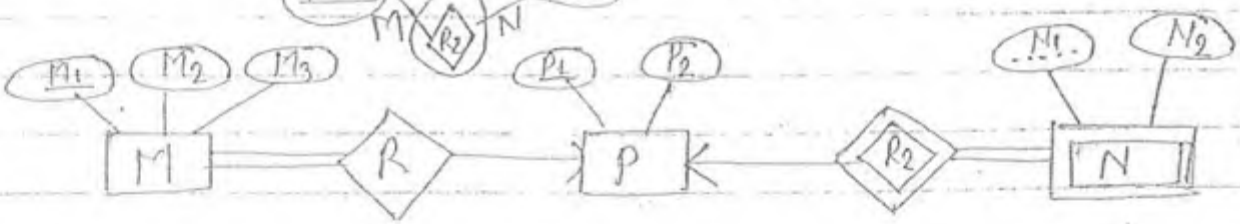
3) Primary key owner Entity set primary key combines with weak Entity Set partial key.

* max no. of Tables or min no. of tables ⇒ ?

Que) E_1 & E_2 are Entity Sets
R_1 & R_2 Relationship sets
Relating b/w E_1 & E_2 with 1:M & M:N cardinality how many minimum no. of tables required to minimize the following ER diagram.

Sol^{n}:-

(E_1), (R_2), $(R_1 E_2)$



1) minimum no. of tables needed to represent M, N, P, R_1, R_2

2) Which of the following is a correct attribute set for one of the table for the correct answer to the above Question?

a) $\{M_1, M_2, M_3, P_1\}$ b) $\{M_1, P_1, N_1, N_2\}$

c) $\{M_1, P_1, N_1\}$ d) $\{M_1, P_1\}$

3) Deletion Anamoly:-

Deletion of some data forced to loose some other useful data.

Example:- Because of deletion of S_4 we lose course C_3 details.

=> The only way to elimination of Redundecy is decomposition of the Relation:-

Decomposition of the Relation:-

Splitting the Relation into two or more sub-relation.

R

Sid	Sname	Cid	Cname	Fee
S_1	A	C_1	C	5K
S_1	A	C_2	C	7K
S_2	B	C_1	C	5K
S_3	B	C_1	C	5K
S_4	B	C_3	C++	10K

Cid → Cname
Sid → Sname
Cid → Fee
Also
SidCid → Sname
→ Cname
→ Fee

R

R1

Sid	Sname	Cid
S_1	A	C_1
S_1	A	C_2
S_2	B	C_1
S_3	B	C_1
S_4	B	C_3

(Sid Cid) primary key

R2

Cid	Cname	Fee
C_1	C	5K
C_2	C	7K
C_3	C++	10K

Cid: Primary key

* To put relationship between two decomposed tables
If their is a foreign key between these relation then
their is no any problem & if their is no any key like

this then put common key or attribute in both
the table. to put relationship between two decompose
table. It is just work as foreign key.

Properties of Decomposition:-

1) Loss less Join Decomposition:-

Join between the
subrelation should be equal to exactly same Relat
on R.

$R_1 \bowtie R_2 \equiv R$ (lossless Join)

$R_1 \bowtie R_2 \supset R$ (Lossy Join)

If R

S_1	C_1
S_1	C_2

$R_1 \bowtie R_2$

S_1	C_1
S_1	C_2
S_1	C_3

→ extra tuple

i.e. in lossy join we lose the original de
of the relation.

2) Dependency Preserving Decomposition:-

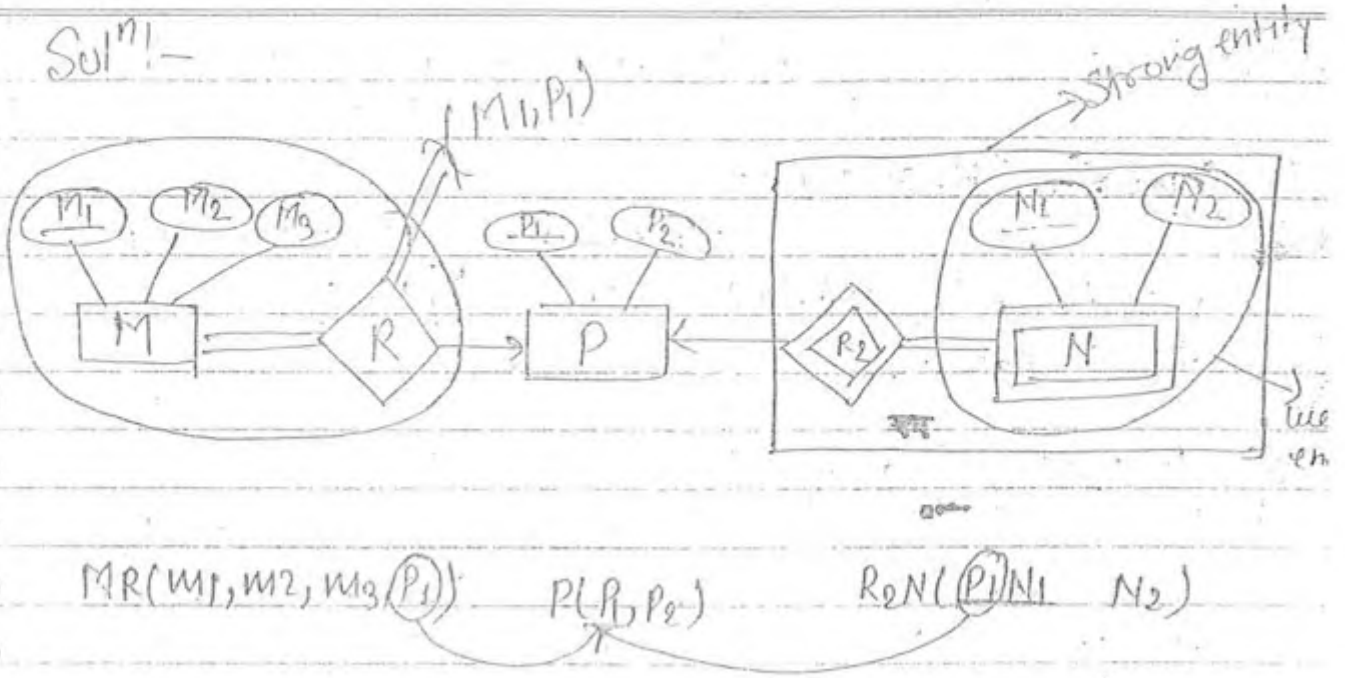
Because
of decomposition, relation should not lose any depen-
dency.

Example:-

cid → cname
Sid → Sname
Cid → fee

} All of the dependen
of relation also exis
after decomposition
into two tables.

Sol^n :-



MR(M1, M2, M3 (P1)) P(P1, P2) R2N(P1 N1 N2)

NORMALIZATION :- (Schema Refinement)

Used to eliminate / Reduce the redundancy

Redundancy :- Duplicate copy of the same data stored in multiple locations then it is called redundancy.

Sid	Sname	Cid (course id)	Cname (course name)	Fee
S1	A	C1	C	5K
S1	A	C2	C	7K
S2	B	C1	C	5K
S3	B	C1	C	5K
S4	B	C3	C++	10K

↳ primary key : (Sid Cid)

⇒ Problems becoz of Redundency :- (Anamalies)

1) Updation Anamaly :-

Updation of single value results all duplicate copies updation which too costly.

2) Insestion Anamoly :-

Because of insestion of some data forced to insert some dummy data.

Sid	Sname	Cid	Cname	Fee
S_1	A	C_1	C	5K
S_1	A	C_2	C	7K
S_2	B	C_1	C	5K
S_3	B	C_1	C	5K
S_4	B	C_3	C++	10K
XX	XX	C_4	Java	15K

Dummy data

primary key (Sid Cid)

If we want to insert (C_4, Java, 15K) course which is not taken by any student then we put some dummy data because of primary key where null values are not allowed

If we

```
Select Count (Distinct Sid)
From Student
```

⇓

(5) (Inconsistent because of dummy data)

Functional Dependency:-

Let R be the relational schema and X, Y be the attribute sets over R and t_1, t_2 are any two tuples.

$X \rightarrow Y$ exists in R

only if $t_1.X = t_2.X$ Then $t_1.Y = t_2.Y$

Example:- If Cid $\rightarrow$ Cname

	X	Y
t_1	C_1	C
t_2	C_1	C

	X	Y
t_1	X_1	Y_1
t_2	X_2	Y_1

$X \rightarrow Y$

	X	Y
t_1	X_1	–
t_2	X_2	–

$X \rightarrow Y$ (X is can key)

	X	Y
t_1	X_1	Y_1
t_2	X_1	Y_2

$X \rightarrow Y$ not Exists in R

Dependency is of Two types:-

1) Trivial dependency:-

Let R be the Relational Schema X, Y are attributes Sets over R if $X \supseteq Y$ Then $X \rightarrow Y$ is trivial.

Example:-

Trivial Functional dependency:

- Sid $\rightarrow$ Sid
- Sid Sname $\rightarrow$ Sid
- Sid Sname $\rightarrow$ Sid Sname
- Sid Sname $\rightarrow$ Sname

exist according to the definition we check only attr
is subset of any set then always trivial dependency
occurs. Do not check for the data of the table.

2) Non Trivial Functional dependency:- (X & Y attr
Se

X → Y is non trivial only if
a) X ∩ Y = ϕ
and b) X → Y should satisfy functional
dependency definition.

Example:-

Cid → Cname
Sid → Sname
SidCid → Sname
} Non trivial Junctio
dependency

3) Semi-non trivial Functional dependency:-

Combin
Of trivial & non trivial Jun dependency is call
Semi-non trivial Jun dependency.

Example:-

Sid → Sid Sname
→ Sid → Sid (Trivial)
→ Sid → Sname (Non triv

Properties of Jun dependency:-

⇒ Reflexivity:-
if X ⊇ Y Then X → Y is reflexive FD

* Trivial Jun dependency are reflexive Jun dependency.

⇒ Transitivity:-

if $X \rightarrow Y$ & $Y \rightarrow Z$ Then $X \rightarrow Z$

⇒ Argumentation:-

if $X \rightarrow Y$ then $XZ \rightarrow YZ$

Example:- if Sid → Sname
then Sid Cid → Sname Cid

⇒ Union of Functional Dependency:-

if $X \rightarrow Y$ & $X \rightarrow Z$

Then $X \rightarrow YZ$

but converse is not true i.e. if $A \rightarrow C$, $B \rightarrow C$ } not allow
then $AB \rightarrow C$

⇒ Splitting of Functional Dependency:-

if $X \rightarrow YZ$ Then $X \rightarrow Y$, $X \rightarrow Z$

* Commutative properties not hold by fun dependency
$X \rightarrow Y$ Then $Y \rightarrow X$ is not possible
* Associative properties is also not hold by fun dependency
i.e. $A \rightarrow BC \neq AB \rightarrow C$

Application of Fun dependency:-

1) Identify the Redundency in the Relation

2) Identify the candidate key of the Relation.

* $X \to Y$ non trivial dependency with X is not candidate key, then we can say that attribute set (X, Y) suffering from Redundancy.

Example:- If R(ABC) & if A: Candidatekey & "$B \to C$"

A	B	C
1	2	a
2	2	a
3	2	a
4	3	b
5	3	b

Que)

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

identify non trivial dependency.

Sol^n:- Using attributes A, B, C no. of non trivial dependency

- ✓ $A \to B$, ✗ $A \to C$, ✗ $A \to BC$
- ✓ $B \to A$, ✗ $B \to C$, ✗ $B \to AC$
- ✓ $C \to A$, ✓ $C \to B$, ✓ $C \to AB$
- ✗ $AB \to C$, ✓ $BC \to A$, ✓ $AC \to B$

$AB \not\to C$, $BC \not\to B$, $AC \not\to B$

Que
If $R(A_1, A_2, \ldots A_n)$ identify possible non-trivial functional dependency.

Attribute Closure (X^+) :- Set of attribute functionally determined by 'X'.

Ex:- $R(ABCD)$
$\{A \to B, B \to C, C \to D\}$

$A^+ = \{A, B, C, D\}$ means $A \to ABCD$
$A \to A$
$A \to B$ } $A \to BC$ } $A \to$
$A \to C$
$A \to D$

$B^+ = \{B, C, D\}$
$C^+ = \{C, D\}$
$D^+ = \{D\}$

note 2006
Que $\{AB \to CD; AF \to D, DE \to F, C \to G, F \to E, G \to A\}$
Option is false :-
a) $(CF)^+ = \{ACDEFG\}$ $CF^+ = \{CFGEAD\}$
b) $(BG)^+ = \{ABCDG\}$ $BG^+ = \{BGACD\}$
c) $(AF)^+ = \{ACDEFG\}$ $AF^+ = \{AFED\}$
d) $(AB)^+ = \{ACDFG\}$ $AB^+ = \{ABCDG\}$

Super key :-

Let R be the Relational schema and X be the set of attributes over R.
X is said to be Super Key of R only if X^+ should determine all the attributes of the Relation R.

* X is said to be minimal super key (candidate key) only if proper subset of X should not be Super key.

* A super key may be a candidate key Or may not be candidate key.

if $(AB)^+ = \{$All the table attributes of R$\}$

AB : Super key.

if $\begin{cases} A^+ = \text{Not determines all attributes of R} \\ B^+ = \text{Not determines all attributes of R} \end{cases}$ ✓ Not Superkey

Then AB : minimal Super key (Candidate key)

Que) R(ABCD)
{A→B, B→C, C→D}
Identify Candidate key

Soln:- $(A)^+ = \{A\ B\ C\ D\}$
∴ A: Superkey ∴ {A}: Candidate key

* If super key is also of only one attribute then it is also candidate key or minimal super key.

Que) R(ABCDE)
{AB→C, C→D, B→E}

Soln:- $(AB)^+ = \{A\ B\ C\ D\ E\}$
⇒ AB: Super key
Check for minimal super key:-
$A^+ = \{A\}$ not superkey
$B^+ = \{B\ E\}$ not superkey
⇒ AB: Candidate key

Que) R(ABCDE)
{AB→C, C→D, B→EA}
Soln:- $AB^+ = \{A\ B\ C\ D\ E\}$
AB: Super key
Check for minimal superkey
$A^+ = \{A\}$
$B^+ = \{B\ E\ A\ C\ D\}$
⇒ AB Superkey but not candidate key
{B}: Candidate key

Que) R(ABCDE)
{AB→C, C→D}

Sol^{n}:- $(AB)^+ = ABCD$

If some attribute is not part of functional dependency then that attribute must in the candidate key. Here E is not determined by any attribut so, E must be in candidate key.

$(ABE)^+ = ABCDE$

(ABE): Candidate Key

Que) R(ABCDEF)
{A → BCDEF, BC → ADEF, B → F, D → E}

Sol^{n}:-

$A^+ = \{ABCDEF\}$
$B^+ = \{ABCDEF\}$
$B^+ = BF$ ✗ not superkey
$C^+ = C$ ✗ not superkey

Candidate key = {A, BC}

* if $X \longrightarrow A$ (Prime attribute)
i.e. A determine all the attribute)
& if A is a super key
Then X is also super key.

* if Prime attribute is present in the right side then more than one candidate key may be possible. Not always but may or may not be possible.

(Que) $R(ABCD)$
$\{A \to B, B \to C, C \to D, D \to A\}$

Soln:-
$A^+ = ABCD$
$D^+ = ABCD$
$C^+ = ABCD$
$B^+ = ABCD$

Candidate Key = $\{A, B, C, D\}$

(Que) $R(ABCDEF)$
$\{AB \to C, C \to D, D \to E, E \to F, F \to B\}$

$(AB)^+ = \{ABCDEF\}$ AB is super key
$A^+ = \{A\}$ ×
$B^+ = \{B\}$ ×

$(AE)^+ = \{ACDEFB\}$ super key
$A^+ = \{A\}$ ×
$E^+ = \{EFB\}$ ×

Candidate key = $\{AB, AE, AD, AC\}$

$(AE)^+ = \{AEFBCD\}$
$E^+ = EFB$ ×

$(AD)^+ = \{ABCDEF\}$
$D^+ = DEFBC$ ×

$(AC)^+ = \{ABCDEF\}$ $C^+ = CDEFB$ ×

Que) R(A B C D E F G H I J)

{ AB → C
A → DE
B → F
F → GH
D → IJ }

$(AB)^+$ = { A B C D E F G H I J }

A^+ = { A D E I J }

B^+ = { B F G H }

Candidate Key = { A B }

Que) R(A B D L P T)

{ B → PT, T → L, A → D }

B^+ = (B P T L)

$(AB)^+$ = (B P T L A D)

A^+ = { A D }

B^+ = { B P T L }

Candidate Key = { A B }

Que) R(A B C D E F G H)

{ AB → C, AC → B, AD → E, B → D, BC → A, E → G }

$(ABFH)^+ = \{ABCDEGFH\}$

$(BCFH)^+ = \{ABCDEFGH\}$

$(ACFH)^+ = \{ABCDEFGH\}$

Candidate Keys = {ABFH, BCFH, ACFH}

Que) R(ABCDE)

$\{A \to BC, CD \to E, B \to D, E \to A\}$

$(A)^+ = \{ABCDE\}$

$(E)^+ = \{ABCDE\}$

$(CD)^+ = \{ABCDE\}$

$(BC)^+ = \{ABCDE\}$

$(AC)^+ = $ _____

Candidate Keys = {A, E, CD, BC}

20/08/11

Que1) R(ABCDEF)
{AB→C, C→DE, E→F, F→A}

Que2) R(ABCDEF)
{AB→C, C→D, D→E, E→A, D→B}

Soln 1)

$AB^+ = \{ABCDEF\}$
$A^+ = \{A\}$
$B^+ = \{B\}$
$FB^+ = \{ABCDEF\}$ $F^+ = \{AF\}$
$EB^+ = \{ABCDEF\}$ $E^+ = \{EFA\}$
$CB^+ = \{ABCDEF\}$ $C^+ = \{ACDEF\}$

Candidate Key :- (AB, FB, EB, CB)

Soln 2)

$ABF^+ = \{ABCDEF\}$
$ADF^+ = \{ABCDEF\}$
$BDF^+ = \{ABCDEF\}$
$(BF)^+ = BF$
$(DF)^+ = DFEABC$
$(CF)^+ = ABCDEF$
$(ABF)^+ = ABCDEF$
$(EBF)^+ = ABCDEF$
$E^+ = EA$
$B^+ = B$
$(DBF)^+ = ABCDEF$

Candidate Key = {ABF, BEF, DF, CF}

Ques) R(ABCDE)

{AB→C, C→D, D→E, E→A, D→B}

AB^+ = {ABCDE}

A^+ = {A}

B^+ = {B}

EB^+ = {ABCDE}

E^+ = {EA}

XDB^+ = {ABCDE}

✓ D^+ = {DEABC}

AB^+ = {ABCDE}

XAD^+ = {ABCDE}

D^+ = {DEBAC}

C^+ = {CDEAB}

Candidate key :- {AB, EB, D, C}

* If R(ABC) with No non trivial Functional dependency

⇒ ABC : Candidate Key

* If R(ABCD)

{AB→C, DE→F, C→D, E→F} DE→A

C→D, C→E

then remaining Relation

{AB→C, C→D}

⇒ In the above we remove those functional dependency whe attribute not belongs to the relation schema attribu i.e. {DE→F, C→E, E→F, DE→C}

Properties of functional dependency set:-

1) Membership test:-

Example:- If $\{AB \to C, C \to D\}$

Check $B \to C$ is member of this set

Check $AB \to D$

Soln:- $B^+ = B$

i.e. B not determine C. So, it is not membership set

Check for $AB \to D$ is membership set

$AB^+ = ABCD$

i.e. AB determine D. So, it is member of the set.

functional dependency

⇒ Let F be the functional dependency and $X \to Y$ and F.D.

$X \to Y$ logically ($\models$) in F only if closure set of X should consists Y using FD which is in F.

2) Functional dependency set closure (F^+):-

Set of all functional Dependency that are belongs to given FD set.

Example:- $F = \{A \to B, B \to C\}$

$F^+ = \{$ $A \to A$, $A \to B$, $B \to C$, $A \to C$,
$B \to B$, $AC \to BC$, $AB \to AC$, $AB \to CB$,
$AB \to A$, $AC \to B$,
$AB \to B$, $AC \to C$,
$AB \to AB$ $\}$

F^+ contain:-

(0) $\Leftarrow \phi^+ = \phi$

(1) $\Leftarrow$
- $A^+ = ABC$
- $B^+ = BC$
- $C^+ = C$

(2) $\Leftarrow$
- $(AB)^+ = ABC$
- $(BC)^+ = BC$
- $(AC)^+ = ABC$

(3) $\Leftarrow (ABC)^+ = ABC$

$\phi \to \phi$				(1)
$A \to \phi$	$A \to B$	$A \to AB$	$A \to BC$	(8)
$A \to A$	$A \to C$	$A \to AC$	$A \to ABC$	
$B \to \phi$	$B \to B$	$B \to C$	$B \to BC$	(4)
$C \to \phi$	$C \to C$			(2)
$AB \to \phi$			$AB \to ABC$	(3)
$BC \to \phi$			$BC \to BC$	(4)
$AC \to \phi$			$AC \to ABC$	(8)
$ABC \to \phi$			$ABC \to ABC$	(8)

48

Here F^+ logically equivalent to F

R(AB)

F = { A→B, B→A }

Identify Total dependencies in F+

Soln- $\phi^+ = \phi$ $\phi \to \phi$ — (1)

$A^+ \Rightarrow AB$ $A \to \phi$ $A \to A$ $A \to B$ $A \to AB$ — (4)

$B^+ = AB$ $B \to \phi$ $B \to A$ $B \to B$ $B \to AB$. — (4)

$AB^+ = AB$ $AB \to \phi$ $AB \to AB$ $AB \to A$ $AB \to B$ — (4).

13

3) Equality b/w FD Sets :-

a) F & H said to be Equal only if $F^+ = H^+$

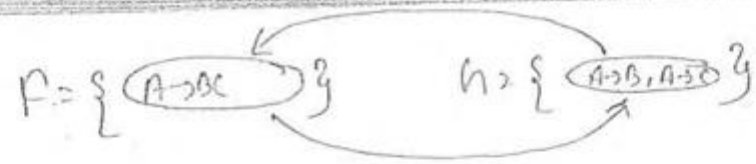
b) F & H said to be Equal only if

[1] F Covers H :-

All FD's of H Set should be logically implied in F Set.

[2] H covers F :-

All FD's of F Set should be logically implied in H.



F Covers G	G Covers F	
T	T	⇒ F = G
T	F	⇒ F ⊃ G
F	T	⇒ G ⊃ F
F	F	⇒ F ≠ G (No comparison val

Que) $F = \{A \to C, AC \to D, E \to AD, E \to H\}$

$G = \{A \to CD, E \to AH\}$

F → A
F → H
A → C
A → D

F COVER G is true $A^+ = ACD$, $E^+ = EADH$

G COVER F is true $AC^+ = ACD$, $A^+ = ACD$, $E^+ = EADH$

Que) $F = \{AB \to CD, B \to C, C \to D\}$

$G = \{AB \to C, AB \to D, C \to D\}$

F COVER G is true

G COVER F is false

⇒ F ⊃ G

Properties of Decomposition:-

1) Lossless Join Decomposition:-

Let R be the Relational Schema decomposed into $R_1, R_2, R_3, \ldots$

In General: $R_1 \bowtie R_2 \bowtie \ldots \bowtie R_n \supseteq R$

If $R_1 \bowtie R_2 \bowtie \ldots \bowtie R_n = R$

Then lossless join decomp

If $R_1 \bowtie R_2 \bowtie \ldots \bowtie R_n \supset R$

Then Lossy join decomposition

Example:- R(ABC) → decomposed into $R_1(AB)$ $R_2(BC)$

R

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

R_1

A	B
1	1
2	1
3	2

R_2

B	C
1	2
1	1
2	1

$R_1 = \Pi_{A,B}(R)$ $\quad R_2 = \Pi_{B,C}(R)$

Π : Projection
To Select attribute of Relation

$\bowtie$: Join :- Cross Product followed by selection of the equality b/w common attribute and then Projects distinct columns.

σ : Selection operator
Select the tuples those are satisfied predicate cond

SECOND NORMAL FORM :-

Let R be the Relational Schema is in 2NF only if

1) R should be in 1NF.
2) R should not consist any partial dependencies

Partial Dependencies :-

Let R be the Relational Schema
X, Y, A attribute sets over R

if $Y \rightarrow A$ exists in R then $Y \rightarrow A$ called partial Dependency

(or)

X: any candidate key of R
Y: Proper Subset of any candidate key
A: Non-Prime attribute

$Y \rightarrow A$: Nontrivial FD is partial dependency : only if

(1) Y is proper Subset of Candidate key.
and
(2) A is Non-Key or Non Prime attribute.

To check the relation is in 2NF or not, use the above condition and checking that above Relation is in 2NF or not.

* If right side of dependency candidate key is not possible.
i.e.
if candidate key is present in right side of dependency then redu
key is not possible.

[2] Non key attribute → Non key attribute (3NF)
used to eliminate this type of redundancy

Example- R(A B C D E) {AB→C, C→D, B→E}
[2] case [1] case

Non key → CK ✗

R(ABCD)

Example- {A→B, B→C, C→D, D→A}

candidate key {A, B, C, D}

This dependency is not possible. Non key → Proper subset of candidate key ✗

Example- R(ABCDE)
{AB→C, C→D, B→E, C→A}

C.K {AB, CB}

[3] Proper subset of one Candidate key → Proper subset of other Candidate key (BCNF)
used to eliminate this type of redundancy

Example- R(ABCD) {AB→CD, D→A}
[3] case

Relation Consists Redundancy becoz of single valued dependency:-

If $X \rightarrow Y$: Non Trivial functional dependency with X is not super key. Then redundancy always possible.

Example:-

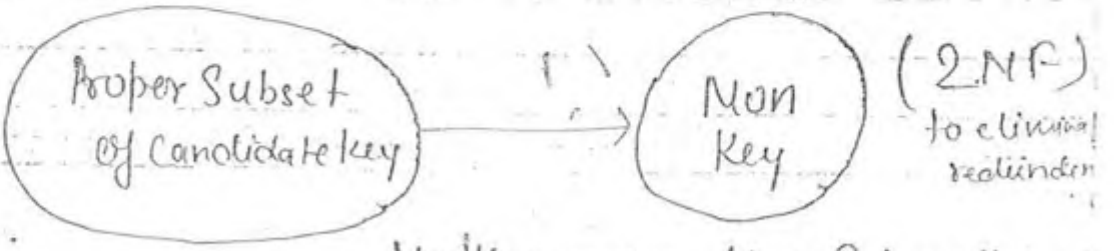
	X	Y
Not super key	x_1	y_1
	x_2	y_1
	x_1	y_1

Or in the another way or contradiction of the above statement:-

(if any Non-trivial $X \rightarrow Y$ exists in R Then X should be Super key.
Then R is free from Redundancy becoz of single value dependencies)

Non-Trivial FD which results Redundancy:-

[1]



Nonkey means Non Prime attribute.

By Converting the above relation in 1NF: We get

Sid	Sname	Cname
S1	A	DB
S1	A	DM
S2	B	DM
S3	B	BM
S3	A	DB

Redundancy

(Sid → Sname
Sid Cname → Sname

(Sid Cname) : Primary key
No multivalued Attribute (1NF)

⇒ Problem of 1NF:-
It Results high redundancy.

Qu) How to convert Relation into 1NF if
R(A, B C, D)
A: Primary key
C, D: Multivalued Attributes
Not in 1NF

Sol^n:-

R(ACD, B) : (1NF Relation)

primary key = ACD

By only changing the key we can convert relation into 1NF

Dependency:- A → B (Before the conversion 1NF relation)

ACD → B (When relation is in 1NF)

21/08/11

NORMAL FORMS:-

(Used to Eliminate / reduce Redunda-

(Set of Rules to perform decomposition)

Types of Normal Forms:-

Single valued dependency { 1NF, 2NF, 3NF, BCNF → 0% Redundancy becoz of single valued depend; May exist Redundancy becoz of Multivalued dependency ($X \rightarrow\rightarrow Y$)

Eliminate Redundancy becoz of MVD { 4NF

FIRST NORMAL FORM:-

Relation R is in 1NF only if No multivalued attribute Exists over R.

(Or)

only single valued attributes Exists over R.

Example:-

Sid	Sname	Cname (Multivalued attribute)
S1	A	DB/DM
S2	B	DM/BM
S3	A	DB

(NOT in 1NF)

Primary key : Sid

Check for correct decomposition

Que) R(A B C D E F G H I J)

{AB→C, A→DE, B→F, F→GH, D→IJ}

D1 = { ABC, ADE, BF, FGH, DIJ }
AB→C A→DE B→F F→GH D→IJ

D2 = { ABCDE, BFGH, DIJ }
AB→C, A→DE B→F, F→GH D→IJ

D3 = { ABCD, DE, BF, FGH, DIJ }

These two decomp preserv

Ans:- Decomposition is proper only if it satisfy lossless join + Dependency preserving.

4) Union (U) :- $R1 \cup R2 = R2 \cup R1$ ✓

5) Intersection ($\cap$) :- ✓

$R1 \cap R2$

6) Set Difference (−) :- ✗

7) / (Division) :- ✗

$A(x,y) / B(y) \neq B(y) / A(x,y)$

8) Natural Join ($\bowtie$) = ✓

Query :- Assume Supplier relation consist 200 tuple & catalog relation consist 100 tuple; How many maximum no. of tuples resulted by Supplier $\bowtie$ catalog.

Supplier

Sid
S1
S2
S3
⋮
S200

200

$\bowtie$

Catalog

Sid	Pid
S1	S1
S1	S2
S1	S3
⋮	⋮
S1	S100

Max no. of tuples in Sup $\bowtie$ catalog 100

with no. of Tuples in Sup $\bowtie$ catalog = (bcoz Sid is foreign key of Supplier Sid so, Sid refer minimum 100 tuples is necessary. Suppliers Tab

[04]

$$\pi_{sname}\left(\pi_{sid}\left(\pi_{pid}\left(\sigma_{col=Red}(Parts)\right) \bowtie_{sid=cid} \pi(Catalog)\right) \bowtie_{sid=sname} \pi(Supplies)\right)$$

* Efficiency point of view all 4 Query are same. But in complexity point of view Q3 & Q4 are more complex.

Check for Commutative

1) π (Projection) = ✗

$$\pi_{A2}(\pi_{A1A2}(R)) \neq \pi_{A1A2}(\pi_{A2}(R))$$

2) σ (Selection) = ✓

$$\sigma_{C2}(\sigma_{C1}(R)) = \sigma_{C1}(\sigma_{C2}(R)) = \sigma_{C1 \wedge C2}(R) = (\sigma_{C1}(R)) \wedge (\sigma_{C2}(R))$$

3) Cross Product = ✓

$$R1 \times R2 = R2 \times R1$$

Que) R(ABCDEGH)

F = { AB → C , AC → B, AD → E , B → D, BC → A, E → G }

D = { ABC ; ACDE, ADG }

ABC	ACDG	ADG

$A^+ = A$
$B^+ = BD$
$C^+ = C$
$AB^+ = ABC$
$BC^+ = ABCDEG$
$AC^+ = ACBDEG$

$A^+ = A$
$C^+ = C$
$D^+ = D$
$AC^+ = ABCDEG$
$AD^+ = ADEG$
$AG^+ = AG$
$CD^+ = CD$
$CG^+ = CG$
$DG^+ = DG$
$ACD^+ = ACDBEG$
$ACG^+ = ACBDEG$
$CDG^+ = ACDBEG$

$A^+ = A$
$D^+ = D$
$G^+ = G$

$R_1(AB)$	$R_2(BC)$	$R_3(CD)$	
$A \to B$	$B \to C$	$C \to D$	$A \to B$
$B \to A$			$B \to C$
	$C \to B$	$D \to C$	$C \to D$
			$D \to A$ ($D \to C$, $C \to B$, $B \to A$)
$A^+ = ABCD$	$B^+ = ABCD$	$C^+ = ABCD$	
$B^+ = ABCD$	$C^+ = CDAB$	$D^+ = ABCD$	

* All the original fun dependency are presed in it so it is Dependency preserving decomposition

Que) Check whether decomposition Preserving : or Not :-

$R(A\,B\,C\,D\,E\,F)$

$F = \{AB \to C, C \to D, D \to E, E \to F\}$

$D = \{AB, CDE, EF\}$

R1(AB)	R2(CDE)	R3(EF)
	$C \to D, C \to E$	$E \to F$
—	$C \to DE$	$F \to E$
	$D \to E, CD \to E$	
	$EC \to D$	

Not preserving dependency

$A^+ = A$

$B^+ = B$

$C^+ = CDEF$

$D^+ = DEF$

$E^+ = EF$

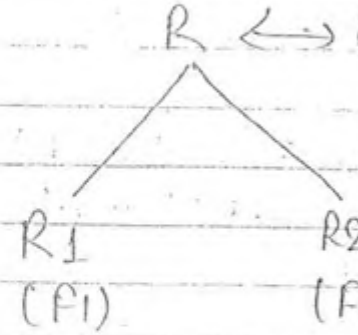
$CD^+ = CDEF$

$DE^+ = DEF$

$EC^+ = ECDF$

$E^+ = FE$

$F^+ = FE$



Which of them is Always false

a) $F1 \cup F2 \supseteq F$
b) $F1 \cup F2 = F$ (Dependency preserving)
c) $F1 \cup F2 \subset F$ (Not preserving dependency)
d) All the above

* If $R1 \bowtie R2 \supseteq R$ &
$F1 \cup F2 \subseteq F$ then lossless and preserving dependency

If $R1 \bowtie R2 \subseteq R$ &
$F1 \cup F2 \subseteq F$ then lossy and not preserving dependency.

Que) Check whether Dependency are Preserving or not
$R = (ABCD)$
$F = \{A \to B, B \to C, C \to D, D \to A\}$

$D2 \{AB, BC, CD\}$

2) Dependency Preserving Decomposition :-

Let R be the Relational schema with FD set F decomposed into R1, R2, ——, RN with FD set F1, F2, ——, FN respectively

In general $F_1 \cup F_2 \cup \text{——} \cup F_N \subseteq F$

if $F_1 \cup F_2 \cup \text{——} \cup F_N \equiv F$

Dependency Preserving

if $F_1 \cup F_2 \cup \text{——} \cup F_N \subset F$

Not preserving dependenc

Que) R(ABCD)

{A → B, B → C, C → D}

D = {AB, BCD}

Sol^{n} :- Identify FD's of Subrelations :-

R_1(AB)	R2(BCD)
[illegible]	B → CD
[illegible]	C → D
A → B	BC → D
	BD → C

Apply membership test

A → B

B → C

C → D

$A^+ = ABCD$

$B^+ = BCD$

$B^+ = BCD$

$C^+ = CD$

$D^+ = D$

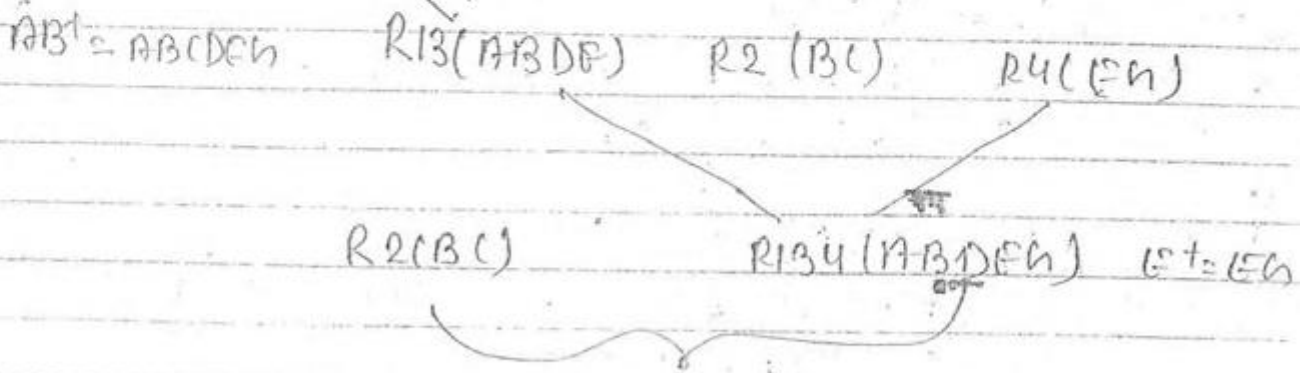
$BC^+ = BCD$

$CD^+ = CD$

$BD^+ = BCD$

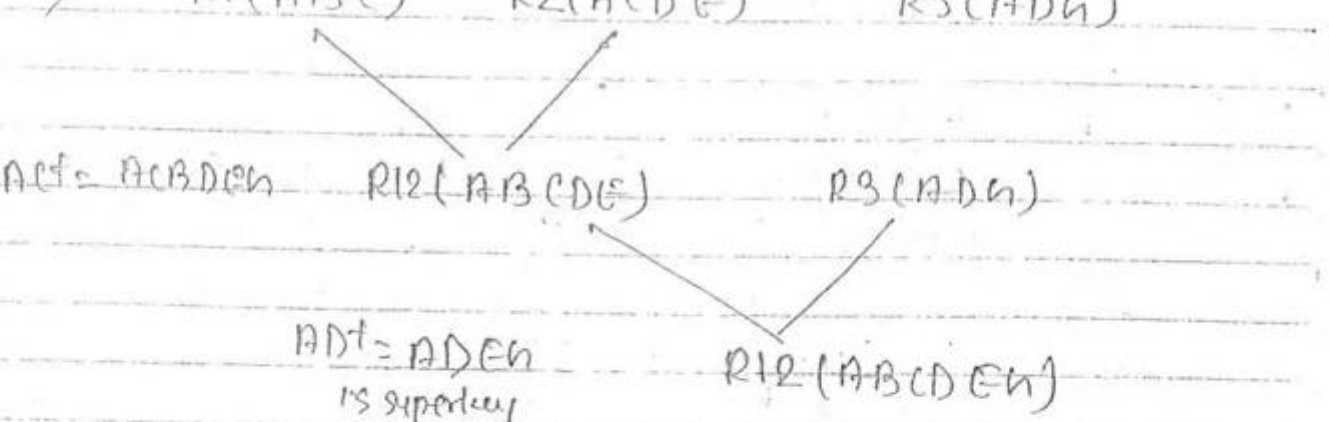
$F_1 \cup F_2$ Covers F Then preserving dependency

a) R1(AB) R2(BC) R3(ABDE) R4(EH)



Not possible to join into single relation. So, it is lossy.

b) R1(ABC) R2(ACDE) R3(AD H)



So it is lossless because it is decomposed into single relation.

Decomposition is lossless only if N Relations Become Single Relation.

Ques) R(A B C D E F)

$F = \{A \to B, C \to DE, AC \to F\}$

$D = \{BE, ACDEF\}$

2) R(A B C D E)

$F = \{A \to BC, CD \to E, B \to D, E \to A\}$

$D = \{ABC, ACDE\}$

3) R(A B C D)

$F = \{B \to C, D \to A\}$

$D = \{BC, AD\}$

4) R(A B C D E G)

$F = \{AB \to C, AC \to B, AD \to E, B \to D, BC \to A, E \to G\}$

a) $\{AB, BC, ABDE, EG\}$

b) $\{ABC, ACDE, ADG\}$

Soln 1) Lossy $E^+ = \{E\}$ not superkey so lossy

2) $AC^+ = \{ABCDE\}$ superkey lossless

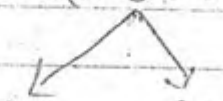
3) $F \cap D = \phi$, so lossy

4)

* Algorithm for checking decomposition is lossless or los

⇒ Let R be the Relational Schema with FD set F decomposed into $R_1, R_2, --, R_n$.
Given Decomposition is lossless only if

[1] $R_1 \cup R_2 \cup ---- \cup R_N = R$
(because of decomposition R should not lose any attribute)
i.e. if R(ABCDE)



AB BCD one attribute is lost after decomposition.

and [2] a) Let R_i & R_j are two relations can be merge into single Relation $R_{ij} = R_i \cup R_j$ only if following conditions are true

I. $R_i \cap R_j \neq \phi$
i.e. R(ABCDE)

R_1(AB) R_2(CDE)
i.e. $R_1 \cap R_2 = \phi$ not possible.

and II. $R_i \cap R_j \longrightarrow R_i$ ($R_i \cap R_j$ is superkey of R_i)
Or
$R_i \cap R_j \longrightarrow R_j$ ($R_i \cap R_j$ is superkey of R_j).

b) Repeat @ until N Relation become single relation.

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

$R_1 \bowtie R_2 = R$

[Lossless Join]

* If common attribute b/w R_1 & R_2 ($R_1 \cap R_2$) not super key of either R_1 or R_2 Then decomposition is lossy.

* If common attribute b/w R_1 & R_2 ($R_1 \cap R_2$) is super key of either R_1 or R_2 or both Then decomposition is loss-less.

Que) $R(ABCD)$ $F = \{A \to B, B \to C, C \to D\}$
decomposed into $R1(ABC)$ $R2(CD)$

Sol^n- C is the common attribute check whether C is super key or not

$C^+ = CD$ (i.e. C is super key of Relation)

$\Rightarrow$ decomposition is loss-less decomposition.

* If $R_1(AB)$ $R_2(CD)$

$R_1 \bowtie R_2 = \pi(R_1 \times R_2) = R_1 \times R_2$

If there is no common attribute. Natural join is equal to cross product as shown above.

Date

Que 1- If $R(ABCD)$ decomposed into

$R_1(ABC)$ $R_2(CD)$

given decomposition is lossy.

$S = R_1 \bowtie R_2$

which one is true

a) $S \subseteq R$ b) $S \supset R$ c) $S \cup R = R$ d) $S \bowtie R = S$

Que) Decomposed into $R_1(AB)$ $R_2(AC)$

R_1

A	B
1	1
2	1
3	2

R_2

A	C
1	2
2	1
3	1

$\pi\left(\sigma_{R_1.A = R_2.A}(R_1 \times R_2)\right) = R_1 \bowtie R_2$

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

Arity!- Means addition of attribute of Relations

Cardinality!- Means multiplication of rows of relation

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

$R_1 \bowtie R_2 =$

$$\pi_{ABC}\left(\sigma_{R_1.B = R_2.B}(R_1 \times R_2)\right)$$

$\Rightarrow$

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

Extra Tuple after decomposition

$r_1 \bowtie r_2 \supset s$

(lossy Join)

* If $R_1(ABC)$ $R_2(BCDE)$

then $R_1 \bowtie R_2 = \pi_{ABCDE}\left(\sigma_{R_1.B = R_2.B \wedge R_1.C = R_2.C}(R_1 \times R_2)\right)$

Example:-

Sid	Sname	Cname
S1	A	DB
S1	A	DM
S2	B	DM
S2	B	BM
S3	A	DB

(Sid Cname) :- Primary key

Sid → Sname (Partial Dependency) } Not in 2NF

Sid Cname → Sname

Decompose into 2NF:-

R1

Sid	Cname
S1	DB
S1	DM
S2	DM
S2	BM
S3	DB

Candidate key : (Sid Cname)

R2

Sid	Sname
S1	A
S2	B
S3	A

Sid → Sname

Above Decomposition is

LossLessJoin
+
Dependency Preserving
+
2NF (No partial Dependency)

Example:-

R(ABCDEF)

{AB → C, C → D, B → EF}

Check the above relation is in 2NF or not.

Soln:- {AB → C, C → D, B → EF}

(Partial Dependency)

Candidate Key :- AB

Convert it to 2NF:-

Put the attribute in one relation which is suffering from partial dependency.

i.e.

R_2 [B E F]

B → EF

B! Candidate key

Put remaining attribute in other relation

R1 [A C D]

Put a common attribute in both the relation so that it is contain all the FD of original relation.

R_1 [A B C D]

AB → C

C → D

AB! Candidate key

R_2 [B E F]

B → EF

B! Candidate key

⇒ Lossless + Dependency Preserving + NO PD (2NF)

Example-

R(A B C D E F)

{AB → C, C → D, B → E, A → F}
(B → E : PD, A → F : PD)

AB : Candidate key

If we put all PD in one relation :

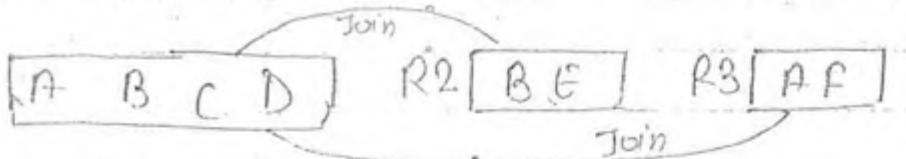
A	B	E	F

B → E, A → F (Again PD)

(AB) : Candidate key

So put all PD in one relation

Dependency preserving + dossless Join + NO PD (2NF)

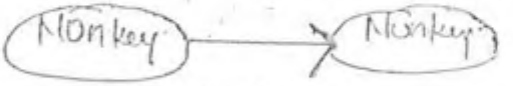


AB → C
C → D

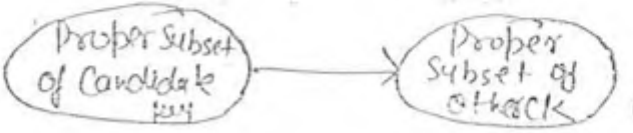
B → E

A → F

* ⇒ 2NF Not Free from Redundency bcoz of

1) 

not violate the condition of 2NF

2) 

not violate the condition of 2NF

* 2NF has less Redundancy than 1NF.

Third Normal Form:-

Let R be the relational Schema with $X \to Y$ any Non-trivial FD over R is in 3NF only if:—

If internally satisfy that the relation is in 2NF. There is no need to check for 2NF if we apply these condition for checking that the relation is in 3NF or not.

a) X is Super key
(OR)
b) Y is prime attribute.

: Non Trivial FD

(Superkey) (OR) (Prime attribute)

⇒ 3NF def is also eliminating partial dependency
OR
we can say that it satisfy the 2NF.

⇒ 3NF eliminates Nonkey → Nonkey dependencies.

3NF:- Relation R is in 3NF only if

1) R should be in 2NF (no partial dependency)

and 2) Nonkey should not transitively determin by candidate key or Super key.
(No transitive dependency)

Failed (i.e. by 3NF definition)

⇒ Nonkey Z Transitive determined by Super X.

(ij 3NF definition satisfied)

here Z is directly determined by Super-key

Check it is 3NF or not.

Que) R{A B C D E}

{AB → CD, D → A, C → E}

Soln:- First check R is in 2NF or not

{AB → CD, D → A, C → E} } No partial dependency
Candidate key {AB, BD} } So, it is in 2NF.

Check for 3NF

{AB → CD, D → A, C → E} } So not is 3NF because E is transitively determined by superkey i.e. AB

Decompose into 3NF:-

R_1 [A B C D] R_2 [C E]

C → E

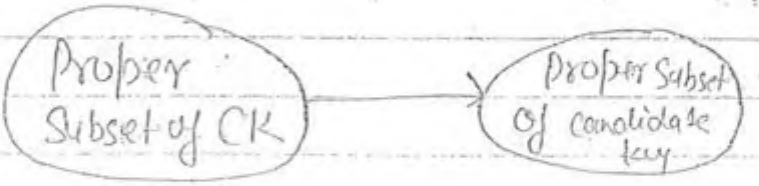
{C}: Candidate key

Yet redundancy AB → CD
exit D → A

Candidate key:- {AB, BD}

Now there is no violation of 3NF.
So, it is lossless join
Dependency preserving
of
3NF

* 3NF is also not free from Redundency becoz



is allowed which is the cause of redundency

* but Redundency is less than 2NF.

BCNF (Boyce Codd Normal Form) :-

Let R be the Relational Schema and $X \rightarrow Y$ any non-trivial functional dependency over R is in BCNF only if "X" is Super key.

$\Rightarrow$ BCNF free from Redundency that are forms because of Functional Dependency.

* It is more restricted then 3NF. Because there is only one condition to satisfy that the relation is in BCNF.

Example:- $R(A\ B\ C\ D)$

$\{AB \rightarrow CD,\ D \rightarrow A\}$

Sol^{n} 1- $\{AB \to CD, D \to A\}$

$\{AB, BD\}$	✓	✓	(2NF)
Candidate key	✓	✓	(3NF)
	✓	✗	(BCNF)

the Relation R is not in BCNF.

Decompose into BCNF :-

BCNF Decomposition
+
Lossless Join Decomposition
+
Not dependency preserving

R_1 [B C D]
$BD \to C$
$\{BD\}$: candidate key

R_2 [A D]
$D \to A$
$\{D\}$: candidate key

* BCNF some time violates the dependency preservin
* Because of BCNF decomposition some time the original relation R candidate key may elust. That results failure of the dependen preservation.

* If there is any dependency exist like

Proper Subset of Candidate key → Proper Subset of any other candidate key

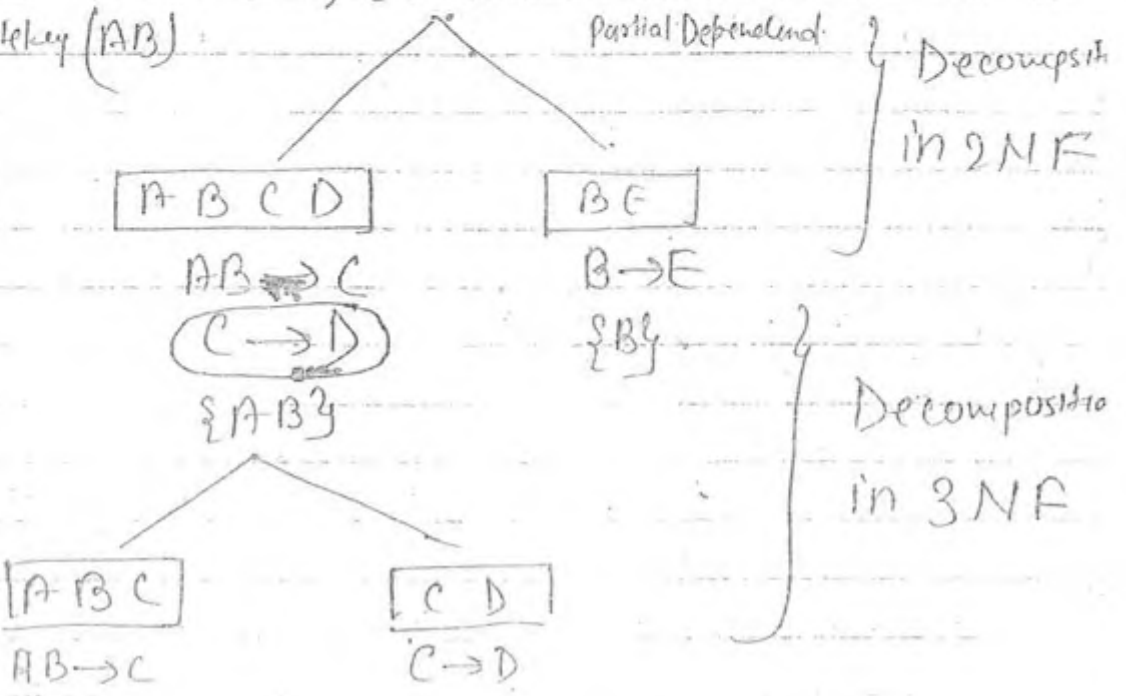
then there must be dependency preservatio is violates.

Example:- R(ABCDE)

R(AB→C, C→D, B→E)

Candidate key (AB)

Partial Dependend



we not lost candidate key of Relation R during 2NF decomposition & 3NF decomposition in the above example. So, there is dependency is preserving.

* If BCNF decomposition of R is also equa to 3NF decomposition of R Then Dependen preservation possible.

Example:- R(ABCDE)

R(AB→C, C→D, B→E)

R1 ABC AB→C

R2 CD C→D

R3 BE B→E

not lose any dependency of relation R.

⇒ If Relation R is in 3NF but not in BCNF Then decomposition of BCNF always failed Dependency preservation.

*

DB Design Goal:-

	1 NF	2NF	3NF	BCNF
1) 0% Redundancy	High	< 1NF	< 2NF	0% Redundancy bcoz of Single valued function dependencies (Redundancy possible becoz of MVD)
2) Lossless Join decomposition	Always	Always	Always	Always
3) Dependency preserving decomposition	Always	Always	Always	Some Relation may not possible to decompose into BCNF by preserving dependency

* 3NF is more accurate then BCNF.
Even BCNF is more restricted then 3NF but 3NF is more applicable then BCNF.

Few Results:-

⇒ If Relation R consist only Simple Candidate key (means no compound candidate key).
The R always in 2NF but may or may not 3NF/BCNF.

⇒ If Relation R consist only prime attribute (NO Non-prime attribute). Then R surely in 3NF but may or may not BCNF.

⇒ If every attribute of R is Super key. Then R surely in BCNF.

⇒ Relation R with NO-Non Trivial functional Depec ncy. Then 'R is in BCNF.

beoz X → Y non trivial i's in BCNF only if X is super
i.e. If non trivial ⟶ BCNF
if not non trivial ⟶ BCNF

→		
T	T	T
F	T	T

⇒ Binary Relation

Relation with Two attributes always in BCNF

R(AB)

		Candidate key
a)	$A \to B$	A
b)	$B \to A$	B
c)	$A \to B, B \to A$	A, B
d)	No Non Trivial FD.	AB

Que) Assume no multivalued attribute

R(A B C D E F)

F = {A → BCDEF ; BC → ADEF , B → E, D → E}

Soln:-

F = {A → BCDEF , BC → ADEF , B → E , D → E}

(✓) (✓) (✗) (✓)

Candidate key - {A, BC} (Not in 2NF)

ABC D E

A → BC D E

BC → A D E

Not in 3NF (D → E)

{A, BC} candidate key

ABC D

A → BC D

BC → A D

{A, BC} ! Candidate key

D E

D → E

{D} Candidate key

B F

B → F

{B} ! Candidate key

B F

B → F

{B} ! Candidate key

2NF + Not in 3NF

3NF + BCNF

B	C	F		A	D	E
				A → DE		

(Ques) R(A B C D E F G H I J)

{AB → C, A → DE, B → F, F → GH, D → IJ}

Soln- {AB → C, A → DE, B → F, F → GH, D → IJ}

Candidate Key {AB}

(AB)+ = ABCDEFGHIJ

2NF Decomposition

A	B	C

AB → C

B	F	G	H

B → F

F → GH

A	D	E	I	J

A → DE

D → IJ

Candidate key {AB}

{B}

{A}

3NF Decomposition

A	B	C

AB → C

B	F

B → F

F	G	H

F → GH

A	D	E

A → DE

D	I	J

D → IJ

It is also in BCNF.

Que) R{A B C D E F}

{A → BCDEF, BC → ADEF, B → F, D → F}
PD

R1: {A B C D E} — A → BCDE, BC → ADE

R2: {B F} — B → F

R3: {D F} — D → F

Que) R(A B C D E F G H)

{AB → C, AC → B, AD → E, B → D, BC → A, E → G}

Soln:-

{AB → C, AC → B, AD → E, B → D, BC → A, E → G}
PD

Candidate key (ABFH, BCFH, ACFH)

2NF Decomposition

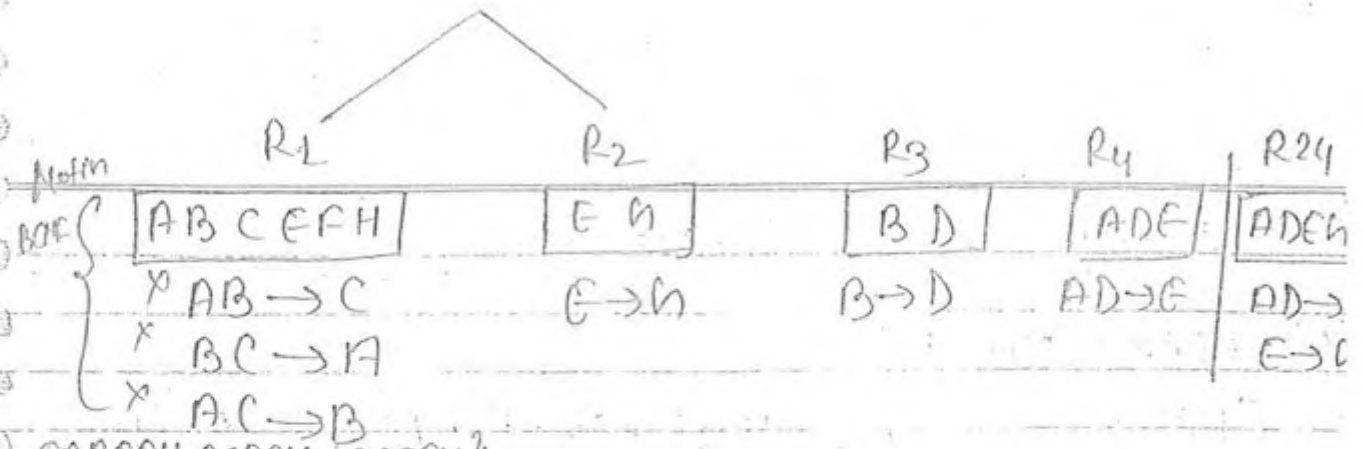
R1 {A B C E F G H} — AB → C, AC → B, BC → A, PD (E → G)

R2 {B D} — B → D

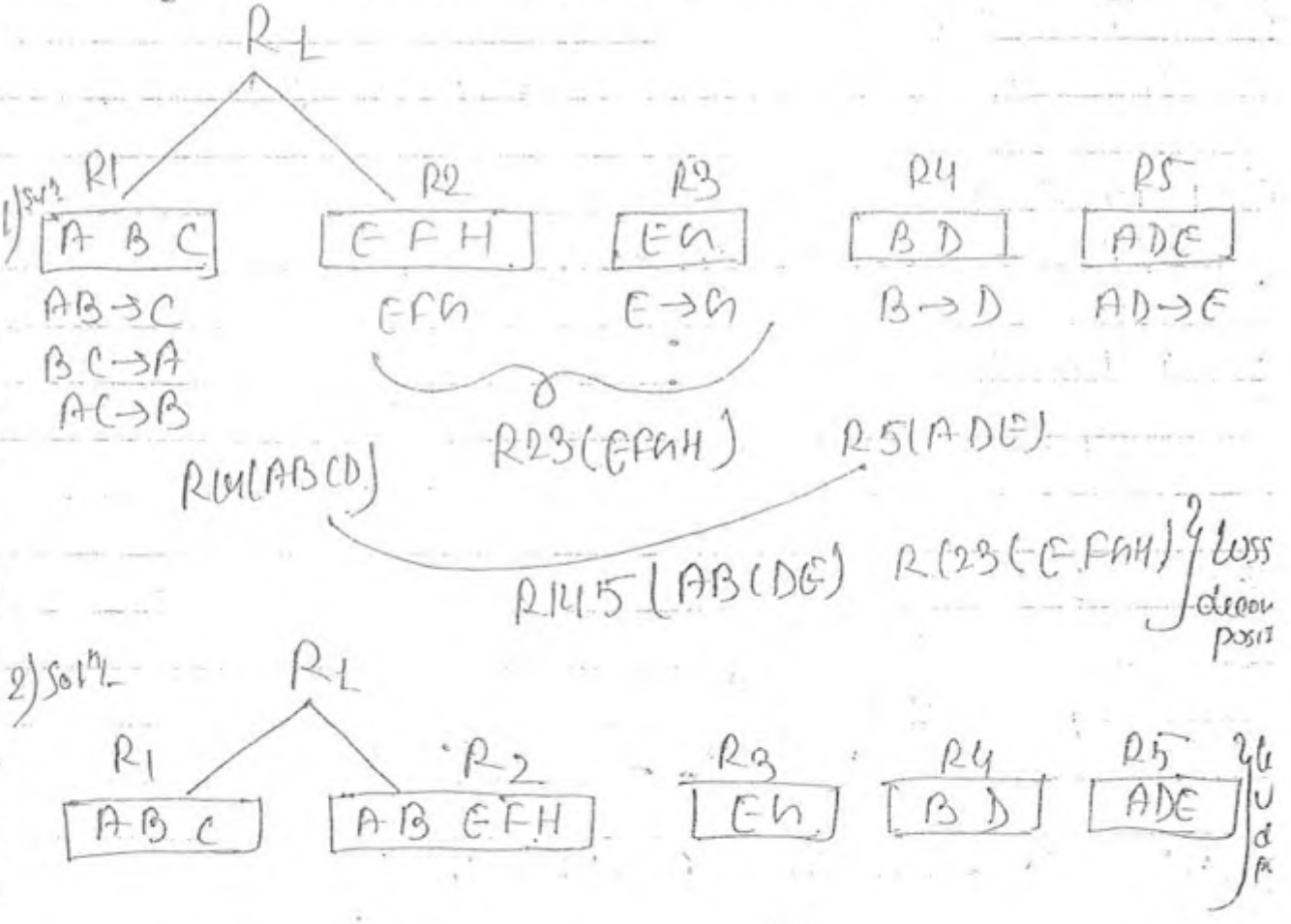
R3 {A D E} — AD → E

Candidate key: {ABFH, BCFH, ACFH} {B} {AD} ...

R1: Again it is not in 2NF



* There is only 3 table needed to be in 2NF but 4 table needed to be in 3NF because if we merge E→H then it violates the condition of 3NF. So, minimum 4 table needed for 3NF



22/08/2014

Que) Identify highest Normal form

[1] R(ABCDEFGH)

{ABC→DE, E→FG, H→G, G→H, ABCD→EF}

[2] R(ABCDE)

{AB→C, C→D, D→E, E→A, D→B}

[3] R(ABCDEFGH)

{AB→CD, D→E, E→F, F→H, C→EF, H→A, G→B, A→B}

[4] R(ABCDEF)

{AB→C, C→D, D→E, E→F, F→A}

[5] R(AB CD E)

{AB→C, C→DE, E→F, F→A}

[1] Soln: Candidate key [AB C]

2NF ✓

3NF ✗

BCNF ✗

[2] Soln: Candidate key [AB, EB, C, D]

{AB→C, C→D, D→E, E→A, D→B}

Right side of FD is all Prime attribute. So, it is always in 3NF.

3NF: ✓

BCNF: ✗

(3) Soln:- Candidate key:- {AH, HA, FA, EA, DA, CA}.

{AB→C, D→E, E→F, F→H, C→EF, H→A, A→B, A→B}
PD

Not in 2NF

So, it is in 1NF

(4) Soln:- Candidate key:- {AB, FB, EB, DB, CB}

Not in BCNF. But it is in 3NF. Because right side of FD {AB→C, C→D, D→E, E→F, F→A} side all Prime attribute. So, it is in 3NF

(5) Soln:- Candidate key:- {AB, FB, EB, CB}

{AB→C, C→DE, E→F, F→A}
PD

not in 2NF

So, it is in 1NF.

Minimal Cover or Canonical Cover:-

It is used to eliminate redundant functional dependency.

Example:

$F = \{A \to B, B \to C, A \to C, AB \to C\}$

$A \to C$ is eliminated

$A \to B, B \to C \Rightarrow A \to C$

$\therefore AB \to C$ is eliminated because

Argumentation Property: If $B \to C$

$AB \to AC$

$\therefore AB \to A$

$AB \to C$

then $G = \{A \to B, B \to C\}$

$\Rightarrow (F \equiv G)$

$\therefore$ it remove those FD that are obtained from other FD within the set. i.e. remove the redundant functional dependency.

Extraneous Attribute:-

(a) $\{AB \to C, A \to C\} \equiv \{A \to C\}$

B is extraneous Attribute:-

Ex:- $F = \{AB \to C, A \to C\}$

$G = \{A \to C\}$

F cover G:- True

G cover F:- $G = \{A \to C\} \models AB \to C$ (By applying union property) True

$\models A \to C$ True

$\Rightarrow A \to C$ determine all the FD of Fs

(b) $\{AB \to C, A \to B\} \equiv \{A \to C, A \to B\}$

Proof:- First check A is extraneous

$F = \{\overset{X}{A}B \to C, A \to B\}$

$G = \{B \to C, A \to B\}$

F cover G:- $F = \{AB \to C, A \to B\} \models B \to C$ ✗, $A \to B$ ✓

G cover F:- $G = \{B \to C, A \to B\} \models AB \to C$ ✓, $A \to B$ ✓

$F \neq G$

Take B is extraneous attribute

$F = \{AB \to C, A \to B\} \models A \to C$ ✓

$\models A \to B$ ✓

$h = \{A \to C, A \to B\} \models AB \to C$ ✓

$\models A \to B$ ✓

F covers h ✓

h covers F ✓

$(F \equiv h)$

c) $\{A \to BC, B \to C\} \equiv \{A \to B, B \to C\}$

d) $\{AB \to CD, BC \to D\} = \{AB \to C, BC \to D\}$

Proof:-

$\{AB \to C, BC \to D\} \models AB \to D$

Combination of transitivity & argumentation

if $AB \to C$

$AB \to BC$ if $BC \to D$

then $AB \to D$

* "c" & "d" are used to remove extraneous attribute of right side, "a" & "b" are used remove extraneous attribute of left side.

(Que) Minimal cover of

$F = \{A \to BC, B \to C, C \to B\}$

Soln - $F = \{A \to BC, B \to C, C \to B\}$ $f = \{A \to BC, B \to C, C \to$

minimal cover $F_1 = \{A \to B, B \to C, C \to B\}$ minimal cover $F_2 = \{A \to C, B \to C, C \to$

$F_1 = F_2 = F$

⟹ Minimal cover is not unique. But all minimal covers logically Equivalent.

(Que) $F = \{AB \to D, E \to D, A \to B, AC \to D\}$

Soln - $\{AC \to D, E \to D, A \to B, AC \to D\}$

$\{AC \to E, E \to D, A \to B, AC \to D\}$

$\{AC \to DE, E \to D, A \to B\}$

$\{AC \to E, E \to D, A \to B\}$

Que) Identify minimal cover:-

$\{A \to BC, B \to C, A \to B, AB \to C\}$

$\{A \to B, B \to C, A \to B, AB \to C\}$

$\{A \to B, B \to C, AB \to C\}$

$\{A \to B, B \to C, A \to C\}$ → $\{A \to B, B \to C\}$

$\{A \to BC, B \to C\}$

$\{A \to B, B \to C\}$

Que) $\{A \to BC, B \to AC, C \to AB\}$

$\{A \to BC, B \to A, B \to C, C \to AB\}$

$\{A \to B, B \to C, B \to A, C \to AB\}$

$\{C \to A, B \to C, B \to A, A \to B\}$

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

$\{C \to A, B \to C, A \to B\}$

$\{A \to BC, B \to AC, C \to AB\}$

$\{A \to B, A \to C, B \to AC, C \to AB\}$

$\{A \to B, A \to C, B \to A, C \to AB\}$ ($A \to B, A \to C, B \to AC, C \to$; $A \to B, A \to C, B \to A, C \to$; $A \to BC, B \to A, C \to$)

$\{A \to B, A \to C, B \to A, C \to A\}$

$\{A \to BC, B \to A, C \to A\}$

Q) Identify minimal cover:-

$\{A \to BC, CD \to E, E \to C, D \to AEH, ABH \to BD, DH \to BC\}$

Sol^n: $\{A \to BC, D \to E, E \to C, D \to AEH, AH \to BD, DH \to BC\}$

$\{A \to BC, E \to C, D \to AEH, AH \to BD, DH \to BC\}$

$\{A \to BC, E \to C, D \to AEH, AH \to D, D \to BC\}$

$\{A \to BC, D \to AEH, E \to C, AH \to D\}$

$\{A \to BC, D \to AEH, E \to C, AH \to D\}$

To check minimal cover is correct or not:-

⇒ [1] Minimal cover should not contain any extraneous attribute.

[2] Minimal cover logically equivalent to original cover. i.e. we obtained all the FD from minimal cover FD, of the original Relation's FD.

Multivalued Dependency:- (MVD)

Representation of Functional Dependency:-

Single Valued FD ⇒ FD, →

Multivalued FD ⇒ MVD →→

Occurance of MVD:-

⇒ If two or more independent relations kept in single relation then MVD possible.

⇒ **Multivalued attribute:-**

More than one value for each record of the attribute.

⇒ If relation contain Multivalued attribute. Then Relation not in 1NF.

* Candidate key can't determine multivalued attribute.
Candidate key only determine single valued attribute

⇒ Example:- R(A B C) C: Multivalued attribute
A: is primary key

R Not in 1NF
Convert into 1NF: R(A C, B)
primary key {AC}

⇒ R(A B C D) C, D: Multivalued attribute
A: is primary key

R Not in 1NF
Convert into 1NF: R(A C D, B)
Primary key {ACD}

⇒ R(A B C D E F)
{A → B, B → C, C → D, D → A}
and E F are Multivalued attribute
Then Candkey :- A, B, C, D
(Not in 1NF)

If we convert above multivalued attribute relation in single valued relation then their is no any changes into FD, Only candidate key changes:-

R(A B C D E F) // No multivalued attributes.
{A → B, B → C, C → D, D → A}
Cand key :- AEF, BEF, CEF, DEF
(1NF relation)

relation does not contain any multivalued attri

Que: If R(A B C D E F)

and {A → B, B → C, C → D, D → A}

& Candidate key :- A, B, C, D

then it is in which normal form:-

Soln:- In the above relation candidate key does not determine all the attribute of relation then in relation multivalued attribute is their. So, it is not even in 1NF. Candidate key not determine E F this implies E & F are multivalued attribute.

⇒ if Relation consist two or more multivalued attribute while conversion of Relation into 1NF Then MVD is possible.

Example:- R{A, B, C, D}

A: Primary key C, D: Multivalued Attribute

R1NF(ACD, B)

ACD: Primary key

[ACD] May Suffers with MVD.

* One multivalued attribute never depends on other multi valued attribute-

Teacher	Course	Book
A	DB/C	Korth/ Navathe
B	DB	Korth

Multivalued attribute Relation

Teacher : Primary key

{Teacher, Course} } dependent relation
{Teacher, Book} }

{Course, Book} → Independent relation

(Single Valued Attribute Relation)

Teacher	Course	Book
A	DB	Korth
A	C	Navathe
B	DB	K

Dataloss occurred such that independent attribute (C, B) becomes depending each other which is incorrect.

To avoide above problem i.e. course & book depends we need to add more tuple!—

T	C	B
A	DB	K
A	DB	N
A	C	K
A	C	N
B	DB	K

BCNF

Ie. To Avoide data loss required to add more tuples)
⇒ There exist redundi data b/w (T, C) & (T, B)

⇒ Candidate Key: TCB

⇒ Highest Normal form of R is BCNF.

(BCNF Not free from redundancy occurred becoz of MVD.)

Example –

Teacher	Course	Book
A	CO/OS/DB/DS/CN	K/N/U

Total (15) tuples

[TC]: repeated 3 times

[TB]: repeated 5 times

Definition of MVD1–

Let R be the Relational schema X, Y are attributes of R and Z is R–(XUY).

$X \rightarrow\rightarrow Y$ exists in R only if following condition should satisfy b/w t_1, t_2, t_3, t_4 tuples!

1) $t_1.X = t_2.X = t_3.X = t_4.X$

and 2) $t_1.Y = t_2.Y$ && $t_3.Y = t_4.Y$

and 3) $t_1.Z = t_3.Z$ && $t_2.Z = t_4.Z$

* To show MVD, minimum 4 tuples are needed as per definition of MV

Representation of definition in table form:-

R	X	Y	Z
t_1	x_1	y_1	z_1
t_2	x_1	y_1	z_2
t_3	x_1	y_2	z_1
t_4	x_1	y_2	z_2

t	X	Y_1/Y_2	Z_1/Z_2

i.e.

$X \rightarrow\rightarrow Y$

After interchanging the tuple we obtain complementation rule

	X	Y	Z
t_1	x_1	y_1	z_1
t_2	x_1	y_1	z_2
t_3	x_1	y_2	z_1
t_4	x_1	y_2	z_2

$X \rightarrow\rightarrow Y$

X	Y	Z
x_1	y_1	z_1
x_1	y_2	z_1
x_1	y_1	z_2
x_1	y_2	z_2

$X \rightarrow\rightarrow Z$

① Complementation Rule:-

if $X \rightarrow\rightarrow Y$ Then $X \rightarrow\rightarrow R-(XUY)$

② Trivial MVD:- (Reflexive MVD)

if $X \supseteq Y$ (or) $XUR = R$ Then $(X \rightarrow\rightarrow Y)$ is trivial

Example:- $R(AB)$ { $A \rightarrow\rightarrow A$, $A \rightarrow\rightarrow B$ } Trivial

* To get redundancy due to single valued dependency we need at least two attribute then it is called non trivial. If there is no any dependency redundancy due to FD then FD is called trivial otherwise non trivial.

[3] Transitivity :-

If $X \to\to Y$ && $Y \to\to Z$ Then $X \to\to (Z - Y)$

ABC CDE DG

If there is no any common attribute

then $X \to\to Z$

[4] Argumentation :-

If $X \to\to Y$ and $W \supseteq Z$ then $XW \to\to YZ$.

Example :-

If $A \to B$

Then $ACD \to\to BC$

If $A \to B$

Then $ACD \to\to BCD$

* Every trivial multivalued dependency is also Reflexive multivalued dependency

[5] Replication :-

If $X \to Y$ Then $X \to\to Y$

* (Union / Splitting Not allowed)

Example:-

A	B
1	2
1	2
1	2
1	2
2	2

$(A \to B) = (A \to\to B)$

$(A \to\to B$ is trivial while χ is evil

Que) All possible MVD

A	B	C
1	2	3
1	2	4
1	3	3
1	3	4

if $A \to\to B$:
Due to complement
$A \to\to C$
* $AC \to\to B$
* $AB \to\to C$
* $AB \to\to BC$ * $AC \to\to BC$

if $C \to A$ then $C \to\to A$, $(B \to\to A$ (Due to argumentation
$CB \to\to AB)$

if $B \to A$ then $B \to\to A$, $BC \to\to A$ (Due to argumenta
$BC \to\to AC$

$BC \to A$ then $BC \to\to A$

4NF:-

Relational Schema R is in 4NF only if

(a) R should be in BCNF
(Non trivial FD $X \to Y$ with X: Superkey)

and (b) Every non-trivial MVD $X \to\to Y$ with X should be Super keys.

Note:-

Super keys or minimal super key or candidate keys identified by Functional Dependency only. Not from Multivalued Dependency.

Example:-

T	C	B
A	DB	R
A	DB	M
A	C	K
A	C	N
B	DB	K

Candidate key (T(B)

Highest NF: BCNF

$\{T \to\to C, T \to\to B\}$

(T: Superkey, T: Superkey)

$\Rightarrow$ It is in BCNF not in 4NF.

Decompose it to be in 4NF.

T	C
A	DB
A	C
B	DB

(TC) : Candidate key

T	B
A	K
A	N
B	K

(TB) : Candidate key

MU Non trivial MVD

4NF

$T \twoheadrightarrow C$

Trivial MVD

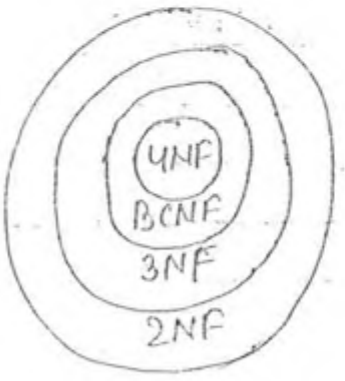
$T \twoheadrightarrow B$

Trivial MVD

Result:-

	2NF	3NF	BCNF	4NF
Redundancy!	X	X	X (Redundancy possible being of MVD)	✓
Lossless	✓	✓	✓	✓
Dependency Preservation	✓	✓	X	X (being it is also in BCNF)

This diagram show that 4NF is most restricted NF than other three NF.



Query languages:-

1) Relational Algebra.
2) SQL
3) Tuple Relational Calculus.

* Always Query execution done row by row & one row at a time.

Example

	Sid	Sname	Marks
X	S1	A	50
✓	S1	B	70
✓	S1	C	60
X	S1	C	40

Query on R.R.
Query is a question on R

Query:- Retrieve Student whose marks greater than 50

Relational Algebra	SQL	Tuple Relational Calculus
π, σ, $\cup$, $-$ etc	SELECT from WHERE Clause	(First Order logic) $\in$, $\exists$, $\forall$, $\wedge$, $\vee$, $\neg$, $\rightarrow$ etc

Relational Algebra:-

* Always Eliminates duplicate tuple from the resu

Ex :-

A	CS
~~A~~	~~CS~~
A	IT

It gives O/P only

A	CS
A	IT

Basic Operations:-

π : Projection

σ : Selection

$\times$: cross product

$\cup$: Union

$-$: Minus

ρ : Rename

Derived operations:-

$\cap$: Intersection

$A \cap B = A - (A - B)$

$\bowtie$: Join

$/$: Division

Set Operations:-

$\cap, \cup, -$: To apply these ~~relations~~ operator, Relations should be Union Compatable.

R & S said to be Union compatable :-

only if

a) domain of R & S should be same

* if no. of attribute in R_1 is 2 & no. of attribute in R_2 is 3 then $\cup, \cap, -$ is not applicable there.

Ex :-

Sid	Sname
S1	A

$\cup$ ✗

Sid	Sname	branch
S1	A	CS

and b) Range of attributes should be similar from L to R.

* not only the range of attribute of R1 & R2 are same but also the name of the attributes of R1 & R2 must be same for $\cup, \cap, -$.

Ex :-

Sid	Sname
	↓
	Varchar(2)

$\cup$ ✗

Sid	branch
	↓
	Varchar(2)

Sid	Sname
	↓
	Varchar(2)

$\cup$ ✓ $\cap$

Sid	Sname
	↓
	Varchar(2)

R & S cardinalities are m & n each.

R ∪ S : [Range of Tuples] { max(m,n) to m+n }

R ∩ S : " { ϕ to min(m,n) }

R − S : " { ϕ to m }

Cross Product :-

R & S Relations Cardinality m, n respectively and arity x, y respectively

then R × S { Cardinality : m * n
Arity : x + y

Example :-

R

A	B	C
1	2	3
2	2	1

S

B	C	D
2	3	3
2	4	1

R × S

A	B	C	B	C
1	2	3	2	3
1	2	3	2	4
2	2	1	2	3
2	2	1	2	4

* Cross product is used to compare any attribute of one relation with any other attribute of same relation or other relation.

⇒ Join :- Cross product followed by selection and then projection.

Cross

* Conditional Join :- ($\bowtie_C$)

* By default it profits all the columns of relation R & S.

$$R \bowtie S = \Pi \left(\sigma_{C_1 \leq C_2} (R \times S) \right)$$

Example $R \underset{R.A < S.B}{\bowtie} S = \underset{ABCBCD}{\Pi} \left(\sigma_{R.A < S.B} (R \times S) \right) \Rightarrow$

A	B	C	B	C	D
1	2	3	2	3	3
1	2	3	2	4	1

* Natural Join :- ($\bowtie$)

* It always profect distinct column

Example

$$R \bowtie S = \underset{ABCD}{\Pi} \left(\sigma_{R.B = S.B \wedge R.C = S.C} (R \times S) \right) \Rightarrow$$

A	B	C	1
1	2	3	3

* Natural join is same as condition join if no any common attribute b/w relations. Thus it profect all the relation.

* Left Outer Join :- ⟕ :-

R ⟕ S and Tuples from left Relation those are failed N. Join condition.)

R ⟕ S

A	B	C	D
1	2	3	3
2	2	Null	

* Right Outer Join :- ⟖

R ⟖ S and Tuples from right side Relation those are failed N. Join condition.)

R ⟖ S

A	B	C	D
1	2	3	3
Null	2	4	1

* Full Outer Join :- R ⟗ S

R ⟗ S = (R ⟕ S) ∪ (R ⟖ S)

A	B	C	D
1	2	3	3
2	2	1	Null
Null	2	4	1

* (some/any/atleast one) used as same in DBMS.
* (All/every) used as same in DBMS.

DIVISION OPERATOR:-

Division operator is used to retrieve tuples those are satisfied conditions for every tuple values of the other Relation S.

Enrolled

Sid	Cid	Fee
S1	C1	-
S1	C2	-
S1	C3	-
S2	C1	-
S2	C2	-
S3	C1	-

Course

Cid	Cname
C1	-
C2	-
C3	-

Query1- Sid's who are Enrolled some course.

Sid
S1
S2
S3

Π_{Sid} (Enrolled)

Query1- Sid's who are Enrolled all courses:

Π(Enrolled) / Π(Course)
Sid Cid Cid

O/P:- S1

* Notation of Division operator:-

P(A, B) / Q(B)

Retrieves 'A' values from P Relation for that there Exist (a, b) tuple specified every value of Q Rela

π(Enrolled) — sid

$\pi_{sid,cid}(Enrolled) \,/\, \pi_{cid}(course) = \pi_{sid}\left(\pi_{sid}(Enrolled) \times Course - Enrolled\right)$

sid	cid
S1	C1
S2	C2
S3	C3

(allstudents) (all courses)

sid	cid
~~S1~~	~~C1~~
~~S1~~	~~C2~~
~~S1~~	~~C3~~
~~S2~~	~~C1~~
~~S2~~	~~C2~~
S2	C3
~~S3~~	~~C1~~
S3	C2
S3	C3

(Universal) (Every Student Enrolled Every Course)

Sid
S1
S1
S1
S2
S2
S3

(Actual Enrolled (Student who are Enrolled some course)

$\pi_{sid,cid}(Enrolled) \,/\, \pi_{cid}(course)$ —

$\pi_{sid}(Enrolled) - \left[\pi_{sid}\left(\pi_{sid}(Enrolled) \times Course - Enrolled\right)\right]$

=

Sid	Cid
S2	C3
S3	C2
S3	C3

(Compliment Data) (Students who Enrolled all courses)

= S2, S3 (disqualified) (At least one course)

23/08/11

* Suppliess (Sid , Sname , rating)

Foreign key

Parts (Pid , Pname , Colos)

PK

Catalog (Sid Pid , Cost)

Query1 - Retrieve Sid's of Supplies's whose rating greater than 10.

$$\Pi_{Sid}\left(\sigma_{rating>10}(Suppliess)\right)$$

Query1- Sid's of suppliess whose rating greates than 10 or who supplied some part.

Supplies

Sid	rating
S1	5
S2	15
S3	7

Catalog

Sid	Pid	—
S1	P1	—
S2	P1	—
S1	P2	—

$$\pi_{sid}\left(\sigma_{rating>10}(suppliers)\right) \cup \pi_{sid}(catalog)$$

$$\{S1,S2\} = \{S2\} \cup \{S1,S2\}$$

Query:- Retrieve Name's of Suppliers whose rating > 10 (and) who are also supplied some part.

Supplies

sid	name	rating
S1	A	5
S2	A	15
S3	B	7

Catalog

Sid	Pid	-
S1	P1	-
S2	P1	-
S1	P2	-

By join we eliminate S3 because it not supply any part because S3 is not in Catalog Sid but it is in supplies Sid

[Q1] $$\pi_{sname}\left(\sigma_{S.sid=C.sid \wedge rating>10}(Supplier \times Catalog)\right)$$

Or

By using natural join

[Q2] $$\pi_{sname}\left(\sigma_{rating>10}(Supplies \bowtie Catalog)\right)$$

Or

By apply conditional join

[Q3] $$\pi_{sname}\left(Supplies \underset{\substack{S.sid=C.sid \\ rating>10}}{\bowtie} Catalog\right)$$

[04] By apply conditional join:-

$$\pi_{sname}\left(\sigma_{rating>10}\left(Supplies \bowtie_{S.sid=Cid} Catalog\right)\right)$$

[01], [02], [03], [04] are different way of writing the same query.

100 tuple 50 tuple

⇒ Cross Product result: 5000

$$\pi_{sname}\left(\sigma_{rating>10}\left(Supplies \bowtie Catalog\right)\right)$$

Out of 100 select tuple

⇒ Cross product result 500

$$\left(\sigma_{rating>10}(Supplies\ S) \bowtie Catalog\right)$$

50 tuple

Out of 100 select only those supplier whose rating is > 10 i.e. only 10 tuple.

Query:- Names of suppliers whose rating > 10 and who supplied part for more than 100 cost.

$$\pi_{sname}\left(\sigma_{rating>10 \wedge cost>100}(Suppliers \bowtie Catalog)\right)$$

$$\Big\Vert$$

$$\pi_{sname}\left(\sigma_{rating>10}(Suppliers) \bowtie \sigma_{cost>100}(Catalog)\right)$$

because rating attribute is only belong to the supplier table & cost attribute condition is only belongs to the catalog table. So first we select only those tuple who satisfy the condition then apply natural join on it & then project it. It reduces the cost of the join because join is depends for the cost of the relation. So, it decrease the cost of the whole relation.

* Let A ∈ Relation R only.

$$\sigma_{A=9}(R \bowtie S) = \sigma_{A=9}(R) \bowtie S$$

Query 1- Retreive Names of the Suppliers who supply some red part.

~~$\pi_{sname}(\sigma_{col=Red}(Supplies \bowtie Catalog \bowtie Parts))$~~

[Q1] $\pi_{sname}\left(\sigma_{col=Red}(Supplies \bowtie Catalog \bowtie Parts)\right)$

|||

[Q2] $\pi_{sname}\left(\sigma_{S.sid=C.sid \wedge C.pid=P.pid \wedge color=RED}(Supplies \times Catalog \times Parts)\right)$

|||

[Q3] $\pi_{name}\left(\pi_{sid}\left(\sigma_{color=RED}(Parts) \bowtie Catalog\right) \bowtie Suppliers\right)$

|||

if foreign key constraints is not there then

max no. of Tuples in sup ⋈ catalog = 100

min " " " " " " = 0

Que) R(ABC) S(BDE)

with R consist 1000 tuples S consists 2000 tuple and following dependencies are hold.

{B → D, D → E}

How many max. no. of tuples in R⋈S.

Soln:- Maximum no of tuples = 1000.

Rename Operator:-

Two syntex for Rename operator

1) Rename the Table $\rho(T, Catalog)$

2) Rename the Colume $\rho(Catalog)$

ρ_{SPC}

$\rho(Catalog)$

1→S, 2→P

T

Sid	Pid	C

Catalog

S	P	C

Catalog

S	P	Cost

$\rho_{SPC}\left(\rho(T, Catalog)\right)$

T

S	P	C

* Use of Rename Operator :-

σ (Catalog X Parts) $\equiv$ σ (ρ(C, catalog) X ρ(P, Parts))
Catalog.Pid = parts.Pid $\wedge$ col=RED C.pid = P.pid $\wedge$ col=RED

$\equiv$ σ (Catalog X ρ(parts))
Pid=P $\wedge$ col=RED P,n,c

Query :- Retrieve Sid of the Suppliers whose supplies some red part or some green part.

[Q1]

π_{sid} (Catalog $\bowtie$ $\sigma_{col=GREEN \vee col=RED}$ (Parts))

By using union operator :-

ρ (T_1, π_{sid} (Catalog $\bowtie$ $\sigma_{col=RED}$ (Parts)))

ρ (T_2, π_{sid} (Catalog $\bowtie$ $\sigma_{col=green}$ (Parts)))

$T_1 \cup T_2$

(Que) Retrieve "Sid" of the suppliers who supplied some red part and some green part.

$\Rightarrow$ [01] $\rho(T_1, \pi_{Sid}(Catalog \bowtie \sigma_{(parts)}))$

$\rho(T_2, \pi_{Sid}(Catalog \bowtie \sigma_{(parts)}))$

$T_1 \cap T_2 \Rightarrow (T_1 \bowtie T_2)$

$\not$ [02] $\pi_{Sid}(Catalog \bowtie (\sigma_{(parts)}))$

color = GREEN $\wedge$ color = RED

* [01] is true But [02] is not correct.

* Without using the Union operator:-

$\rho(P1, Parts)$

$\rho(P2, Parts)$

$\rho(C1, Catalog)$

$\rho(C2, Catalog)$

$\pi_{C1.Sid}\left(\sigma_{C1.Sid = C2.Sid}\left((C1 \bowtie \sigma_{(P1)}) \times (C2 \bowtie \sigma_{(P2)})\right)\right)$

col=RED, col=GREEN

By using natural join :-

$$\pi_{cname}\left(\pi_{sid}\left(\left(C1 \bowtie \sigma_{col=red}(P1)\right) \bowtie \pi_{sid}\left(C2 \bowtie \sigma_{col=green}(P2)\right)\right)\right)$$

Query :- Sid of the suppliers who supplied some red parts

$$\pi_{sid}\left(Catalog \bowtie \sigma_{color=Red}(parts)\right)$$

Sid of the suppliers who supplied every red parts.

(Sid who supplied past) / (All red parts)

$$\pi(Catalog_{sidpid}) \Big/ \pi_{pid}\left(\sigma_{col=RED}(parts)\right) =$$

$$\pi_{sid}(Catalog) - \pi_{sid}\left(\pi_{sid}(catalog) \times \pi_{pid}\left(\sigma_{col=RED}(parts) - \pi_{sidpid}(catalog)\right)\right)$$

Example- Catalog

Sid	Pid
S1	P1
S2	P2
S3	P2
S2	P1
S1	P3

Part :

Pid	Color
P1	R
P2	G
P3	R

Query :- Pid's of parts that are supplied by every supplier.

$$\pi_{pid,sid}(catalog) / \pi_{sid}(supplies) =$$

$$\pi_{pid}(catalog) - \pi_{pid}\left(\left(\pi_{pid}(catalog) \times \pi_{sid}(supplies)\right) - \pi_{pid,sid}(catalog)\right)$$

These are parts not supplied by every supplier

Query- Retrieve Sid of the Suppliers who supplied atleast two parts.

$$\rho(Q1, catalog)$$
$$\rho(Q2, catalog)$$

Sid	Pid	Cost
S1	P1	—
S2	P1	—
S1	P2	—

×

Sid	Pid	Cost
S1	P1	—
S2	P1	—
S1	P2	—

Sid	Pid	Cost	Sid	Pid
S1	P1	—	S1	P1
S1	P1	—	S2	P1
S1	P1	—	S1	P2
S2	P1	—	S1	P1
S2	P1	—	S2	P1
S2	P1	—	S1	P2
S1	P2	—	S1	P1
S2	P2	—	S2	P1
S2	P2	—	S1	P2

$$\left(\pi\left(\sigma(catalog) \times \rho(catalog)\right)\right)$$
$$Sid = Sid$$
$$Pid \neq Pid$$

$S(T1, Catalog)$ $S(T2, Catalog)$

$$\pi_{T1.Sid}\left(\sigma_{T1.Sid = T2.Sid \wedge T1.Pid \neq T2.Pid}(T1 \times T2)\right)$$

$$\Downarrow$$

$$\pi_{Sid}\left(Catalog \bowtie_{SPC} S(Catalog)\right)$$

$$Sid = S1 \wedge Pid \neq P$$

Query1. Retrieve Sid of the Suppliers who supplied at least Three parts.

$S(T1, Catalog)$ $S(T2, Catalog)$ $S(T3, Catalog)$

$$\pi_{T1.Sid}\left(\sigma_{T1.Sid = T2.Sid \wedge T2.Sid = T3.Sid \wedge T1.Pid \neq T2.Pid \wedge T2.Pid \neq T3.Pid \wedge T3.Pid \neq T1.Pid}(T1 \times T2 \times T3)\right)$$

Query1. Retrieve Sid's of suppliers who supplied exactly two parts.

| Sid's of Suppliers who supplied at least two parts | — | Sid's of Suppliers who supplied at least three parts |

Query 1:- Sid of the suppliers who supplied at most two parts.

π_{sid}(Supplies) − | Sid's of suppliers who supplied at least three parts |

if supplies table is not in DB then we take Catalog table in place of supplies because then their is no any table who gives 0 parts.

Query 1:- Retrieve Pairs of sid's (sid1, sid2) such the supplier with the first Sid charges more than supplier with the second sid for some parts

$\rho(T1, Catalog)$
$\rho(T2, Catalog)$

$$\pi_{T1.Sid,\ T2.Sid}\left(\sigma_{(T1 \times T2)\ T1.cost = T2.cost \wedge T1.Pid = T2.Pid}\right)$$

~~T1.Sid ≠ T2.Sid~~ No need to write this condition because of Key constraints because Sid & Pid are Primary key. So, Sid must not be repeated.

Que) Retrieve Sid of the Supplies who supplied most expensive cost.

Sid	Pid	cost
S1	P1	10
S2	P1	20
S3	P2	30

X

Sid	Pid	cost
S1	P1	10
S2	P1	20
S3	P2	30

Sid	Pid	cost	Sid	Pid	cost
S1	P1	10	S1	P1	10
S1	P1	10	S2	P1	20
S1	P1	10	S1	P2	30
S2	P1	20	S2	P1	10
S2	P1	20	S2	P1	20
S2	P1	20	S1	P2	30
S3	P2	30	S1	P1	10
S3	P2	30	S2	P1	20
S3	P2	30	S1	P2	30

S1 P1 10 S2 P1 20	S2 P1 20 S1 P2 30
Supplies Info who doesn't supplied most exp part	Suppliess Info who doesn't supplied least expensive part

$\rho(T1, Catalog)$ $\rho(T2, Catalog)$

$$\rho\left(T3, \overset{\wedge}{T1.Sid, \Pi.Pid, T1.Cost}\left(\sigma_{T1.Cost<T2.Cost}(T1 \times T2)\right)\right)$$

$$\overset{\wedge}{Sid}\,(Catalog - T3)$$

⇒ Supplies who supplied least expensive part

$$\rho\left(T3, \overset{\wedge}{T1.Sid, \Pi.Pid, T1.Cost}\left(\sigma_{T1.Cost>T2.Cost}(T1 \times T2)\right)\right)$$

$$\overset{\wedge}{Sid}\,(Catalog - T3)$$

Or

$$\rho\left(T3, \overset{\wedge}{T2.Sid, T2.Pid, T2.Cost}\left(\sigma_{T1.Cost<T2.Cost}(T1 \times T2)\right)\right)$$

$$\overset{\wedge}{Sid}\,(Catalog - T3)$$

Query- Retrieve suppliers who supplied 2nd most expensive part.

$\rho(T1, catalog)$ $\rho(T2, catalog)$

$$\rho\left(T3, \overset{A}{T1.sid, T1.Pid, T1.cost}\left(\sigma_{T1.cost<T2.cost}(T1 \times T2)\right)\right)$$

$\rho(T4, T3)$

$$\rho\left(T5, \overset{A}{T3.sid, T3.Pid, T3.cost}\left(\sigma_{T3.cost<T4.cost}(T4 \times T3)\right)\right)$$

$$\overset{A}{sid}(T3 - T5)$$

T3

sid	Pid	cost
S1	P1	10

T5

sid	Pid	cost
S1	P1	10
S2	P1	20

(Que) Enroll (sid, cid)

Student (sid , name, sex)

Every course Taken by at least one male & atleast one female student.

what this relation algebra query represent

$$\pi_{cid}\left(\left(\pi_{sid}\left(\sigma_{sex=F}(Student)\right) \times \pi_{cid}(Enroll)\right) - enroll\right)$$

a) course in which all F students are enrolled

b) course in which proper subset of 'F' students are enrol

c) course " " only male students are Enrolle

d) Now.

$$\pi_{cid}\left(\left(\boxed{\pi_{sid}\left(\sigma_{sex=F}(Student)\right)} \times \boxed{\pi_{cid}(Enroll)}\right) - enroll\right)$$

All female student

All course

{ (Every female student Enrolled by every course) } − { Students who are Enrolled some course }

In the question. Discard the option on the basis of the output obtained after solving that question on the table record.

Sid	Sname	Sex
S1		F
S2		F
S3		F

Sid	Cid
S1	C1
S2	C1
S3	C1
S1	C2
S2	C3
S3	C3

S1 C1
S2 X C2
S3 C3

= S1 C1, S1 C2, S1 C3, S2 C1, S2 C2, S2 C3, S3 C1, S3 C2, S3 C3

S1 C1, S2 C1, S3 C1, S1 C2, S2 C3, S3 C3

S1 C3, S2 C2, S3 C2

Course,
C2 → {S1}
C3 → {S2 S3}

Que) Student(Sid, Sname, marks, Sex)

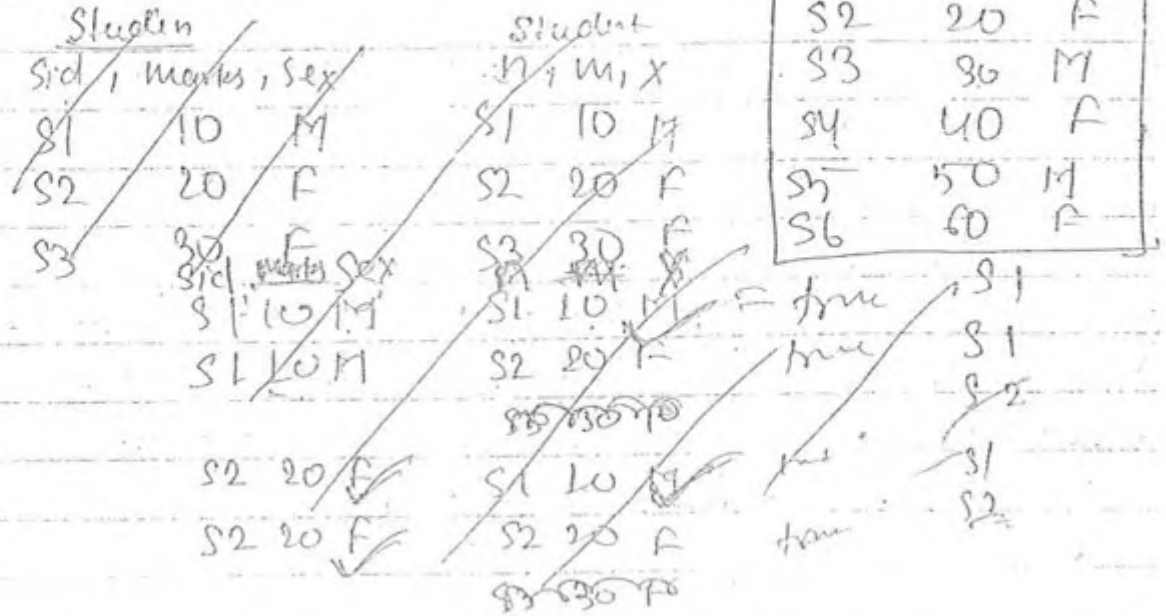
Every class there exist at least one male & one female student.

$$\Pi_{Sid}\left(\sigma_{Sex=Female}(Student)\right) - \Pi_{Sid}\left(Student \bowtie_{M,M,x} \rho_{(Student)}\right)$$

Sex = 'Female'
X = 'male'
marks <= M

All female student ~~id~~

Sid	marks	Sex
S1	10	M
S2	20	F
S3	30	M
S4	40	F
S5	50	M
S6	60	F



Soln:-

$$\pi_{Sid}\left(\sigma_{sex='female'}(Student) \bowtie_{marks<m} \sigma_{x=max}(\rho_{n,m,x}(Student))\right)$$

Sid	marks
S2	20
S4	40
S6	60

$\bowtie$

n	m
S1	10
S3	30
S5	50

Sid	marks	n	m
S2	20	S3	30
S2	20	S5	50
S4	40	S5	50

S2
S4

Comparison

>	<=
all	some
>=	<

All female student — (Female students who scored less than or equal to some male student)

a) [illegible] than some male st

b) " " not less than some male st

c) " " more marks than all male

d) " " more than equal to " "

Ans

F Student who scored more than or equal to all male students → (All female students) − (Female student who scored less than some male student)

⇒ (b), (d) is correct option

* ① $\{\forall, >\} = \} \Leftrightarrow$ (Universal set) − $(\exists\ "<")$

② $\{\forall, >\ \} \Leftrightarrow$ (Universal set) − $(\exists, <=)$

③ $\{\exists, >\} = \} \Leftrightarrow$ (Universal Set) − $\{\forall, <\}$

④ $\{\exists, >\} \Leftrightarrow$ (Universal Set) − $\{\forall, <=\}$

* $>=$ All $\Leftrightarrow$ not $<$ Some

$>$ All $\Leftrightarrow$ not $<=$ Some

$>=$ Some $\Leftrightarrow$ not $<$ all

$>$ Some $\Leftrightarrow$ not $<=$ all

R(P0, R1, R2, R3) S(P0, S1, S2)

Que) Which one of the following Query are equivalent

1) $\pi_P(R \bowtie S)$

2) $\pi_P(R) \bowtie \pi_P(S)$

3) $\pi_P(\pi_{P0}(R) \cap \pi_{P0}(S))$

4) $\pi_P(\pi_{P0}(R) - (\pi_{P0}(R) - \pi_{P0}(S)))$

1) $\pi_P(\sigma_{R.P=S.P1 \; R.0=S.0}(R \times S))$ → P: 1

R

P	0
1	2
2	3
3	3

S

P	0
1	2
2	4
4	3

2) $\pi_{R.P}(\sigma_{R.P=S.P}(\pi_P(R) \times \pi_P(S)))$ → P: 1, 2

$\frac{1}{2}$ ⋈ 1 : 3

24/08/2011

SQL :- (Structured Query language) :-

Sublanguage of SQL :-

1) DDL (Data Definition language)
⇒ modify the schemas of the DB or to manipulate the table, not the content or data of the table.

a) Create
b) Drop : (Drop Table Suppliers)
(Entire schema is deleted or we can say the entire table is de
c) Alter :- Add new columns;
remove existing columns.

Suppliers (Sid, Sname, rating)
Alter Table Suppliers add age integer;
Alter Table Suppliers remove rating;

* This language Effects physical schema.

2) DML (Data Manipulation language) :-

a) Insert
b) Delete :- Here content of the table is deleted not the table.
c) Update

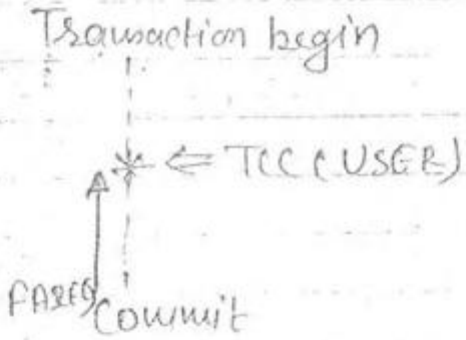
* Independent of physical & conceptual schema.

3) DCL (Data Control Language) :-

a) Rollback / Abost

b) Commit

c) Transaction Consistency checkpoint:



4) Data Query Language :-

Used to retrieve data from DB.

a) Select,
b) From,
c) where,

Query1- SELECT DISTINCT A1, A2, ——, AN
FROM R1, R2, ——, Rm
Where P

* To eliminate duplicate Tuple from result use (DISTINCT)
* whatever the data is comes in From is equal to (R1 X R2 X — X Rm)
* whatever the condition is selected is comes under where
* whatever the attribute specified in the select is equal to the projection

$$\Pi_{A1, A2 \cdots AN}\left(\sigma_{P}(R1 \times R2 \times \cdots \times Rm)\right)$$ ⇒ By default it eliminate the duplicate tuples in the result.

* (SELECT DISTINCT) Equal to projection (Π) operator in Relational Algebra.
* SELECT & FROM Clause are mandatory clause without using these clause we can't create any SQL query.

Basic SQL Query: (Operators) :-

[5] SELECT [DISTINCT] A1, A2, ——, AN
[1] FROM R1, R2, ——, Rm
[2] [WHERE P]
[3] [GROUP BY (Attribute)
[4] [HAVING condition]]
[ORDER BY (Attribute) [DESC]]

[1] FROM Clause:- Cross Product of Related Relati in FROM clause.

[2] WHERE Clause:- Select tuples those are satisfied by predicate condition 'P'.

[3] GROUP BY Clause:- Used to group the result (table) based on specified attribu

Sid	Marks	Branch
S1	50	CS
S2	60	IT
S3	50	IT
S4	70	CS
S5	80	EC
S6	NULL	NULL

Eg ⇒ Group by (Branch) : It group the entire table based on Branc

O/P:-

Sid	Marks	Branch
S1	50	CS
S4	70	CS
S2	60	IT
S3	50	IT
S5	80	EC

Aggregation Operators:-

⇒ COUNT ([DISTINCT] Attr)
⇒ SUM ([DISTINCT] Attr)
⇒ AVG ([DISTINCT] Attr)
⇒ MIN (Attr)
⇒ Max (Attr)
⇒ (Aggregation always discard Null value i.e. in every aggregation Null value is not counted)

* If count is based on single attribute then it check for null value
If count is based on entire table it count all the tuple of the table

count (mark) = 5
count (*) = 6

count (DISTINCT marks) = 4

sum (marks) = 310
SUM (DISTINCT marks) = 260

Avg (marks) = $\frac{\text{sum (marks)}}{\text{count (marks)}}$

AVG (DISTINCT marks) = $\frac{\text{SUM (DISTINCT Marks)}}{\text{Count (DISTINCT Marks)}}$

*. Arithmetic operation with Null results NULL.

i.e. NULL + 5 = NULL

Que)

RNO	Marks
1	~~10~~ 15
2	~~20~~ 25
3	~~30~~ 35
4	NULL

Update T1 set marks = marks + 5
Select Avg(marks) from T1

What the o/p of the select statement.

Sol:- Avg(marks) = $\frac{75}{3}$ = 25 = $\frac{\text{sum(marks)}}{\text{count(marks)}}$

* (AS) (rename operator):-
Rename for either attribute or Table.
*"Space" is also takes as rename operator in SQ

Query:- Select min(marks) AS minmarks / (marks) mi
FROM S

O/P:-

minmarks
50

Query:- ✗ Select Branch, min(marks)
Syntax Error from student.
O/P:- CS 50
} It is not able to display o/p in tabular format. So, it can't be displayed by SQL. So, Syntax Err

⇒ Aggregation along with other attribute not allowed in select clause.

Query1- Select Branch, min(marks)
FROM Student
GROUP BY (Branch).

O/P1- CS 50
IT 50
EC 80

Here min(marks) is applied on each group of the table.)

⇒ Aggregation along with other attribute is not allowed in SELECT clause without "Group by" clause of that specified other attribute

Query1- Select min(marks)
FROM Student
GROUP BY (Branch)

O/P1- 50
50
80

⇒ whenever we use the Group by clause. Aggregation is done on "Group By" only i.e. on group.

Query1- Select Sid, marks
① FROM Student
② GROUP BY (Branch)

O/P

S1	50
S4	70
S2	60
S3	50
S5	80

Query1- Select Sid, Avg(marks)
FROM Student
GROUP BY (Branch)

} If is not allowed

Query 1- Select Sid, marks
from Student
GROUP BY (Branch)

Syntatically allowing

Query 1- Select Sid, ~~marks~~ branch, Avg(marks)
from Student
GROUP BY (Sid, branch)

Allowed

⇒ if Group By Clause exists aggregation done for every group.

Query 1- Select min (Avg(marks))
FROM Student
GROUP BY (Branch)

O/P1-

60
55
80

* In SQL mostly two nested aggregation is meaningless.

* Having Clause is only applicable when GROUP BY Clause is in the Query i.e. without using the GROUP BY Clause if we use Having Clause is meaningless.

*Only in SQL sub Query expression we can use Group By clause not allowed in any other query.

[4] Having Clause:-

To retrieve Groups those are satisfied by Having Clause Condition.
*It is used to select the Group not the tuple)
(Group Selection condition)

Query:- Retrieve Students whose Group average marks more than 55.

```
Select *
FROM Student
GROUP BY (Branch)
    Having Avg (marks) > 50
```

O/P- Avg
60 { S1 50 CS 1st group
S4 70 CS
80 { S5 80 ES 2nd group } Satisfy the Having Clause Condition

Example:- Employee (name, sex, salary, dname)

SQL Query:-

```
① Select deptName
② From Employee          } Male Employee
  Where Sex = 'M'
③ Group By dname
④ Having Avg(sal) { Select (Avg(sal)
                    FROM Emp)
```

company avg salary

What this SQL Query Returns?

*"Where clause" condition is applied by "every tuple"
"Having clause" condition is applied for every group

Return:-
Name of department names of the male employee whose avg salary of the dept is greater than the avg salary of company.

[5] Select Clause:- It is used to Select the colum

[6] DISTINCT:- Eliminate duplicate tuple from select.

Select, DISTINCT } Equal to RA operates Π (projection)

[7] ORDER BY:- default Ascending Orders

(It is used to print the selected O/P in the order)

DESC: for decending order.

SET OPERATORS:-

UNION / UNIONALL
INTERSECT / INTERSECTALL
MINUS (OR) EXCEPT / MINUS ALL

$\pi_{Pname}\left(\sigma_{(Catalog \times \sigma_{(Parts)}\ C.Pid=P.Pid\ \cdot\ col=RED}\right) \cup \pi_{pid}\left(\sigma_{(catalog \times \sigma_{(Parts)}\ C.Pid=P.Pid\ \cdot\ col=GREEN}\right)$

Query:- Retreive Pnames of the red parts or green parts that are supplied by some supplier.

```
(Select Pname
FROM Parts P, Catalog C
where P.color = RED and
      P.Pid = C.Pid)
UNION
(Select Pname
FROM Parts P, Catalog C
where P.color = Green and
      P.Pid = C.Cid)
```

Let us assume:
O/P:- 1st Query: A A A B B C D
2nd Query: A A B C D E
Union: A B C D E

UNION : discards duplicates in Result.

A A B B C D ∪ A A B C E = A B C D E

* SQL 1992 Standard: Having clause condition should be only aggregation.
(Having clause condition for entire group).

SQL 1999 Standard: Having clause condition can be applied every tuple of each Group by using two more clause { ANY, EVERY }.

Ex:- FROM student
GROUP BY (branch)
HAVING marks < Any 5 ;
HAVING marks Every > 20

It is used to Select as { discard the entire group only.

[04]

$$\pi_{name}\left(\pi_{sid}\left(\pi_{Pid}\left(\sigma_{col=Red}(Parts)\right) \bowtie_{Sid=id} \pi(Catalog)\right) \bowtie_{Sid=Sname} \pi(Supplies)\right)$$

* Efficiency point of view all 4 Query are same. But in complexity point of view Q3 & Q4 are more complex.

Check for Commutative:-

1) π (Projection) = ✗

$$\pi_{A2}(\pi_{A1A2}(R)) \neq \pi_{A1A2}(\pi_{A2}(R))$$

2) σ (Selection) = ✓

$$\sigma_{C2}(\sigma_{C1}(R)) = \sigma_{C1}(\sigma_{C2}(R)) = \sigma_{C1 \wedge C2}(R) = (\sigma_{C1}(R)) \cap (\sigma_{C2}(R))$$

3) Cross Product:- = ✓

$$R1 \times R2 = R2 \times R1$$

4) Union (U) :- $R1 \cup R2 = R2 \cup R1$ ✓

5) Intersection ($\cap$) :- ✓

$R1 \cap R2$

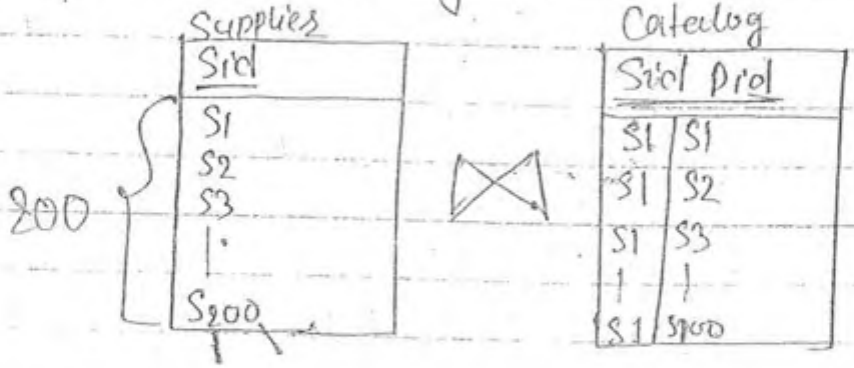
6) Set Difference (—) :- ✗

7) / (Division) :- ✗

$A(x,y) / B(y) \neq B(y) / A(x,y)$

8) Natural Join ($\bowtie$) = ✓

Query :- Assume Supplier relation consist 200 tuples & catalog relation consist 100 tuples. How many maximum no. of tuples resulted by supplies $\bowtie$ catalog.



Max no. of tuples in Sup $\bowtie$ Catalog = 100

Min no. of Tuples in Sup $\bowtie$ Catalog = 100 (bcoz catalog Sid is foreign key of supplies Sid so, Sid reference Suppliers Table minimum 100 tuples is necessary.

* M = {A, A, B, B, C, D} UNION ALL {A, A, B, C, D, D, E} = (A, A, B, B, C, D, A, A, B, C, D, E)

UNION ALL: ~~Duplicates~~ Duplicates all the tuple of the O/P.

* INTERSECTION-

Discards duplicates in result

{A, A, B, B, C, D} INTERSECT {A, A, B, C, D, D, E} = {A, B, C, D}

*common tuple of both the relation with duplicate tuples.

*INTERSECT ALL-

*~~Intersection~~ Maximum no. of common in both the set.

1 \ 3 {A, A, 2 {B, B, 1 {C, 1 {D} INTERSECT ALL 2 {A, A, 1 {B, 1 {C, 2 {D, D, 1 {E} = A, A, B, C, D, E

MINUS1- Distinct tuples from first relation those are not in second relation.

A
A
A
B
B
C
D
D
G

MINUS

A
A
B
C
D
D
E

:=

G

MINUSALL- No. of duplicates in the first relation - no. of duplicates in 2nd relation.

A
A
A
B
B
C
D
G
G

MINUS ALL

A
A
B
C
D
D
E

:=

A
B
G
G

Nested Queries:-

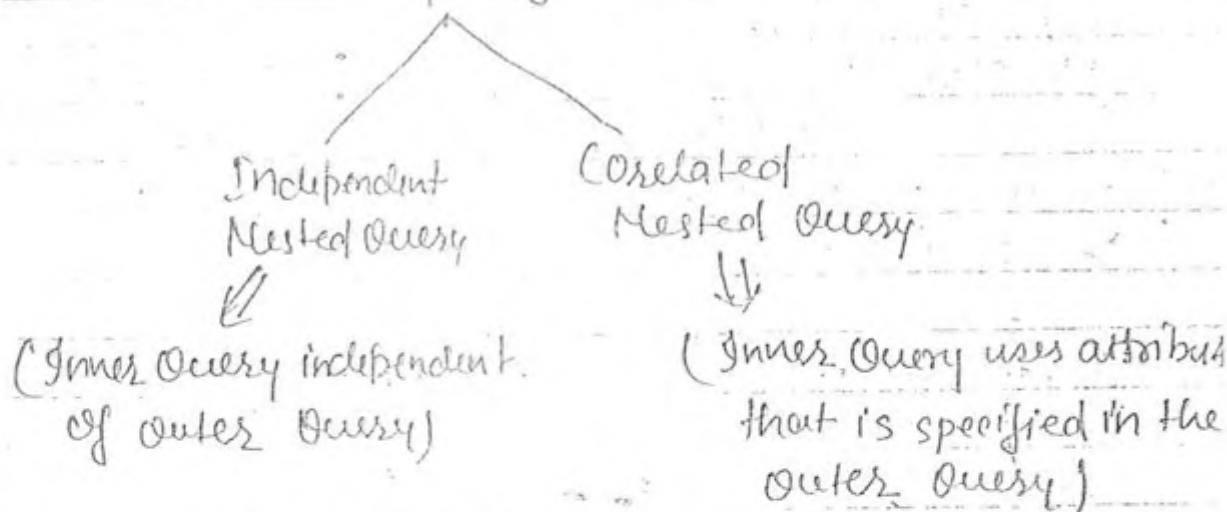
SQL supports Query inside

a) From Clause
b) Or WHERE Clause
c) Or HAVING Clause

Example:-

```
SELECT
FROM ( SELECT
       FROM
       WHERE ---( SELECT
                  FROM) )
```

There are two types of nested Query:-



Example:- Retrieve Sid of the Supplies who supplied some red part.

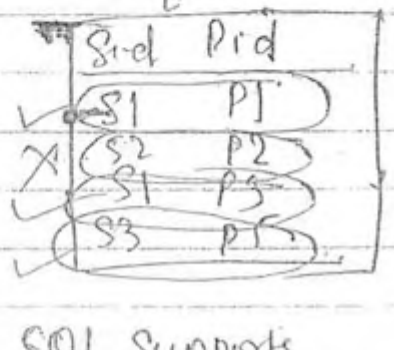
```
{Select Sid
FROM Catalog S , Parts P
where P.col = Red And
      P.Pid = S.Pid }
```

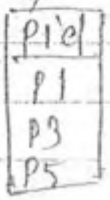
In Nested Query:-

```
Select Sid
FROM Catalog
where pid IN (SELECT PID
              FROM Parts
              where Colos = Red)
```

O/P:

S1
S1
S3





SQL Supports

IN / NOT Clause:-

IN Clause:- Used compare given tuple value exists in the set of value as not. i.e. if given value is in set of values then "IN" return "true"

Using Nested Query within FROM:-

```
Select Sid,
FROM Catalog S, (select Pid
                 FROM Parts
                 where Color=RED)P
where SPid = P.Pid
```

Query:- Emp(Eno, Ename, rating)

Dependent (ENO Dname, age)

Retrieve employee names who have no dependent.

[Q1]

```
SELECT ename
FROM EMP E, Dependent D
Where E.ENO ≠ D.ENO
```

[Q2]

```
FROM Employee
Where ENO NOT IN (SELECT ENO
                  FROM Dependent)
```

[a] Q1 correct but not Q2

[b] Q2 " " " Q1

[c] Q1 & Q2 correct.

[d] Q1 & Q2 incorrect.

Example:

Eno
e1
e2
e3
e4

Eno	Dname
e1	A
e1	B
e2	A

O/p Q1 [e1, e2, e3, e4]

Q2 [e3, e4]

Set Comparison operators:-

x IN {---} ⇒ x is in given set as result
if x is in set then it True
Otherwise false.

⇒ Op ANY

⇒ Op ALL

⇒ Op <, <=, >, >=, =, <> (not equal to)

Example:-

x > ANY {---}
returns True if x greater than
any value in the set.

x > ALL {---}
returns True if x greater than
all value in the set.

Query: Retrieve Sid's of the suppliers whose rating greater than rating of the suppliers who supplies some Red part.

Select Sid
FROM SUPPLIER
where y (Select rating
FROM SUPPLIER S, Catalog C, Parts P
where S.Sid = C.Sid and
P.Pid = C.Pid and
P.Pcolor = Red)

Select Sid
FROM SUPPLIER
WHERE rating > ANY (Select rating
FROM SUPPLIERS S, Catalog C, Parts
Where S.Pid = C.Sid and
P.Pid = C.Pid and
Colos = RED)

Query: Retrieve Sid of the Suppliers whose group average rating is more than the suppliers rati of every group of rating. Age > Max(age)

Query: Retrieve Sid of the Supplier whose age is greater than all

Query Sid's of Suppliers who are highest age.

Supplies (Sid, Sname, Age)

Select sid, age
FROM Supplies
Where age = (Select max(age)
FROM Suppliers)

* Aggregation is done on Attribute

age = max(age) is not correct

* Modification or insertion may violate integrity violation but retrival of any data does not create any integrity violation

Corelated Nested Queries:-

Example:-

```
Select Sid, Sname
FROM Suppliers S
where EXISTS (Select *
              FROM Catalog C
              where C.Sid = S.Sid)
```

Supplies

Sid
S1
S2
S3
S4

Sid	Pid	Cost
S1	P1	
S1	P2	
S2	P3	

Returns Sid, Sname of the suppliers who supply some parts.

* It is work as nested for loop i.e. we take one tuple from supplies and execute the inner query for it if it is not empty then that tuple of supplier is selected if the innerquery return false then supplier tuple is not select.

EXISTS (Class):- Returns true if inner Query results non-empty.

Ques) Student(Sid, Sname, marks)
Enrolled (Sid Cid)

No foreign key, No NULL value in the DB.

Query 1) Select Sid
FROM student
where Sid in (Select sid
From Enrolled)

S1
S2

S1 S2 S3 S2 S3

[Query 2] Select Sid S1
FROM Enrolled S1
Where Sid in (Select Sid S2
from student)

[Query 3] Select S.Sid S1
FROM Student S, Enrolled E S1
Where S.Sid = E.Sid S2

[Query 4]

Select S.Sid S1
FROM Student S S2
Where EXISTS (Select *
From Enrolled E
Where E.Sid = S.Sid)

Which is true :-

[a] all Queries returns identical row for any DB.
[b] Q1 not same as Q2 for some DB but Q2 & Q3 same for every DB.
[c] Q1 & Q2 same for any DB and Q4 returns fewer row set than Q3.
[d] Q4 encounters integrity violation at run time.

Student

Sid
S1
S2
S3
S4

Enrolled

Sid	Cid
S1	C1
S1	C2
S2	
S5	

Qeee) movie(id, title, yr)
actor(id, name)
Casting(movieid, actorid)

Write an SQL Query to return titles of the all 1959 Movie films.

Soln:-

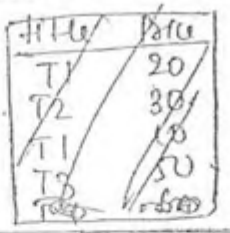
```
Select Titles
From(Movies, actor a, Casting C
where m.id = movieid and
      a.id = actor id and
      yr = 1959
      name: 'Monroe'
```

Qeee) Book(title, Price)

Assume that no two books have the same price. Or no null value.

```
SELECT title
FROM Book as B
where (Select Count(*)
       FROM Book as T
       where T.price > B.price) < 5
```

a) Title of the 4 most expensive Book
b) " " " 5th most inexpensive "
c) " " " " expensive "
d) " " " five " " "



20

(B')	Title	Price
X	B1	10
X	B2	20
✓	B3	30
✓	B4	40
✓	B5	50
✓	B6	60
✓	B7	70

(1)

* If there is no any data then "count" returns 'zero'

Que) R Account(Cid, balance)

No null values

We would like to rank Customers according to decreasing balance.

[Q1] Select A.Cid, count (B.Cid)
FROM Account A, Account B
where A.bal <= B.bal
GROUP BY A.Cid

[Q2] Select A.Cid, Count(B.Cid) ≠ 1
FROM Account A, Account B
Where A.bal < B.bal
GROUP By A.Cid

a) Q1 Correct but not Q2
b) Q2 " " " " Q1
c) Q1 & Q2 in correct bat Q1 results correct O/P for some datab but both are incorr.
d) Q1 & Q2 returns identical rowset for some database

Input

Cust	Balance
C1	10
C2	20
C3	20
C4	30
C5	30
C6	30

O/P

Cid	Rank
C5	1
C4	1
C6	1
C3	4
C2	4
C1	6

C1 3 4
C3 2 4

C1	10	C1	10	✓ 10 10
C2	20	C2	20	C1 ✓ 10 20 ✓
C3	20	C3	20	C1 ✓ 10 20 ✓
C4	30	C4	30	C1 ✓ 10 30 ✓
				× 20 10
				C2 ✓ 20 20
				C2 ✓ 20 20
				C2 ✓ 20 30 ✓
				× 20 10
				C3 ✓ 20 20
				C3 ✓ 20 20
				C3 ✓ 20 30 ✓
				× 30 10
				× 30 20
				× 30 20
				C4 ✓ 30 30

C1 10
C1 10 3
C1 10
C2 20
C2 20 3
C2 20
C3 20 3
C3 20
C3 20 1
C4 30

⇒ Query1 is correct only when balance is different or distinct but if any two customer of same balance then Query1 result is incorrect

⇒ Quer 2 is correct except for the 1st rank.

* EXISTS return True when inner condition is non empty.

Similary

"NOT EXISTS" return True when inner condition is "false" or "empty". Or return false when it is non empty.

Query:-

```
Select S.Sid
FROM suppliess S
where NOT EXISTS ( select P.Pid
                   FROM Parts P
                   where NOT EXISTS (SELECT C.Sid
                                     FROM Catalog C
                                     where C.Sid = S.Sid and
                                     C.Pid = P.Pid ))
```

Supplies (Sid, Sname)
Parts (Pid, Part, Colos)
Catalog (Sid, Pid,

```
for(i=1 to n)
  for(j=1 to n)
    for(k=1 to n)
```

Supplies

Sid
S1
S2
S3

Catalog

Sid	Pid
S1	P1
S2	P2
S1	P2
S3	P2

Parts

Pid
P1
P2
P3

(Sid) — { All parts supplied } — (Enroll)
all supplies

O/P:- It returns all the suppliess

* This above program is just work as division operation work.

25/08/2011

SOL:-

Supposts pattern matching of strings.

"%" ⇒ used to denote 0 or more characters

"_" ⇒ used to denote Any Single character

Example:- "Names" Starts with 'R'

'R%'

⇒ Names Starts with 'D' Ends with 'A' and at least 5 characters

'D___%A'

Qu) If we want to retrieve 'MADE%EASY' string.

Ans use Escape Character '\' :-

used treat patter character as like normal character.

⇒ 'MADE\%EASY'

Query:- Name Starts with 'D' Ends with 'A' and at least 5 characters.

```
Select *
From Student
where Sname LIKE 'D___%A'
```

LIKE/NOTLIKE:-

Used to compare with pattern string.
NOTLIKE is complemented value of LIKE.

NULL:- UNKNOWN or Unexisted Value.

⇒ NULL is non-zero and No two NULL Values are equal.

⇒ (NULL is set of ASCII character Randomly assigned by DBMS such that every NULL is different.)

Eno	Ename	PPno
E1	A	P1
E2	A	NULL
E3	B	P2

Query:- Retrieve Employee who have no passport.

```
Select *
FROM Employee
Where PPno = NULL
```

} It is not compare with null value

So we use IS/ISNOT Clause

⇒ IS/ISNOT:- Compare with Null Values.

```
Select *
FROM Employee
Where PPno IS NULL
```

There are three truth value in DBMS

- True ⇒ If it is '0'
- FALSE ⇒ If it is '1'
- UNKNOWN: Comparison with Null value is treated as Unknow.
  then it is 1/2 (If is not true, not false, it is between true & false)

X	OR	Y	Z = max(x, y)
1		1	1
1		0	1
0		1	1
0		0	0
0		1/2	1/2
1		1/2	1

X	Y	Z = min(x, y)
1	1	1
1	0	0
0	1	0
0	0	0
0	1/2	0
1	1/2	1/2

⇒

OR	T	F	U
T	T	T	T
F	T	F	U
U	T	U	U

AND	T	F	U
T	T	F	U
F	F	F	F
U	U	F	U

⇒ NOT UNKNOWN = UNKNOWN

Que) Relation can contain NULL values. Suppose all comparison eqs with NULL value are treated as False.
Which of the following pairs is not equalent.

a) $X = 5$ and $not(not(X = 5))$

b) $X = 5$ and $X > 4$ and $X < 6$ where X is an integer

X c) $X \neq 5$ and $not(X = 5)$

d) None

If we take Table as

X
NULL
5
6

Check one by one the L.H.S & R.H.S in the above questn we obtain the values by pu

a) $X = 5$ → X: [5] ; $not(not(X = 5))$ → X: [5]

b) $X \neq 5$ → X: [5] ; $X > 4$ and $X < 6$ → X: [5]

c) $X \neq 5$ → X: [6] ; $not(X = 5)$ → X: [NULL, 6]

Transaction and Concurrency Control:-

Transaction:-

Set of logically related operations used to perform unit of work.

Example:-

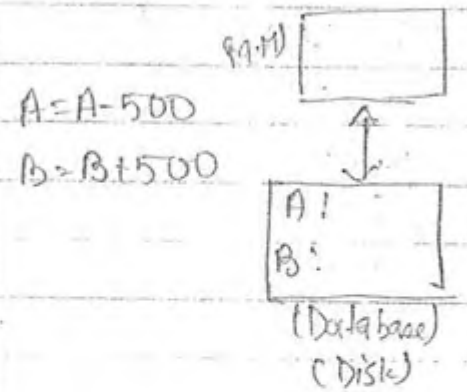
T1: Transfer 500 Rs from Account A to Account

Read (A):-

Accessing the data item A from secondary memory (or Database) to programmed variable.

Write (A):-

Updates data items A into the database.




```
Begin Trans
R(A)                R(A)
A=A-500             W(A)
W(A)          ⇒     R(B)
R(B)                W(B)
B=B+500             Commit
W(B)
Commit
```

I/O operations:- R(A), R(B), W(A), W(B)

Memory operations:- A-500, B+500

Commit:- Transaction executed sucessfully.

* Every Read/Write operation is I/O operation
* Commit is data control operation. So, it is mandatory.

To Preserve Consistency Transaction should follow "ACID" properties.

A: Atomicity :- Execute all operations of the transaction Or none of them.

Transaction Failure May be becoz. of
1) Power failure
2) S/W failure
3) H/W failure
4) Disk Crash. Etc.

⇒ Recovery Management component performs the Rollbacking Or Abort.
⇒ Atomicity is done by DBMS.

Rollback / Abort :-
Undo the modifications that are done so far from failure point to the beginning of Transaction. It it in sequence order not arbitra

Example :-

If A = 1000 → 500 → 1000 (rollback)
B = 2000

	begin Trans
	R(A)
	A = A - 500
	W(A)
Failed	R(B)
	B = B + 500
	W(B)
	Commit

i.e. rollback is done with the help of Transaction log file.

Transaction log:-

File created by DBMS for every Transaction to keep track of transaction operations.

T1 log
Undo
Redo

"used to perform Rollbacking."

⇒ The main purpose of transaction log is Rollbacking

Undo: operation:-

Database is updated by old values of the log file.

Redo: operation:-

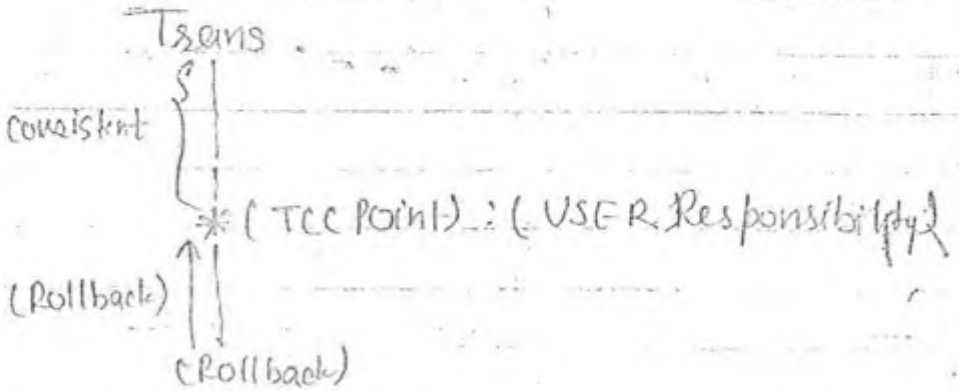
Updating the database by new values.

"OR"

Database updated with New Values.

⇒ Log file retained in the DB until transaction committed or rollbacking completed, then after the log file is automatically removed.

Transaction Consistent (check Point):-



C: Consistency:-

DB should be consistent before & after execution of the Transaction.

⇒ Consistency is checked by Uses i.e. it's user responsibility.

⇒ Either Atomicity failed OR Isolation failed then consistency is going to failed.

I: Isolation:-

Schedule:-

Time order sequence of two or more transaction.

1) Serial Schedule:-
After commit of one transaction then begin the other transaction.

2) Concurrent Schedule.

Example:- T1: Trans 500 from A to B
T2: display total balance R(A) R(B) : A+B

i.e. R(A)
R(B)
display(A+B)

Serial schedule:-

T1	T2
R(A)	
W(A)	
R(B)	
W(B)	
	R(A): 500
	R(B): 2500
	3000

T1 : T2

T1	T2
	R(A): 1000
	R(B): 2000
	3000
R(A)	
W(A)	
R(B)	
W(B)	

T2 T1

⇒ i.e. If n Trans then
n! Serial schedule

⇒ All serial schedules are consistent.

⇒ Less throughput:-

T1
R(A)
I/O operation, DMA Busy } (CPU idle)
W(A)
DMA Busy } Again (CPU idle) time when one trans. chanis done

because of this i.e. because most of the CPU is idle due to the I/O operation.:—

⇒ Poor Resource Utilization
⇒ More Response time.

Concurrent Schedule:-

Interleaved execution or simultaneous execution of two or more transactions

T1	T2
R(A)	
W(A)	
	R(A) 500
R(B)	
W(B)	
	R(B) 2500
	3000

T1	T2
R(A)	
W(A)	
	R(A) 500
	R(B) 2000
	2500
R(B)	
W(B)	

[Consistent] (Equal to T1:T2 Serial) [Inconsistent] $T_1 \rightarrow T_2$ ✗ (not equal)
$T_2 \rightarrow T_1$ ✗

Advantage:- 1) More throughput
2) Less Response time
3) Better Resource Utilization.

Disadvantages:-
Concurrent execution may result in inconsistency.

Que) How many Total Schedules are possible with T1 & T2 Transaction:-

$\frac{6!}{4!2!}$ ..... $\frac{6!}{4!*2!}$ ..... (By Permutation) i.e. ${}^6C_4 * 4!$
$= {}^6C_4 *$

Assume T_1 & T_2 are m, n operations each:- Total possible schedules :-

$$(m+n)C_m \text{ (OR) } \frac{(m+n)!}{m! * n!}$$

Assume T_1, T_2, T_3 are m, n, p operations each. Then Total possible schedule :-

$$(m+n+p)C_m * (n+p)C_n * 1 = \frac{(m+n+p)!}{m!\,n!\,p!}$$

$\Rightarrow$ if concurrent schedule execution not equal to any serial then it is Inconsistent.

[***]

I: Isolation :-

Concurrent execution of two or more transaction should be equal to any serial schedule then it is consistent as we called it as (Serializable schedule). then it is care taken by

$\Rightarrow$ Concurrency control management component

D: Durability:- Database should be recoverable under any kind of failure.

RAID 1- Redundant Array of Independent Disks.

RAID 0:- Redundant Array of Independent Disks not durable

RAID 1:- Mirror Image

It is durable

Disadvantage: write operation is too costly in RAID 1.

Problems Because of Concurrent Execution:-

[1] WR Problem :- [Read After Write Problem]

T1	T2
W(A)	
⋮	R(A): Uncommitted Read (Or) Dirty Read. (R(A) read da when W(A) i not committe this transact

T1	T2
1000 R(A)	
500 W(A)	500 R(A): Uncommitted Read
	R(B) 2000
2000 R(B)	
2500 W(B)	

Inconsistent (WR Problem)

T1	T2
1000 R(A)	
W(A)	500 R(A): Uncon Rea
2000 R(B)	
2500 W(B)	2500 R(B)

Consistent (No R

T1	T2
	R(A) 3000
1000 R(A)	
500 W(A)	
	R(B) 2000
2000 R(B)	
2500 W(B)	

Consistent

We can't found that which problem is exist in this transaction due to concurrency i.e. Problem is not just found after seeing the transaction. Firstly we have to check that if Transaction is consistent then no problem, but if the Transaction is not consistent then after we check, due to which problem it is inconsistent or to be found i.e. problem is checked after checking the transaction is inconsistent.

[2] RW Problem [Write After Read Problem]

T1	T2
R(A)	
	W(A)
Rollback	

T1 performed Read(A) before commit / Rollback of T1, trans T2 updated data item A.

Example:-

A: No. of copies of Textbook. To issue one book from A book then what operation are required during transaction.

```
T
R(A)
if (A>0)
   A=A-1
   W(A)
   commit
```

If Transaction T1 & T2 both simultaneously request for the same book. the transaction operation

T1	T2
R(A)	
if (A>0)	
A=A-1	
	R(A)
	if (A>0)
	A=A-1
	W(A)
	Commit
W(A)	
Commit	

if initially A = ~~10~~ 9 (due to RW problem)
due to transaction T2 due to transaction T1

Inconsistent (RW Problem Exist)

	T1	T2
A=10	R(A)	
		W(A)
Avg	R(A)	

It is called temporary read problem bcoz either their is no any updation in even after some time if it rea then it get the other value previous. Because there may any other transaction T2 w updated the previous value read by the transactio (T1).

(Temporary Read Problem)

OR

(Phantom phenomena problem)

[3] WW Problem :- [Write After Write Problem]

T1	T2
W(A)	
	W(A)

(Write-Write Problem is possible)

Example- T1: Set A, B values as 1000 (W(A), W(B)

T2: Set A, B values as 2000 (W(A), W(B)

If T1: T2 Serial: A = B = 2000
(Firstly Trans T1 executed then T2 executed)

T2: T1 Serial: A = B = 1000
(Firstly Transaction complete its execution then T1 starts)

T1	T2
W(A)	
	W(A)
	W(B)
W(B)	

A = 1000, 2000
B = 2000, 1000

(Non-Serizable) as Inconsistent Because of WW Problem

T1	T2
W(A)	
	W(A)
W(B)	
	W(B)

A = 1000, 2000
B = 1000, 2000

(Serializable) (No concurrent Execution Problem)

* when the transaction is non-serizable then only WR Problem, RW Problem, WW Problem are reasons for non serializable & Inconsistency.

* Some times WW Problem is LOST UPDADATE PROBL

⇒ LOST UPDATE PROBLEM1-

A = 0, ~~10~~, 20 ○

T1	T2
R(A)	
W(A)	
↑	W(A) ⇐ (lost update problem becoz of T1 Rollback)
(FAILED)	

Rollback

* Even if the schedule is serializable Lost Update Problem possible is still possible.

26/08/2011

Problem becoz of concurrent execution!-

⇒ If the schedule is non-serializable Then RW, WR, WW Problems may exists.

⇒ If the schedule is serializable then schedule free from RW, WR, WW Problems. But lost Update Problem may possible.

Serizable Schedule!-

⇒ Schedule which satisfy isolation property.

⇒ Concurrent execution b/w two or more transaction should be equal to any serial schedule.

Classification of the Schedule:-

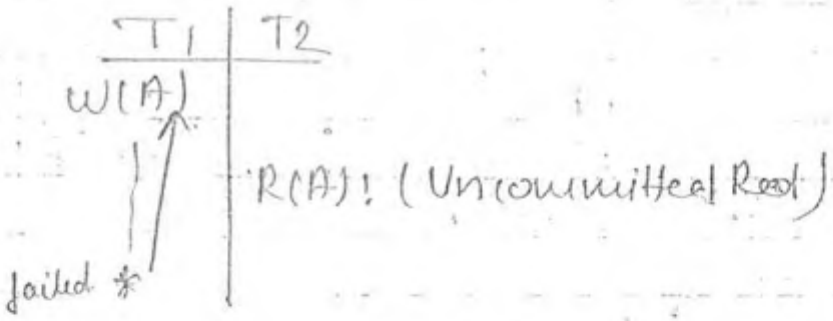
Two classification are possible. Based on:-

* Recoverability
* Sesiglizability

* Recoverability:-

If Trans T2 read the data item A that is updated by Uncommitted Trans T1.

$\Rightarrow$ (T2 depending on T1 result



Which of them are dependening on other

1) T1: W(A), C | T2: R(A)

2) T1: R(A) | T2: W(A)

3) T1: W(A) | T2: W(B)

} NO dependency b/w Transaction

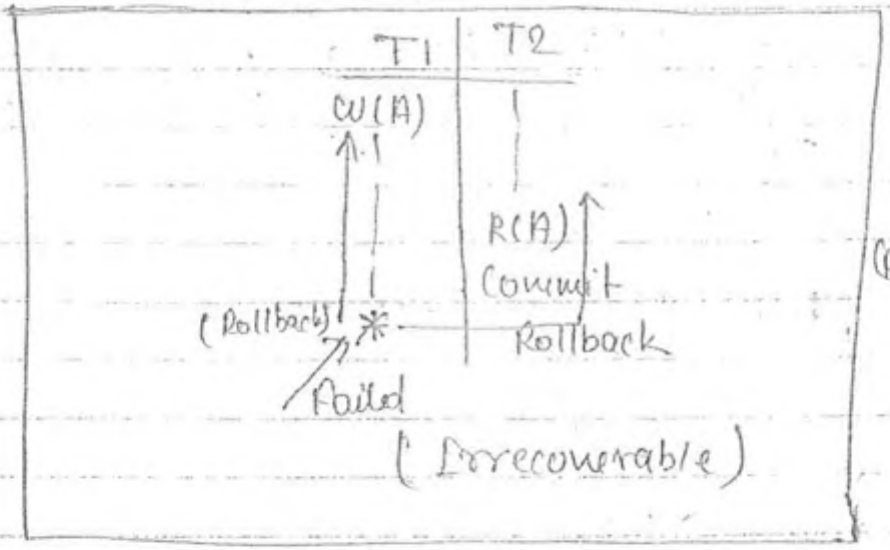
(If two transaction updated simultaneously that does not mean that there is a dependency between them).

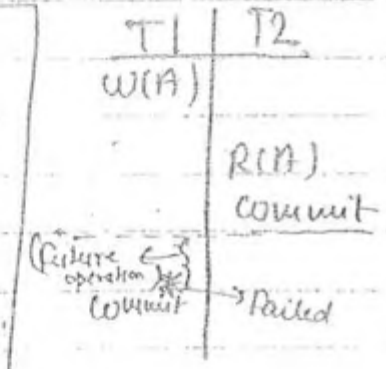
T1	T2
W(A)	
	R(A)
	W(B)
R(B)	

Dependency exist

(1) Irrecoverable Schedule :-

Rollbacking of committed transaction.





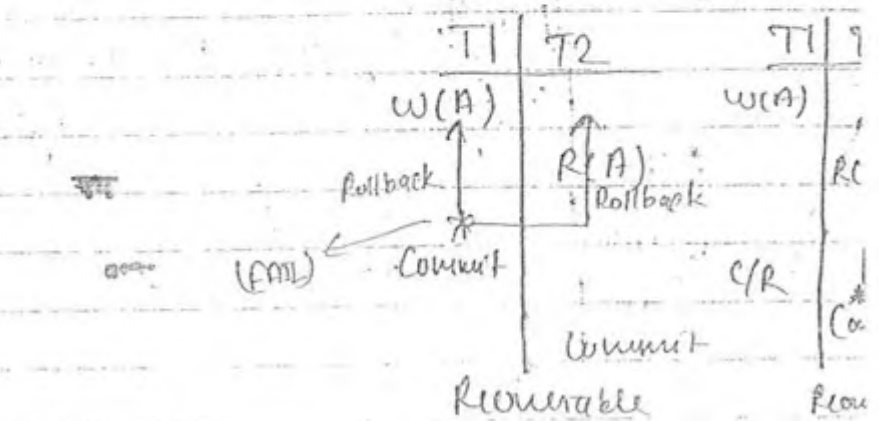
T1	T2
W(A)	
	R(A)
	Commit
Commit/Rollback	

Transaction T2 Reads the data item A updated by some other Transaction T1 if commit operation of T2 before the commit/Rollback of T1 then schedule is irrecoverable.

[1] Recoverable schedule :-

Rollbacking of only uncommi transaction.

T1	T2
W(A)	
	R(A)
Commit/Rollback	
	Commit



If T2 Reads the dataitem A updated by T1. Then commit of T2 should be delayed until C/R of T1

→ If both the Transaction depending on each oth then whatever the order of dependency, in each case it is irrecoverable.

T1	T2
W(A)	
	R(A)
	W(B)
R(B)	
Commit	
	Commit

→ It is recoverable

→ It is not recoverable because it not satisfy the recoverable condition

(Irrecoverable)

T1	T2
R(A)	
W(A)	
	R(A)
	W(B)
R(B)	
W(B)	
C1	
	C2

Dependency

Dependency cycle is present

Recoverable but not Serializable (WR Problem)

T1	T2
W(A)	
	W(A)
	W(B)
W(B)	
C1	
	C2

(NO Dependency)
Recoverable schedule but not Serializable schedule
(WW Problem Exist)

T1	T2
R(A)	
	R(A)
	W(A)
	Commit
W(A)	
Commit	

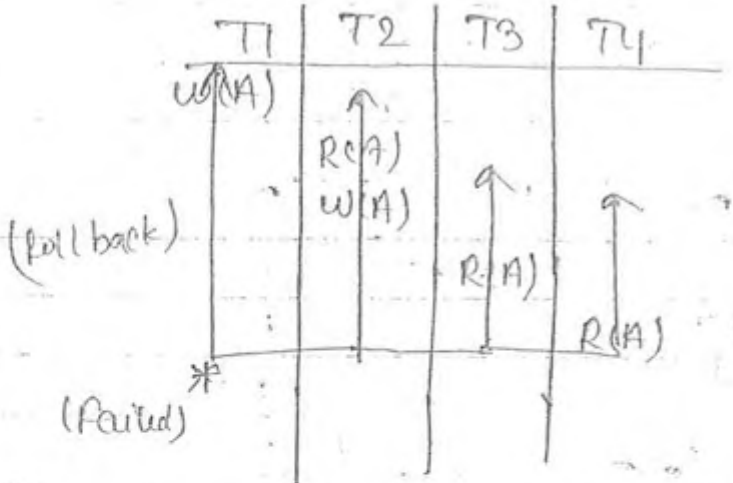
No Dependency So it is Recoverable schedule but not Serializable (RW Problem)

T1	T2
W(A)	
	W(A)
Rollback	
	Commit/Rollback

Recoverable Schedule but lost Update Problem exist.

⇒ Recoverable Schedule May not free from RW, WR, WW Problems and lost update Problem and Cascading Rollback Problem

Cascading Rollbacking!-



(Recoverable but Cascading Rollback

Here T2 depending on T1, T3 depending on T2 & T4 depending on T3. If there is any failure occur in T1 before the commit then we roll back T1 along with we roll back all the dependending transaction must be roll backed.

⇒ it results wastage of CPU execution time and I/O operations

[3] Cascadless Rollbacking Recoverable Schedule:-

To remove cascading rollbacking problem we use the cascadless rollbacking i.e. no transaction is depending on other transaction.

T1	T2
W(A)	
C/R	
	R(A)

{ Uncommitted Read not allowed in cascadless Rollbacking Recoverable Schedule }

T1	T2
W(A)	
	W(A)
	W(B)
C(B)	
C1	
	C2

(WW Problem) is exist

T1	T2
R(A)	
	R(A)
	W(A)
W(A)	
C1	
	C2

(RW Problem) is exist

T1	T2
W(A)	
	R(B)
	R(C)
	W(C)
W(C)	
C1	
	C2

⇒ WR Problem not possible there and cascading Rollback are not possible but still WW, RW are possible.
Lost update problem is still possible.

[4] Strict Recoverable Schedule:-

To remove the lost update problem Strict Recoverable schedule is come, in this schedule we avoide simultaneously writefrom any two transaction. due to this WW problem also recoverable removed.

T1	T2
W(A)	
C/R	
	R(A)/W(A)

if Trans T1 updates the data item A Then some other transaction T2 not allowed to R(A)/W(A) Until C/R of T1.

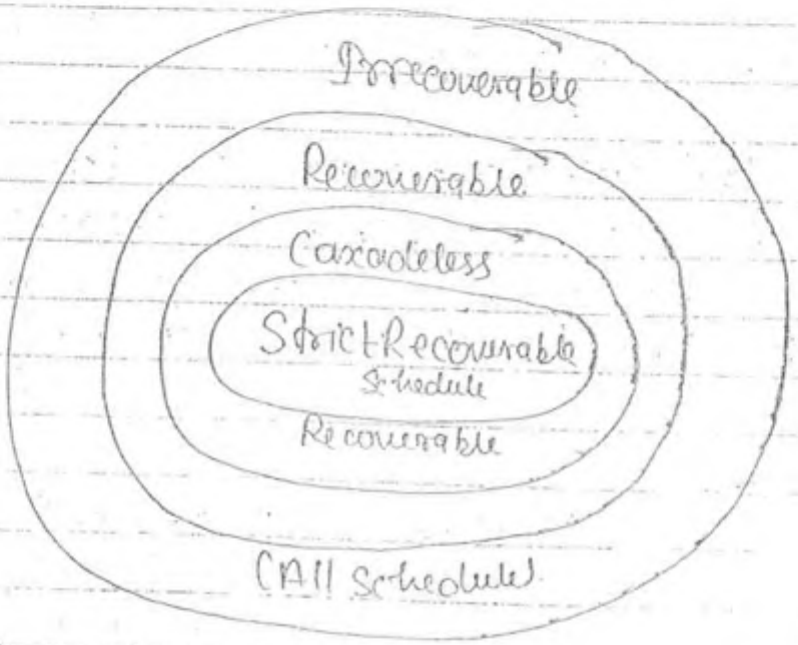
T1	T2
R(A)	
	R(A)
	W(A)
	C2
W(A)	
C1	

This schedule satisfied Strict Recoverable condition but non-serializable schedule due to RW problem.

⇒ Strict Recoverable Problem is free from WW, WR, Problem and also lost update, Cascading Rollback problem.

But RW problem may also exist here.

*



* $W_1(A) W_2(A) W_2(B) W_1(B) C_1 C_2$

(operation, data item, Transaction No)

Que) S: $S_1(X) S_2(Z) S_1(Z) S_3(Z) S_3(Y) W_1(X) W_3(Y) S_2(Y) W_2(Z) W_3(Y) C_1 C_2 C_3$

Check which higher recoverability is present here.

T1	T2	T3
$r_1(x)$		
	$r_2(z)$	
$r_1(z)$		
		$r_3(z)$
		$r_3(y)$
$w_1(x)$		
		$w_3(y)$ → dependency exist
	$r_2(y)$	
	$w_2(z)$	
	$w_2(y)$	
C1		
	C2	C3

⇒ It is irrecoverable.

Que) $w_1(x)\, w_3(y)\, C3\, r_2(y)\, w_2(z)\, w_2(y)\, C1\, C2$

⇒ Recoverable ✓
Cascadless ✓
Strict Recoverable ✓

Que) S21 - $r_1(x)\, r_2(z)\, r_3(x)\, r_4(z)\, r_2(y)\, r_3(y)\, w_1(x)\, C1\, w_2(z)\, w_3(y)\, w_2(y)\, C3\, C2$

$w_1(x)$ C1	$w_2(z)$ $w_2(y)$ C2	$w_3(y)$ C3

Cascadless but not strict

Ques3: $S_3(X) S_1(X) W_3(X) S_2(X) W_1(Y) S_2(Y) W_2(Y) C_3 C_1 C_2$

RECOVERABLE ✓

Cascadless ✗

S4: $S_1(X) W_1(X) W_2(X) S_2(X) S_1(Y) W_2(Y) C_1 C_2$

T1	T2	
R(X)		No dependency
W1(X)		
	W2(X)	REC ✓
	S2(X)	CAS ✓
S1(Y)		Strict ✗
	W2(Y)	
C1		
	C2	

Ques) T1 operations is

T1
R(A)
W(A)
R1(B)
W1(B)
C1

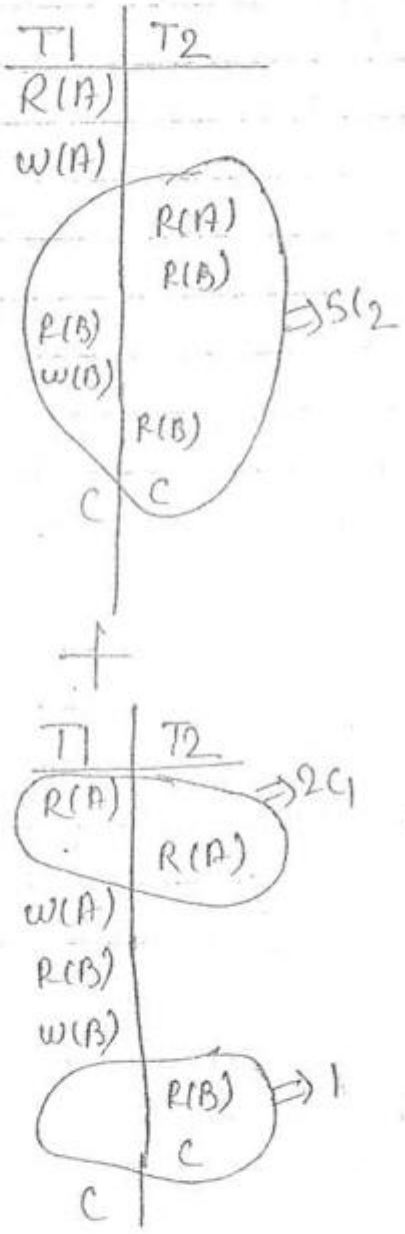
T2 operation is

T2
R2(A)
2(B)
C2

Total no. of schedules = $8C_5 = \frac{8!}{5!3!} = \frac{8 \times 7 \times 6 \times 5!}{5! \times 6} = 56$

No. of recoverable schedule:-

Example:-



= All schedule - Irrecoverable schedu

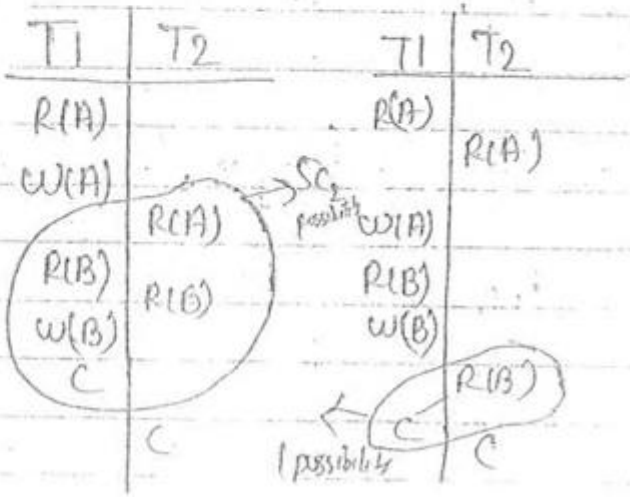
$= {}^{8}C_5 - {}^{5}C_2 - ({}^{2}C_1 * 1)$

$= 56 - 10 - 2$

$= 44$

Nog Cascadless Recoverable Schedule :-

= (Recoverable Schedule) – (Cascading Rollback)



⇒ if two schedules are conflict equalent means that both schedules are logically equalent.

⇒ if schedules are non-conflict equalent then may or may not exped logically equalent.

Example:-

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

A = ~~10~~ 20 30

(S1) A = T3

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

A = ~~20~~ 10 30

(S2) A = T3

S1 and S2 are ~~not~~ logically equalent becoz S2 resulted after swapping conflict pairs in S1

So it is Not conflict Equale

evenly ~~is not~~ S1 & S2 are equalent schedule.

Qu:

T1	T2
R(A)	
W(A)	
	R(A)
R(B)	R(B)
W(B)	

S1

T1	T2
R(A)	
W(A)	
	R(A)
R(B)	
	R(B)
W(B)	

S2

S1 is logically equalent to S2 becoz we swapping two consequective non conflict pairs in the schedule

Note 2008

Que S1: $R_2(A)\ W_2(A)\ R_3(C)\ W_2(B)\ W_3(A)\ W_3(C)\ R_1(A)\ R_1(B)\ W_1(A)\ W_1(B)$

S2: $R_3(C)\ R_2(A)\ W_2(A)\ W_2(B)\ W_3(A)\ R_1(A)\ R_1(B)\ W_1(A)\ W_1(B)\ W_3(C)$

S3: $R_2(A)\ R_3(C)\ W_3(A)\ W_2(A)\ W_2(B)\ W_3(C)\ R_1(A)\ R_1(B)\ W_1(A)\ W_1(B)$

Which one of them are conflict equalent schedules-

T1	T3
$R_2(A)$	
$W_2(A)$	
	$R_3(C)$

T2	T3
$R_2(A)$	
$W_2(A)$	
	$R_3(C)$

⇒ S1 & S2 are conflict equalent

S2: $R_3(C)\ R_2(A)\ W_2(A)\ W_2(B)\ W_3(A)\ R_1(A)\ R_1(B)\ W_1(A)\ W_1(B)\ W_3(C)$

(conflict pair)

$R_2(A)\ R_3(C)\ W_3(A)\ W_2(A)\ W_2(B)\ W_3(C)\ R_1(A)\ R_1(B)\ W_1(A)\ W_1(B)$

T2	T3
$W_1(A)$	
$W_1(B)$	$W(A)$
$W(B)$	$W(A)$

T1	T3
	$W_3(A)$
$W_2(A)$	
$W_2(B)$	

Conflict Serializable Schedule-

Schedule is said to be conflict Serializable schedule only if there should be conflict Equalent Serial Schedule.

Que Check the given schedule is conflict equalent!-

T1	T2
$R_1(A)$	
$W_1(A)$	
	$R_2(A)$
	$R_2(B)$
$R_1(B)$	
$W_1(B)$	

$T1 \rightarrow T2$ ✗ (Not Possible)

only if we are able to swap this pair to above

T1	T2
R(A)	
W(A)	
	R(A)
	R(B)
R(B)	
W(B)	

between conflict pairs $W_1(A)$ & $R_2(A)$ and between $R_2(B)$ & $BU_1(B)$ $T_1 \rightarrow T_2$ & $T_2 \rightarrow T_1$

$T2 \rightarrow T1$ ✗ (Not possible)

but not possible becoz there is a conflict pairs

(T1 followed by T2) is $T1 \rightarrow T2$ ✗ not possible

(T2 followed by T1) is also not possible i.e. $T2 \rightarrow T1$ ✗.

(Que).

T1	T2
R(A)	
W(A)	
	R(A)
R(B)	
W(B)	
	R(B)

between conflict pairs
W(A) & R(A)
$T1 \rightarrow T2$ &
W(B) & R(B)
$T1 \rightarrow T2$
So it is conflict equivalent because there is no any cycle

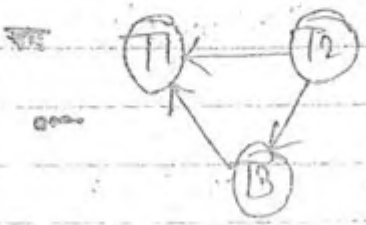
Conflict Serializable schedule

becoz equalent serial schedule is possible

T1	T2
R(A)	
W(A)	
R(B)	R(A)
W(B)	
	R(A)
	R(B)

Que) $R_2(A)\,W_2(A)\,R_3(C)\,W_2(B)\,W_3(A)\,W_3(C)\,R_1(A)\,R_1(B)\,W_1(A)\,W_1(B)$

$T2 \rightarrow T3$	$R2(A)$ & $W3(A)$
$T2 \rightarrow T1$	$W2(A)$ & $R1(A)$
$T2 \rightarrow T1$	$W2(B)$ & $R1(B)$
$T3 \rightarrow T1$	$W3(A)$ & $R1(A)$



$T2 \rightarrow (T1\ T3)$

$T2 \rightarrow T3 \rightarrow T1$ (conflict Serializable Schedule)

Que) $R_2(A)\,R_3(C)\,W_3(A)\,W_2(A)\,W_2(B)\,W_3(C)\,W_1(A)\,R_1(B)\,W_1(A)\,W_1(B)$

$R2(A)$ & $W3(A)$	$2 \rightarrow 3$
$W3(A)$ & $R1(A)$	$3 \rightarrow 1$
$W3(A)$ & $W2(A)$	$3 \rightarrow 2$

$2 \rightarrow 3$ & $3 \rightarrow 2$ (non conflict Serializable schedule).

Precedence graphs–

Graph $G = (V, E)$

Vertices (V): Transaction of the schedule.

Edges (E): 

We can draw edge between Ti to Tj only if there exist conflict pair between Ti & Tj such that Ti precoded then Tj.

i.e. if $T_i : R(A)$ $T_j : W(A)$

$T_i : W(A)$ $T_j : R(A)$

$T_i : W(A)$ $T_j : W(A)$

Testing Condition:-

if Precedence graph is Cyclic ⇒ Non conflict Serializabl

if Precedence graph is Acyclic ⇒ Conflict serializab

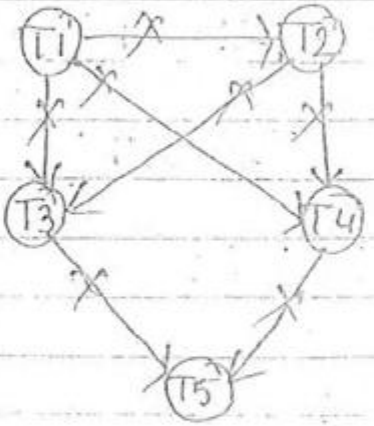
⇒ Equivalent Serial Schedule based on topological (sequence) order of acyclic Precedence graph.

F 27/08/14

Topological Osorts:-

[1] Visit vertex (v) with indegree '0' and delete visited vertex (v) along with outdegree edges of v.

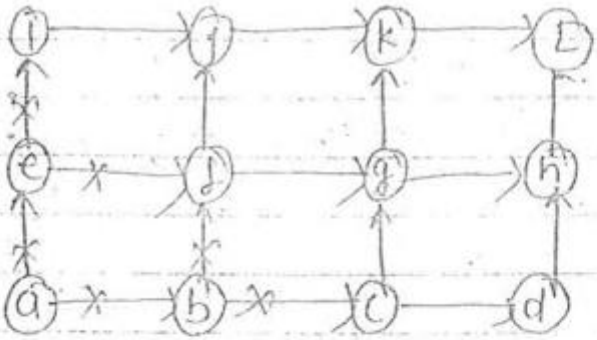
[2] Repeat (1) step until graph becomes empty.



These are the two topological sequences of given graph.

T1 : T2 : T3 : T4 : T5

T1 : T2 : T4 : T3 : T5



a : b : f : c : d : h :

a : b : c :

a : e : f :

a : e : i :

a : b : f : c : g : k :

a : b : f : c : g : h :

Based on Serializability:-

1) Conflict Serializable:-
2) View Serializable:

Conflict Serializable Schedule:-

Non Conflict Pairs:-

After swapping of two operation of two different transaction then there is no any sort of change in old schedule due to the resultant schedule. Then that operation is no conflict Pair.

[a] T1: R(A) T2: R(A) : Non conflict Pair

T1	T2
R(A)	
	R(A)

=

T1	T2
	R(A)
R(A)	

S1 = S2 because ⇒ S1 = S2

[b] T1: R(A) T2: W(A) : Conflict Pair.

T1	T2
R(A)	
	W(A)

T1	T2
	W(A)
R(A)	

S1 ≠ S2 ⇒ S1 ≠ S2

[c] T1: W(A) T2: R(A) : Conflict Pair

T1	T2
W(A)	
	R(A)

T1	T2
	R(A)
W(A)	

[d] T1: W(A) T2: W(A) : Conflict Pair

T1	T2
W(A)	
	W(A)

Final updation A is done by T2

T1	T2
	W(A)
W(A)	

Final updation of A is done by T1

$\Rightarrow S1 \neq S2$

[e] T1: R(A)/W(A) T2: R(B)/W(B) : Non Conflict

Conflict Pair:-

Pair of operations said to be conflict only if

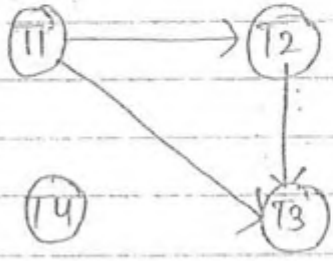
1) At least one write operation
and 2) on same data item
and 3) b/w different transaction

Conflict Equalent Schedule:-

S1 & S2 said to be conflict Equalent only if S2 resulted after swapping consecutive non conflict pairs from S1.

Q.) Identify Conflict Equalent Serial Schedulet-

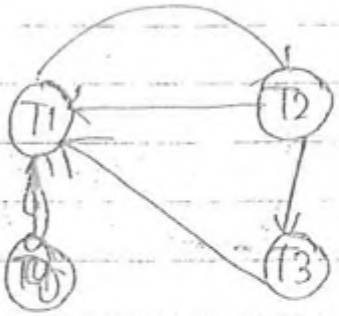
[1] $S_1(A) S_2(A) S_3(A) S_4(A) W_1(B) W_2(B) W_3(B)$



T4 : T1 : T2 : T3
T1 : T4 : T2 : T3
T1 : T2 : T4 : T3
T1 : T2 : T3 : T4

} Conflict Equalent Serial Schedule

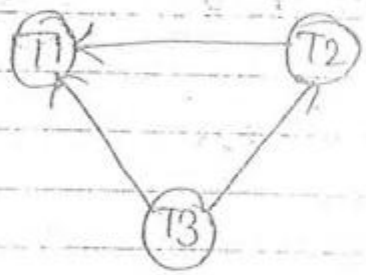
[2] $S_2(z) S_2(y) W_2(y) S_3(y) S_3(z) S_1(x) W_1(x) W_3(y) S_2(x) S_1(y) W_1(y) W_2(x)$



Non Conflict Equalent Schedule

[3] $R_1(A)\ R_2(A)\ R_3(B)\ W_1(A)\ R_3(C)\ R_2(B)\ W_2(B)\ W_1(C)$

$R_2(A) \rightarrow W_1(A)$
$R_3(B) \rightarrow W_2(B)$
$R_3(C) \rightarrow W_1(C)$



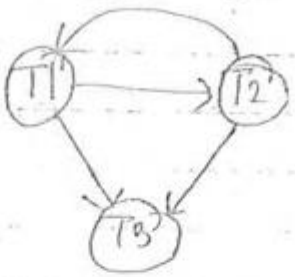
Conflict Serializable Schedule

T3 : T2 : T1

* If conflict schedule is serializable then only we can say that it is conflict serializable schedule.

Q.) S1:

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



∴ Non conflict Serializable schedule:-
If we execute this schedule in serial way!-

S2:

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

S2

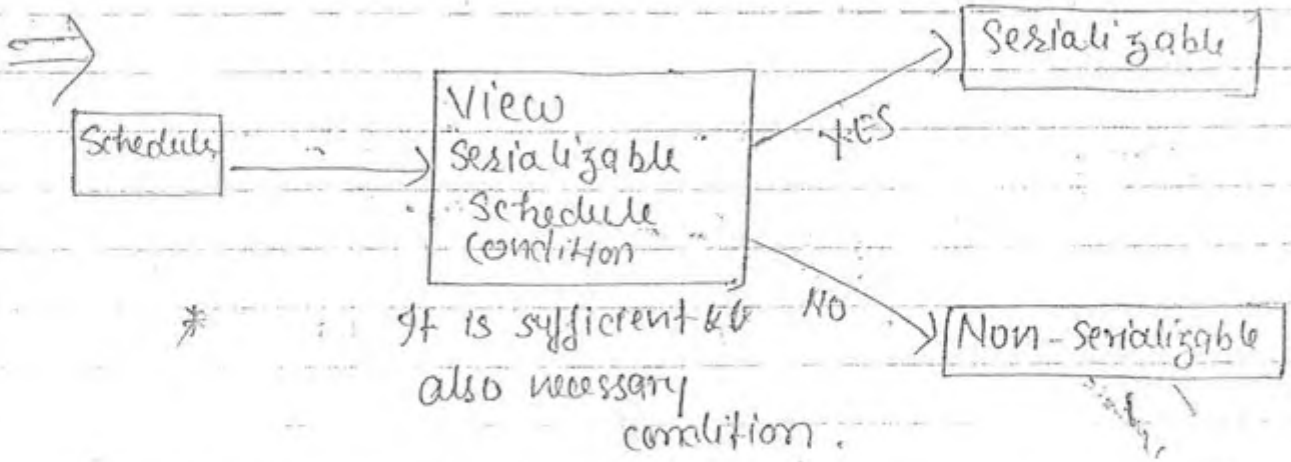
S1 = S2 i.e. schedule are equal but not conflict equivalent.

S2 is equalent sesial schedule of S1, so, it is equalent serial schedule ⇒ (Serializable Schedule).

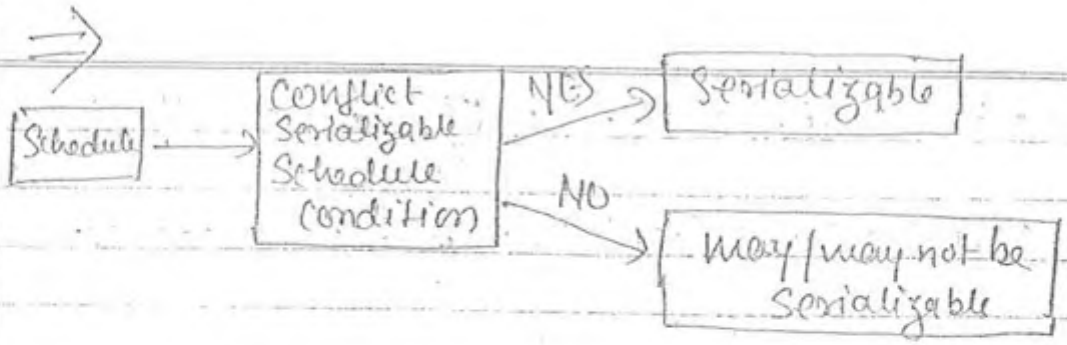
⇒ if (Acyclic Precedence graph)
{
conflict serializable
(Serializable)
} // It is sufficient to say that schedule is serializab

else
{ Not conflict Serializable Schedule
may / may not be Serializable
} // Not Necessary condition to say non serializable

* It is only sufficient but not necessary.



Exponential Time complexity
(NP-Complete problems)

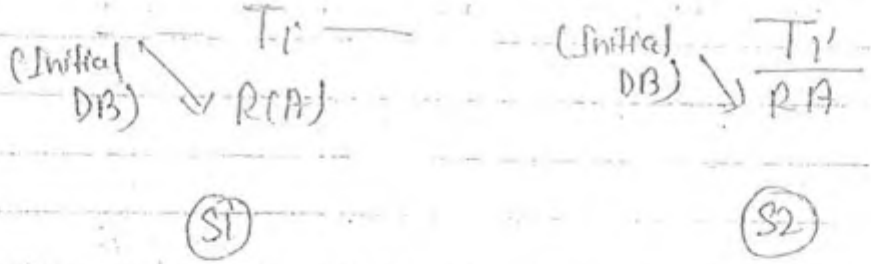


⇒ Polynomial Time Complexity $O(n^2)$

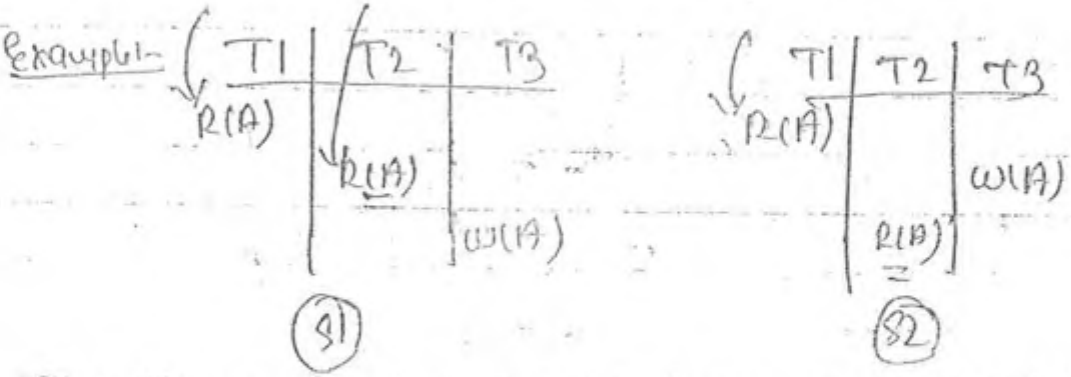
View Serializable :-

S1 & S2 are view equalent only if following condition are true :-

[1] Initial Reads of S1 & S2 should be same.



Example :-



S1 & S2 are not equalent bcoz in S1 $R_1(A)$ & $R_2(A)$ are initially read data item from Data Base but in S2 $R_2(A)$ read the data item updated by $T_3(A)$. So it is not equalent $S1 \neq S2$

[2] → Write, Read Sequence should be same in the S1 & S2 schedu

Ti	Tj
	W(A)
R(A)	

(S1)

Ti	Tj
	W(A)
R(A)	

Example:-

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

(S1)

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

(S2)

⇒ S1 ≠ S2 becoz here in S1 R3(A) read the dataitem updated by W2(A)
But in S2 R3(A) read the dataitem updated by W1(A).
So, S1 ≠ S2 because Write & Read sequence not same.

[3] Final Write should be same.

Ti	Ti
W(A)	W(A)
↑ (final write)	↑ (final write)
(S1)	(S2)

Example:-

T1	T2	T3
w(A)		
	w(A)	
		R(A)

(S1)

T1	T2	T3
	w(A)	
w(A)		
		R(A)

(S2)

S1 ≠ S2 because in S1 final updation is done by $w_2(A)$ i.e. by transaction T2
but in S2 final updation is done by $w_1(A)$ i.e. by transaction T1.
So, S1 ≠ S2 because final write is not same

View Serializable Schedule:-

View equalent of given schedule should be any serial.

*

i.e. every conflict S.S is view serializable but every view serializable may or may not be conflict Serializable.

Example:-

T1	T2	T3
R(A)		
	w(A)	
w(A)		
		w(A)

there exist a serial schedule for above example:-

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

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

(S1) Not CSS $\Longleftrightarrow$ T1 : T2 : T3 (S2)

S2 is equalent serial schedule for S1. But S1 is not conflict serializable schedule even if view seriable schedule is possible.

$\Rightarrow$

T1	T2	T3	T4
W(A)			
	W(A)		
		W(A)	
			W(A)

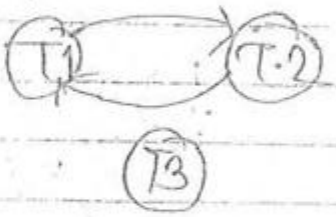
T1 : T2 : T3 : T4
(conflict SS)

There is 6 different serial schedules are possible for it, because only we have to save the final updation of data for transaction. We can change the order of initial three transaction in any order. Because they always not change the serial of any schedule

(T1 T2 T3 T4)

T1 T3 T2 T4
T2 T3 T1 T4
T2 T1 T3 T4
T3 T1 T2 T4
T3 T2 T1 T4

Que) S: $R_2(B)\ R_2(A)\ W_1(A)\ R_3(A)\ W_1(B)\ W_2(B)\ W_3(B)$
Identify all view equalent Serial Schedule.



i.e. Cycle is there so it is Non conflict serializable schedule

For checking View Serializable Schedule:-

[1] Final Updation:-

A: T1

B: T1 T2 T3 (T1 T2) → T3

⇒ Either T1 T2 T3 (due to Intial read condition) T2 T1 T3

[2] Initial Read:-

	Initial Read	Write
A:	T2	T1 ⇒ T2 → T1
B:	T2	T1, T2, T3 ⇒ T2 → T1
		T2 → T2
		T2 → T3

[3] WR Sequence:-

W1(A) : R3(A) ⇒ T1 → T3

Note:-

X! T1 T2 — T2 T3 T4

T1 T2 → T3 T4 ✗

Split it as many as possible

T1 → T2 T3 T4

T2 → T3 T4

By combining all the condition i-e.

(T1 T2) → T3

T2 → T1 T3

T1 → T3

} T2 → T1 → T3 are one by the view equalent serializable schedule.

Que) S: $S_2(A)\, S_1(A)\, W_1(C)\, S_3(C)\, W_1(B)\, S_4(B)\, W_3(A)$ $S_4(C)\, W_2(B)\, S_2(B)\, W_4(A)\, W_4(B)$

For checking View Serializable schedule:-

[1] Final Updation:-

A: T3 (T1) — T3 → T4

B: T1 T2 (T4) — (T1 T2) → T4

C: (T1)

[2] Initial Read:-

	Initial Read	Write	
A:	T1 T2	T3 T4	⇒
B:	—	T1 T2 T4	⇒
C:	—	T1	⇒

	Initial Read	Write
A	T1 T2	T3 T4
B	=	T1 T2 T4
C	—	T1

[3] WR Sequence-

1) $W_1(C)$ $R_3(C)$ $T1 \rightarrow T3$

2) $W_1(B)$ $R_4(B)$ ~~$T1 \rightarrow T4$~~

3) $W_1(C)$ $R_4(C)$ $T1 \rightarrow T4$

4) $W_2(B)$ $R_2(B)$ ~~$T2 \rightarrow T2$~~

$T1 \rightarrow (T3\,T4)$

[1] $(T1\,T2\,T3) \leftrightarrow T4$

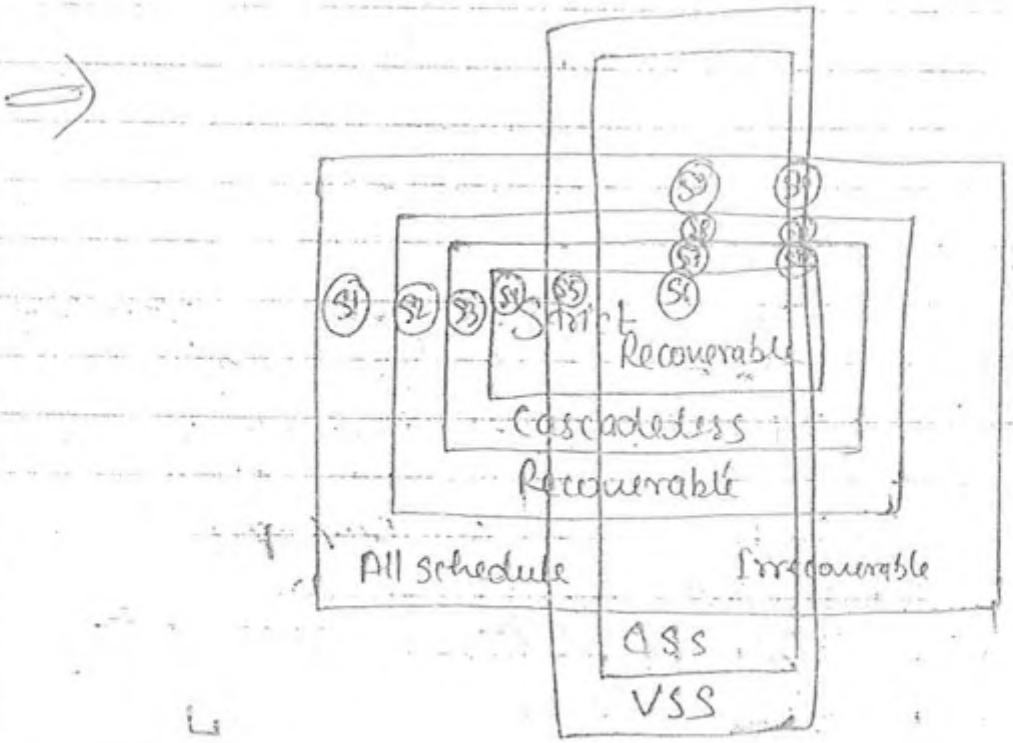
[2] $(T1\,T2) \rightarrow (T3\,T4)$

[3] $T1 \rightarrow (T3\,T4)$

$(T1\,T2) \rightarrow T3 \rightarrow T4$

$T2 \rightarrow T1 \rightarrow T3 \rightarrow T4$

Two view equivalent schedule are possible.



Schedule is Irrecoverable & not Serializable:-

T1	T2
R(A)	
W(A)	
	R(A)
	R(B)
R(B)	
W(B)	
	C2
C1	

(S1)

Schedule is Recoverable but not Cascadless

T1	T2
R(A)	
W(A)	
	R(A)
	R(B)
R(B)	
W(B)	
C1	
	C2

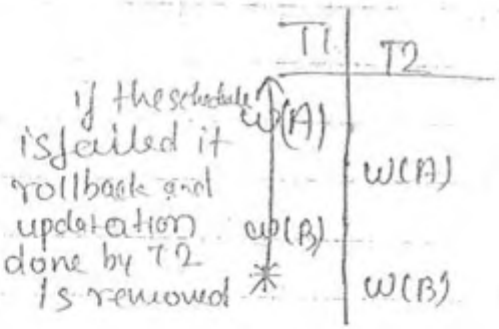
29/08/11

Concurrency Control Protocol:-

Ideal- This protocol is allowed to execute only 1) Serializable Schedule & NO RW, WR, WW Problem
Strict
2) Recoverable Schedule & May possible lost update problem

Difference Between WW problem & Lost Update Problem

WW Problem: result non serializable schedule
lost Update: Possible even in serializable schedule



(NO WW Problem
But lost Update Problem may possible.)

⇒ Strict Recoverable schedule (lost update problem not exist)

* Types of Protocol:-

a) locking Protocol.
b) 2 Phase locking Protocol.

[A] Locking Protocol:-

Lock:- Lock is a variable which is used to identify the status of the data item.

Legal Transaction:-

Transaction is said to be legal transaction if transaction should lock the data item before usage of data item & release data item after usage.

⇒ Every transaction in the schedule is legal transaction

Ex:-
```
L(A)
R(A)
W(A)
U(A)
```

* "lock" work as same like "critical section" in Operating system. That in Operating system critical section is used for "shared Resources".

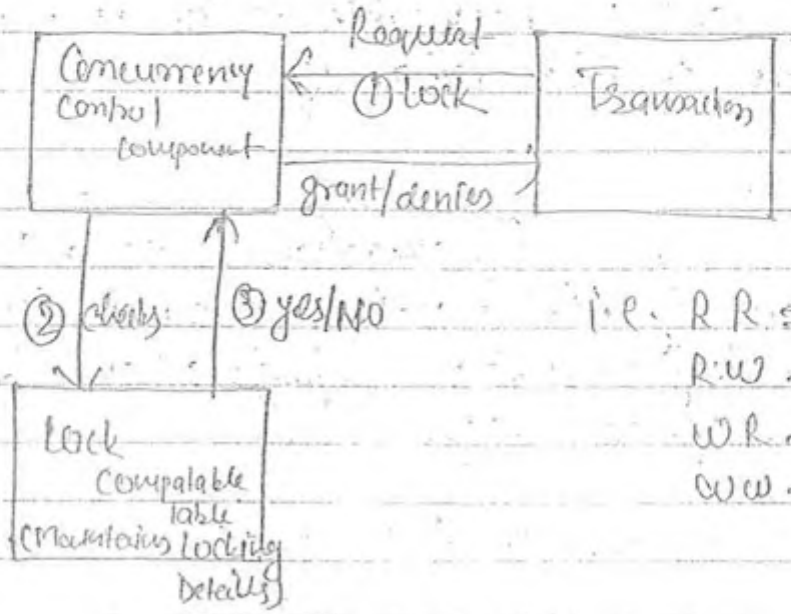
Critical Section - It is a shared resource for O.S:

For this we use Semaphore : Mutex that also shared resource.

```
Process
{
if (mutex=0)
  mutex = 1
  CS
  mutex = 0
}
```

```
if (mutex=0)
{
}
else
  {P2 leave to wait}
}
until mutex become 0( means
```

Components of legal transaction by locking Protocol :-



i.e. R R = allowed
R W - not allowed
W R - not allowed
W W - not allowed

Compatability is in "Yes" condition only if :-

a) More than one transition used only for "Reading" the data item

and

b) Data item is free from any transtion i.e. not used by any transaction.

Example :-

⇒ Shared / Exclusive locking :- [S/X]

Shared :- [S] :- Transaction locked the data item A in Shared mode means that transaction can Read data item A.

* Modification is not allowed in shared lock.

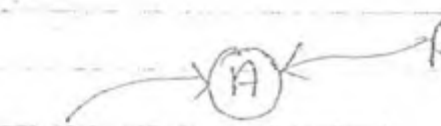
* Shared gives only read permission but exclusive lock gives read as well as update the data item.

Exclusive Lock (X) :-

Transition T locked the data item A in exclusive mode means that T can either Read/Write the Data item A.

T
X(A)
R(A)
W(A)

Lock Compatable Table :-



Requested Tj ↓ / Hold Ti →	S	X
S	YES	NO
X	NO	NO

Example :-

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

Transaction is legal and also it follow the compatable tablem.

⇒ Legal Trans & compatebulity satisfied b/w S/X locks.
but non-serializable schedule

⇒ (S/X may not free from Non Serializability)

i.e. locking protocol is not free from serializability so put some restriction on the locking protocol to remove the above problem. We can call such protocol as Two Phase locking Protocol

Two Phase Locking Protocol:-

Transition T not allowed to request a lock if it is already performed any unlock operation

Ex:- S(A)
X(B)
X(C)
S(D)
|
|
U(C)
S(E)
U(B)
U(A)
U(D)

(locking phase (growing phase))

(Lock Point:- Position of the last lock (or) position of the first unlock)

Not possible to again lock after we firstly unlock one of the data item according to 2PL.

(Unlocking Phase (Shrinking phase))

T1	T2
X(A)	
R(A)	
W(A)	
~~U(A)~~	← (denied)
	S(A)
	R(A)
	U(A)
	S(B)
	R(B)
	U(B)
X(B)	
R(B)	
W(B)	
U(B)	

denied

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

Not possible to execute using 2PL.

⇒ 2PL: Always ensure conflict Serializability. But reverse is not true. (Not possible To execute any non-serializ schedule).

⇒ If it is 2PL then it is serializable. But reverse it not always true i.e. every serializable is execute 2PL.

⇒ Equalent Serial Schedule based on the position o lock point.

Que) Check that schedule is allowed to executed by 2PL or not

T1	T2
X(A)	
W(A)	
lock point → R(B) U(B)	S(A)
	R(A)
W(B)	
U(B)	X(B) ← lock point
	W(B)
	U(B)
	U(A)

Allowed To Execute by 2PL so it is conflict serializable.

T1 : T2

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

⇒ Not possible using 2PL

T1	T2	T3
	X(A)	
	R(A)	
X(B)		
R(B)		
	W(A)	X(A) → Not possible i.e. Exclusive locks not allowed
X U(B)	X X(B)	R(A)
	X U(A)	
W(B)		
		W(A)
	R(B)	
	W(B)	

T1 → T2 → T3

Conflict Serializable Schedule but not possible using 2PL.

⇒ Every conflict serializable schedule may not possible to execute by using 2PL.

Lock Upgrading Technique in 2PL :- If there is "read" operation initially only we give shared lock only till we get the write operation on A. If write operation come on same data item then we upgrade the lock to exclusive lock.

T1
S(A)
R(A)
⋮
X(A) : request exclusive without
W(A) unlocking of the previous shared lock.

⇒ Increase the concurrency level.

(Que) Check this schedule is allowed or not allowed to execute 2PL using lock upgrading technique :-

T1	T2	T3
S(A)		
R(A)		
U(A)	X(A)	
	W(A)	
W(A)		
		U(A)

Not possible even by using 2PL with lock upgrading.

T1	T2	T3
	S(A)	
	R(A)	
S(B)		
R(B)		
	X(A)	
X U(B)	W(A)	
	U(A)	S(A)
		R(A)
W(B)		
		W(A)
	R(B)	
	W(B)	

Not possible even by using 2PL with lock upgrading

* ⇒ All conflict serializable schedule may not possible to execute even by using lock upgrading of 2PL.

Que) 2PL Allowed to execute?

T1	T2	T3
S(A) R(A)		R(C)
S(B) R(B) U(A) U(B)	X(B) R(B) W(B)	
		X(A) W(A)
S(C) R(C)		U(C)
		W(C)

⇒ Not allowed in 2PL.

Lock upgrading technique using 2PL allowed or Not

T1	T2	T3
(S(A) R(A)		R(C)
(Lock point) ⇒ S(B) R(B) U(A) U(B)	S(B) R(B) X(B) W(B)	
		X(B) W(A)
(lock point)	⇒ S(C) R(C) U(C) U(B)	⇒ X(C) U(C) U(A)

⇒ allowed to execute using lock updating technique.

Equivalent Serial schedule

T1 → T2 → T3

2PL work with S/X locking!-

Limitation of 2PL!-

[1] Basic 2PL!-

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

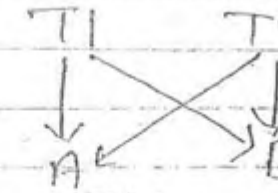
⇒ Serializable schedule is allowed to execute by 2PL.

but IRRECOVERABLE Schedule.

*

[2]

T1	T2
X(A)	
W(A)	
(Denied)	X(B)
	W(B)
X(B)	X(A) (denied)
W(B) waiting	W(A) waiting.



i.e. T1 transaction need T2 for execution & T need T1 for execut then T1 & T2 gets deadlock condition.

* 2PL may results the deadlock also or may not free from Deadlock.

3) Starvation:- If any process or transaction gets infinite block for a data item then starvation is occur. One transaction permanently block due to unavailability of resources but other transaction is executed this scenario is called starvation.

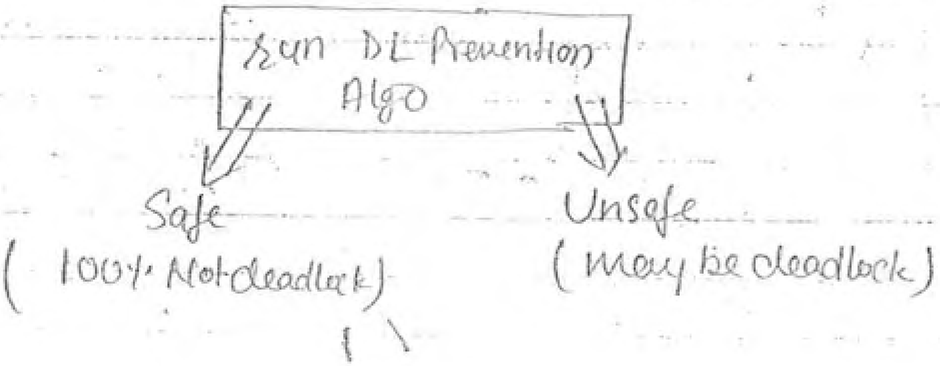
Example:-

	T1	T2	T3	T4
		S(A)		
denied (due to T2)	X(A)			
			S(A)	
		U(A)		
denied (due to T3)	X(A)			S(A)
			U(A)	
denied (due to T4)	X(A)			

* May not free from starvation.

Deadlock Prevention:- not allow to occur the deadlock. (OS) means not to get the deadlock

Process P1: requested R1



T1	T2
W(A)	
	R(A)
	C
R/C	

(Irrecoverable)

T1	T2
W(A)	
	R(A)
	Abort
C	

(Recoverable)

What modification is need in 2PL to become a Recoverable is called Strict 2PL protocol!-

[2] Strict 2PL Protocol!-

It is a Basic 2PL mean it also contain locking phase & unlocking phase + all Excutsive locks should be ho until commit Rollback

T

X(A)

S(B)

X(C)

S(D)

U(D) → we may unlock the shared lock before the commit or rollback

U(B)

Commit / Rollback

U(A) → But we always unlock the exclusive lock after the commit or rollback of transaction.

U(C)

Example:-

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

Advantage of Strict 2PL:-

* Always ensure strict Recoverable condition because of holding Exclusive locks until commit.
* Because of 2PL ensuring the conflict serializability schedule Equalent Serial based on lock point.
* This schedule is enough or sufficient to solve all the problem like strict Recoverable & conflict SS

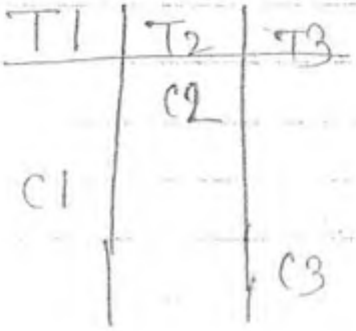
[3] Rigorious 2PL Protocol:-

Basic 2PL + All Exclusive locks / shared lock should be hold until commit.

T1
S(A)
X(B)
commit
U(B)
U(A)

* Difference between Strict 2PL & Regorious 2PL is that instrict 2PL only all exclusive lock should be hold until commit but in Regosious all exclusive lock as well as all strict lock also should be holol after commit.

⇒ Ensuring strict Recoverability
⇒ Ensuring serializability
⇒ Only difference is that Equalent Serial schedule based on order of commit.



⇒ T2 ! T1 ! T3

Conservative 2PL Protocol :-

⇒ This protocol is basically made for To Avoid Deadlock.
⇒ It says that lock the all dataitems that are required to execute by the transaction, before beginning of the execution of Transaction. If any data item is not free release the existing lock and repeat the same after time out

S(A)	S(B)
X(C)	X(D)

→ R(A)
R(B)
W(C)
W(D)

1) Some data item is not free :-

S(A)
S(B)
X(C)
X(D) ← denied (i.e. not free)

Release all the remaining lock → U(C) U(B) U(A)

wait for timeout

Again request for same data item → S(A)

Disadvantages of Conservative 2PL :-

a) : Starvation Possible

b) Irrecoverability is possible.

Example :-

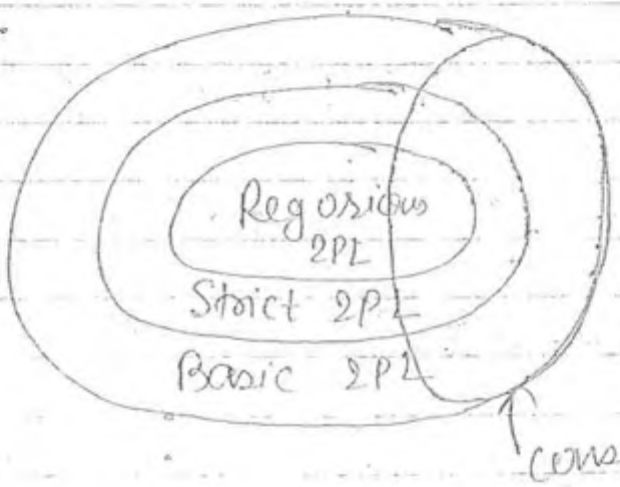
T1	T2
X(A)	
X(B)	
W(A)	
U(A)	
	S(A)
	R(A)
	U(A)
W(B)	
W(B)	
	Commit
Commit	

Here it is irrecoverable but also conservative 2PL protol is possible.

* If we only put the restriction of unlock then only strict recoverability is possible.

[c] [The resources are holded long before the requirement.] becoz of this is less concurrency is occour, so throughput is less.

[d] Prediction of locks required by the transaction is too difficult.



Q(iii)

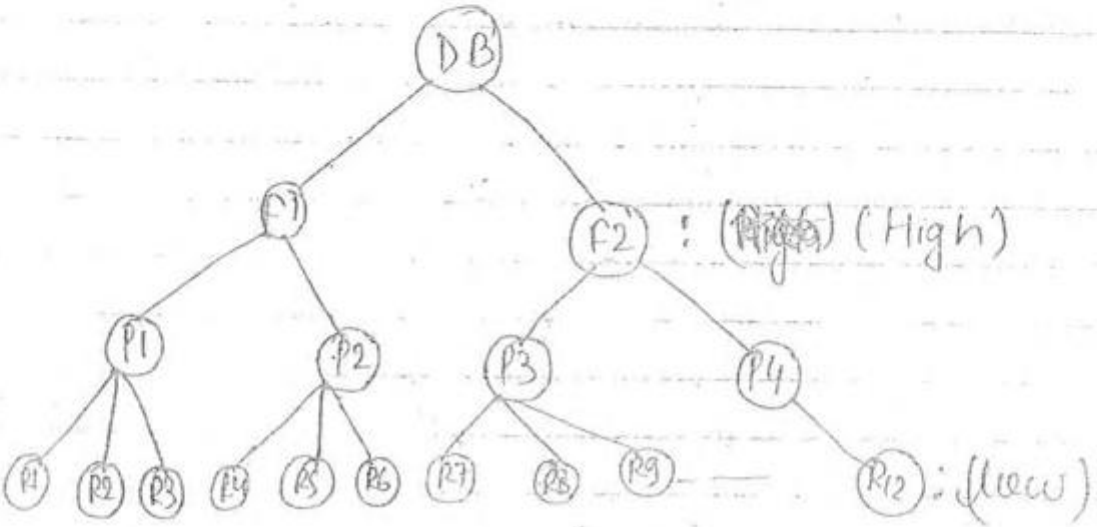
T1	T2	T3
S(A) R(A)		
	S(C) R(C)	
X(B) U(A)		X(A) W(A)
W(B) U(B)	S(A) R(A)	U(A) (cleared in strict 2PL)
	U(A) U(C)	Commit
Commit	Commit	

a) Not in 2PL
b) 2PL but not strict 2PL
c) Strict 2PL but not Regorious
d) Regorious 2PL

Multilevel Granularity Protocol :- (Tree Protocol :-)

Granularity :-

Granularity means lock in the DB.



There are two types of lock on the basic of Granularity

1] High level locking
2] low level locking

Example :- If T1: Update R1, R2, R3, R4, R5, R6
T2: update R2 & R12

[1] By using High level locking :- * $T_1 : (L_1(F_1))$

Advantage: No of locks are very less
(lock compatable Table
easy to maintain)

* T2: $L_1(F_1)$ $L_2(F_2)$

Disadvantage :- results less concurrency
because Transition may
hold Un-necessary resource

[2] Low level locking :-

Disadvantage :-

* T1 : $l_1(R_1)\, l_1(R_2)\, l_1(R_3) - - - \{l_1(R_6)\}$ } Complex to maintain lock compatability

Advantage

* T2 : $l_2(R_2)\, l_2(R_{12})$ } (More concurrency level)

[3] Multilevel granularity :- (Best possible lock depends on transaction requirement)

Depends on transaction Acme should allow to lock at any level.

* T1 : $l_1(F_1)$

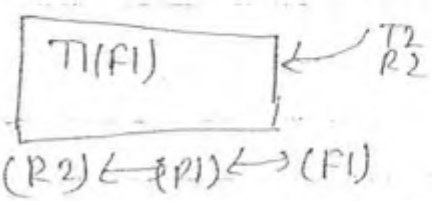
* T2 : $l_2(F_2)\, l_2(R_{12})$

* Firstly file is locked then after request for locking record

Example :-

T1	T2
$l_1(F_1)$	
	$l_2(R_2)$ ← denied

Then how to check which file is locked then how the concurrency control do for grant ordering the permission



* No. of comparison = height of Tree = constant. (To grant or to deny the permission)

"record"

⇒ If first type record is locked & we request for locking the file in which lock record is belongs!-

T1	T2
	$l_2(R_2)$
$l_1(F_1)$ ↗ denied	

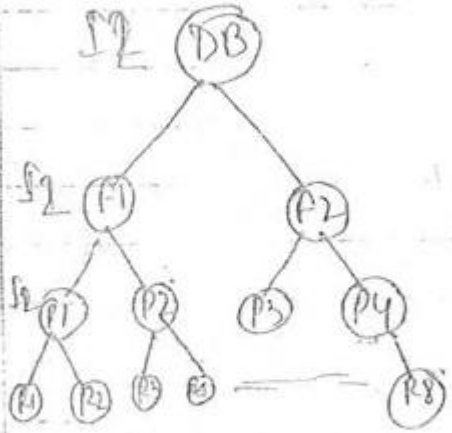
Difficulty comes for concurrency control to allow or grant the permission or deny the permission!-

$l_2(R_2)$

checked for locking of $F_1, P_1, P_2, R_1, \ldots R_6$)

$O(2^n)$ i.e. exponential time.

To solve the above difficulty!-
We have to maintain Intention lock at every level i.e.



$I_2(DB)\ I_2(F_1)\ I_2(P_1)\ l_2(R_2$

30/08/11

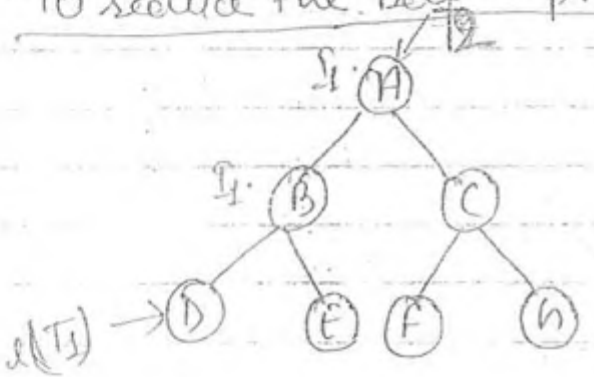
Tree Protocol:-

a) Multilevel Granularity Protocol:-

Intension lock:-

A Node N can be locked by Trans T in Intension mode means that trans T can allowed to request lock on Any descendent of N

To reduce the below problem we use Intension lock



T1	T2
$I_1(D)$	$d_2(A)$ ← (exponential time) because lock status required to check Any descendents of A.

Intension lock

T1
I(B)
I(D)
R(D)
I(E)
W(E)

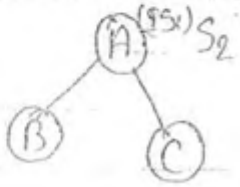
Direct lock

T1
I(B)
R(D)
W(E)

These are following Intension locks:-

[1] Intension Shared lock [IS]:- A Node N locked by Transaction T in intension shared (IS) mode means that any descendent of N can be request shared lock (S) by transaction T.

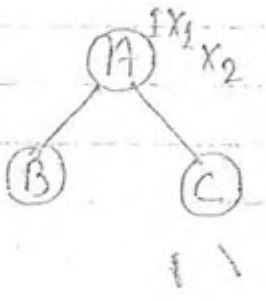
⇒ If parent is Intension shared lock it only allow the the read the data item.



T1	T2
IS(A)	S(A) (only shared lock)
S(B)	R(B)
R(B)	R(C)
S(C)	~~W(C)~~
R(C)	
~~X~~	

[2] Intension Exclusive lock [IX]:-

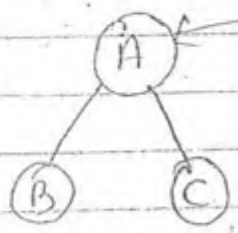
A node N locked by Transaction T in intension Exclusive mode means that Any descendent of N can be request shared / Exclusive lock by transaction T.



T1	T2
IX(A)	X(A) (only Exclusive lock)
S(B)	R(B)
R(B)	W(C)
X(C)	
W(C)	

[3] Shared Intension Exclusive lock [SIX] :-

→ Its Combination of Intension Exclusive lock & Shared direct lock.



T1

SIX(A)
R(B)
R(C)
X(C)
W(C)

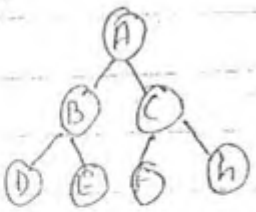
Direct shared lock
Intension Exclusive lock

Combination of Intension exclusive & Direct shared lo

Multilevel Granularity Protocol Conditions :-

[1] A node N can be locked by Transaction T only parent of N is already locked by Transaction T.

If we want to lock detailed item "F". Then

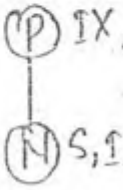


T1

IS(A)
IS(C)
S(F)

[2] A node N can be locked by Trans T in S, IS only if parent of N is already locked by Tra T in either 'IS' or 'IX' mode.

If we want to lock N in shared mode :-



T1

IX(P)
S(N)

or

T1

IS(P)
S(N)

or

T1

IX(P)
IS(N)

or

Que) Worst case of Heap sort
a) $\log n$ b) n c) $n \log n$ d) n^2

Que) which of the rays are not emitted by any particle.
a) α b) β c) γ d) None of these

Que) which of the layer is responsible for routing
a) Physical layer b) Data link layer c) Network layer d) Transport layer

Que) There are 16 element in the array. then what will the value from after execution of this program

a = 4 4 5 5 6 6 6 7 7 7 8 8

```
count = 0
for (i = 1 to a[end])
    if (a[i] = a[i+1]) then
        count = count + 1
    else
        i++;
```

a) 3 b) 5 c) 6 d) 7

Que)
```
float a = 35.75  b = 71.50
b = a
if (a = b)
    b = 2 a
b = b/2
```
then what will the value of b

Que) which join is used to select only distinct column
a) Natural join b) Outer join c) equi join d) none of them

Que) which one of them is so important in Digital signature
a) Hash fun b) Encryption c) data hiding d) Decryption

Que) int *p, *q then which of them is not current operation
a) p+q b) p-q c) p*q d) p/q

Que) In the given sorted array if two element are same then which type of sorting keep the element in same order after sorting them.
a) Stable b) strict

Que) what is the full form of URL

Que) In complete binary tree what the difference between two sub tree of any node are possible
a) 1 b) 2 c) 0 d) 3

Que) If xxx yyyy and zzz are element arranged in order then in preorder which element is taken first
a) xxx b) yyy c) zzz d) None of these

Que) Calculate the value of -ab+-cdb if a=2, b=4 c=3 d=5

Que) what is the full form of SSL in security purpose in SSL.

Que) If any table is in BCNF and if there is no any multivalued dependency then which of them is correct
a) 1st Normal form b) 2NF 3) 4NF 4) None of these

Que) In a ER model which of them is not so important to taken
a) Entities b) Attribute c) Primary key d) Relationship

Que) No. of minimum interchange to required to keep the maximum no. of element at the root in the tree.
89, 17, 12, 15, 2, 3, 4, 79,

Que) if (x>y>z?200:300) if x=4 y=4 z=6
a) 200 b) 300 c) 100 d) None of these

Que) S = $a^n b^n$ / $n \geq 1$ which of them is not belongs to this language
a) aabb b) abab c) aaabbb d) aabab

T1	T2	T1	T2
S(X(B))		S X(A)	
X(B)	S(X(A))	X(B)	S X(A)
ω(B)	X(B)	ω(B)	X(B)
	ω(A)		ω(A)

1484101023221
Krishna Kumar Pa
Canara Bank
Jitsingh Marg New
110067

red
green
S4
S3

S1
S2
S3

S4	S1
S1	S3
S1	S4
S1	S1
S2	S3
S2	S4
S2	S1
S3	S3B
S3	S4
S3	

AB
A1 B1
A2 B2
A3 B4
A3 B3
A3 A2

3
3%

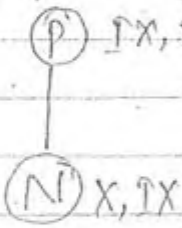
629, 680, 507,

7150

AB CD A D

a=b

[3] A Node N can be locked by Transaction T in "X, IX, SIX" mode only if parent of N is a locked by Transaction T in IX, SIX mode.



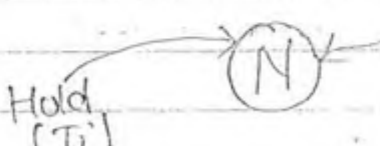
[4] Transaction T can be request for any lock on Node only if Transaction T not performed any unlock operation (2PL)
(This condition is required to ensure seria buli

[5] A Node N can be unlock by Transaction T only if None of the descendents of N is already locked by T. (i.e. unlock is done from bottom Top.)

[6] Compatable Table:-

Compatability:-

If Node N is locked by Transa (Ti) & another transaction (Tj) request fo locking the same Node then compatability that whether to grant the permission or deny the request of transaction (Tj).

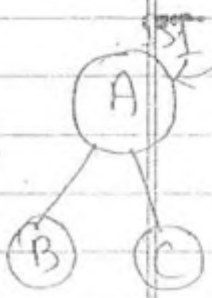


where i ≠ j

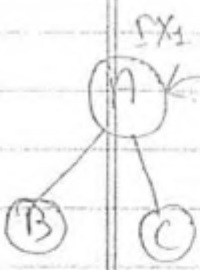
→ Hold T_i

T_i \ T_j	IS	IX	S	SIX	X
IS	YES	YES	YES	YES	No
IX	YES	YES	No	No	No
S	YES	No	YES	NO	NO
SIX	YES	NO	NO	NO	No
X	No	No	NO	NO	NO

Request T_j ↓

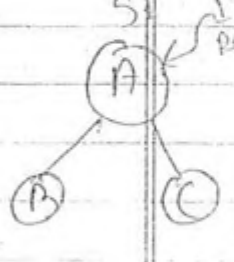


T1	T1
IS(A)	
	IS(A)
S(B)	
R(B)	
	S(B)
	R(B)

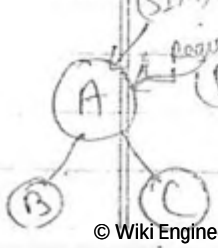


T1	T2
IX(A)	
	IS(A)
X(B)	
W(B)	
No inconsistency	S(B) ← denied
	S(C)
	R(B)

If we allow concurrency increases



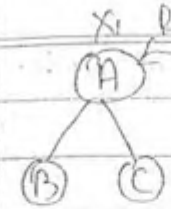
T1	T2
S(A)	
	IS(A)
R(B)	
	S(B)
	R(B)



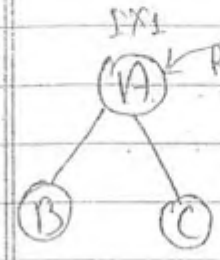
SIX ↔ IS

S — IS ⇒ Yes

IX — IS ⇒ Yes



T1	T2
X(A)	
	IS(A) ← if we grant
	S(B)
	R(B) → if we grant then RW problem may possible
W(B)	



T1	T2
S(A)	
	IX(A) ← grant
	X(B)
	W(B)
R(B)	

SIX — IX < S – IX (N)
IX – IX (Y)

SIX — IS < S – IS (Y)
IX – IS (Y)

* Intension — Intension lock

IX	—	IX	Compatable
IX	—	IS	
IS	—	IX	
IS	—	IS	

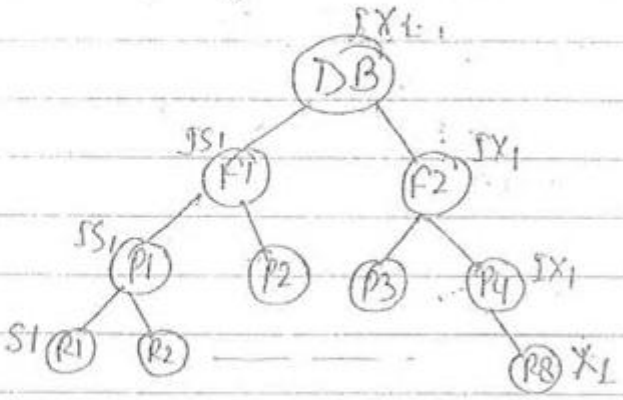
Simultaneous Read Compatable :-

S	—	IS
S	—	S
IS	—	S

Compatable

* If the transaction satisfy all the above six condition then only it is called multilevel granular

(Qu) T1: Read R1 & update R8 then what will be the locking pattern.



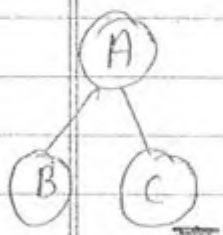
T1

IX1(DB)
IS1(F1)
IS1(P1)
S1(R1)
Read(R1)
IX(F2)
IX(P4)
X(R8)
write(R8)
U(R8)
U(P4)
U(F2)
U(R1)
U(P1)
U(F1)
U(DB)

=> Multilevel Granularity Protocol ensure serializability.
(Equivalent Serial schedule based on lock points)
(because it is in 2PL)

=> Irrecoverability Possible becoz there is no any restriction for commit of any transaction

Example:- check multilevel granularity is possible or not



T1	T2
IX(A)	
X(B)	
W(B)	
U(B)	
U(A)	
	IS(A)
	S(B)
	R(B)
	U(B)
	U(A)
	Commit
Commit	

i.e. multilevel granularity is possible but it Irrecoverable schedule

=> Deadlock & Starvation possible.

* To remove the Irrecoverability from Multilevel granularity protocol we put some restriction on it then it become called as strict Multilevel granularity protocol.

Strict Multilevel Granularity protocol:-

Multilevel Granularity protocol + all Exclusive lock should be hold until commit/rollback

=> Ensure Serializability & Strict Recoverability.

=> Even deadlock & starvation is possible.

Timestamp Ordering Protocol:-

One difference between locking & Timestamp protocol is that timestamp is a protocol which is always free from Deadlock.
⇒ (Free from Deadlock)

Time Stamp Value:-

Time stamp is a unique value assigned by DBMS to every transaction in ascending order.

Advantages of Timestamp value:-

1) Because it is unique value, TS is used for identification.

2) Because it is in ascending order, TS is used for to set priority.
(Transation with less TS value gets high priority).

Example:- 10 11 12 13
T1 T2 T3 T4
older → younger

* Every data in the dB maintain two TS values

1) Read TS(A):-

Highest Transaction TS value that has performed R(A) operation successfully.

10	20	30	40
T1	T2	T3	T4
R(A)			
	R(A)		
		R(A)	

RTS(A) = ~~10~~ 30

2) Write TS(A):-

Highest Transaction TS value that has performed W(A) operation successfully.

10	30	20
T1	T2	T3

⇒ Allowed to execute concurrently. Concurrent execution should be equal to Serial, based on Time stap ordering

(T1 : T3 ! T2)

* Concurrency of the Timestamp protocol is less than locking protocol because in timestamp protocol concurrent execution should be equal to only one serial schedule based on timestamp ordering but in locking protocol concurrent execution should be equal to any serial schedule which satisfy 2PL condition. So, Timestamp protocol is less concurrent.

Example:-

10	30	20
T1	T2	T3
	R(A)	
R(B)		
		W(B)
	W(B)	rollback
		W(A) → violets the serial i.e. T1→T3→T2 so T3 should be rollback

If given serial is $T_1 \rightarrow T_3 \rightarrow T_2$

$\Rightarrow$ T1 → T3
T1 → T2
T3 → T2

* T2 → T1 is possible but T1 → T2 is not allowed i.e. cycle dependency is not possible so it is deadlock free protocol.

Possibility of Rollback :-

1)

10 T1	20 T2	
	W(A)	i.e. T1 → T2 "If younger Transaction updated dataitem. Then olders not allowed to Read" bcoz conflict pair from T2 → T1.
R(A) (Rollbacking T1)		

Transaction T_i issues R(A) operation :-

$$WTS > TS(T_i)$$

Rollback T_i

2)

10 T1	20 T2	
	R(A)	i.e. T1 → T2 "If younger transaction read data item A. Then olders transaction not allowed update A" bcoz conflict pair orders is from T2 → T1.
W(A) (Rollback T1)		

Transaction T1 issues W(A) operation :-

$$RTS > TS(T1)$$

Rollback T1

3)

10 T1	20 T2	
	W(A)	i.e. T1 → T2 "If younger transaction updated data item. Then olders transaction not allowed to update" bcoz conflict pair order is from T2 → T1.
W(A) Rollback T1		

* When conflict pair is based on time stamp order then there no need to rollback. But when conflict pair is not based on time stamp order then we have to rollback the transaction.

Transition T1 issues W(A) operation :-

WTS(A) > TS(T1)
Rollback T1

Basic Time Stamp Ordering Protocol :-

a) Transaction T issues R(A) operation :-

[I] if WTS(A) > TS(T) Then Rollback T

[II] Otherwise allowed to execute R(A) operation by Transtion T
Then Set RTS(A) = max {RTS(A), TS(T)}

Ex :-

10 T1	20 T2	30 T3
R(A)		
		R(A)
	R(A)	

RTS(A) = ~~10~~ 30

Here is no any conflict pair then we allow this trans

b) Transaction T issues W(A) operation :-

[I] if RTS(A) > TS(T) Rollback T
[II] if WTS(A) > TS(T) Rollback T
[III] Otherwise allowed to execute W(A) operation by Transaction T
then
Set WTS(A) = TS(T)

Ex :-

10 T1	20 T2	30 T3
W(A)		
	Rollback W(A)	
		W(A)

WTS(A) = ~~10~~ 30

Classmate
Date
Page

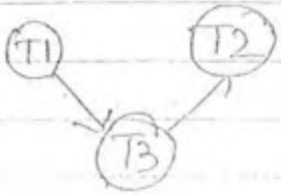
* By default we take transaction no. as timestamp value if the timestamp value is not given.

(Que) which of the time stamp values allow to execute the schedules

S: $R_1(A) R_2(B) W_1(C) R_3(B) R_3(C) W_2(B) W_3(A)$

a) {10, 20, 30}
b) {30, 10, 20}
c) {10, 30, 20}
d) {30, 20, 10}

$R_1(A) : W_3(A)$ 1 → 3
$R_3(B) : W_2(B)$ 3 → 2



→ T1 : T3 ! T2

(Que) If timestamp {10, 20, 30} are given then which transition is roll back.

{10, 20, 30}
T1 → T2 → T3
T1 → T2
T2 → T3
T1 → T3

then check in the transition one by one by take operation from above schedule and check by taking previous operation, that where conflict pair is and is the conflict pair satisfy above precedence of the given question.

S: $R_1(A) R_2(B) W_1(C) R_3(B) R_3(C) W_2(B) W_3(A)$

T3 → T2 ≠ T2 → T3

So, T2 transition rollback T2

once you discard the one transaction then discard all the operation related to that transaction for further checking that whose other transaction n be roll backed.

Ex:- $R_1(A)\ R_2(B)\ W_1(C)\ R_3(B)\ R_3(C)\ W_2(B)\ W_3(A)\ W_2$ (W2 and W2(B) crossed out)

T1, T2, T3
(30, 10, 20)

[b]

T2 → T3 → T1

i.e. T2 → T3
T3 → T1
T2 → T1

S! $R_1(A)\ R_2(B)\ W_1(C)\ R_3(B)\ R_3(C)\ W_2(B)\ W_3(A)$ (W1(C), R3(C), W3(A) crossed out)

$W_1(C)$ is conflict with $R_3(C)$ ⇒ 1 → 3 not allo

So, we discard all operation of T
⇒ Rollback the Transition T_3.

T1, T2, T3
{30, 20, 10}

[d]

T3 → T2 → T1
T3 → T2
T2 → T1
T3 → T1

S! $R_1(A)\ R_2(B)\ W_1(C)\ R_3(B)\ R_3(C)\ W_2(B)\ W_3(A)$

$R_1(A) \rightarrow W_3(A)$ ⇒ 1 → 3 (not possible)

So, T_3 is rollbacked.

THOMAS WRITE TIMESTAMP ORDERING PROTOCOL:

Possibilities of Rollback!-

[1]

10	20
T1	T2
R(A)	W(A)
R(A)	

(Rollback T1)

[2]

10	20
T1	T2
	R(A)
W(A)	

(Rollback T1)

[3]

10	20
T1	T2
	W(A)
[W(A)]	

Ignore Write(A) but continue the Execution of T1

"If younger transaction updated data item Then older transaction updations are ignored"

Thomas write timestamp ordering protocol :-

(a) Transaction T issues R(A) operation :-

i] if WTS(A) > TS(T) then Rollback T

ii] Otherwise allowed to execute R(A) operation by Transaction T
Set RTS(A) = max{RTS(A), TS(T)}

(b) Transaction T issues W(A) operation :-

[i] if RTS(A) > TS(T) Rollback T

[ii] if WTS(A) > TS(T) (ignore W(A) continue execution of T)

[iii] Otherwise allowed to execute W(A) operation by Transaction T.
Set WTS(A) = TS(T)

(Qu)

10 T1	20 T2	30 T3
R(A)		
	W(A)	
W(A)		
Rollback		W(A)

Basic TO protocol :- (close to conflict Serializable
Rollback T_1 because of $W_1(A)$

TWR TO protocol :- (close to view serializable sch
No Rollback
($W_1(A)$ ignored)

10 T1	20 T2	30 T3
R(A)		
	W(A)	
W(A)		
Ignored		W(A)

$\Rightarrow T1 \to T2 \to T3$ is equal to view serializable

RULE of Thomas Write Time Stamp Ordering protocol & Basic TS ordering :-

[1] Serializability
(Equivalent Serial based on order of TS values)

[2] Free from Deadlocks

[3] Not free from Starvation.

10	20	30	40
T1	T2	T1	T3
	W(A)		W(A)
R(A)		R(A)	
Rollback		Rollback	

[4] May not free from irrecoverability.

10	20
T1	T2
W(A)	
	R(A)
	Commit
Commit	

BTO + TWRTO allowed to execute the schedule. But this schedule is irrecoverable.

Strict Recoverability :-

10	T1	T2	20

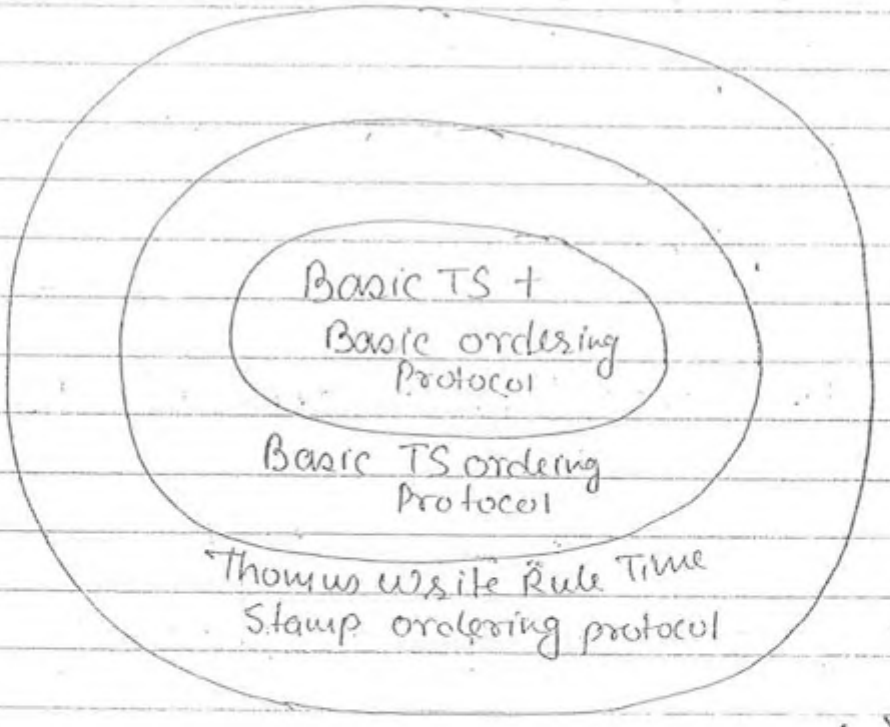
10 20

T1	T2
W(A)	
C/R	
	R(A)/W(A)

i.e. $WTS(A) < TS(T2)$

Strict Timestamp Ordering Protocol:-

Transaction T_2 issues R(A)/W(A) operations if $WTS(A) < TS(T_2)$. Then transaction (T2) which issuing R(A)/W(A) operation should be delayed until commit/Rollback of T_1 that is already performed write operation.



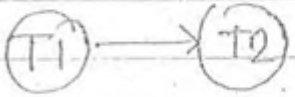
31/08/2011

Deadlock Prevention Algorithm :-

The basic purpose of deadlock prevention algorithm is to prevent the deadlocks in lock based protocol using Time Stamp Values.

Precedence Graph :-

T1	T2
X(A)	
	S(A)



Dependency Graph :-

T1	T2
X(A)	
	S(A)

⇒ i.e. dependency graph is opposite of Precedence graph.

These are two types of protocol :-

[1] WAIT - DIE Protocol :-

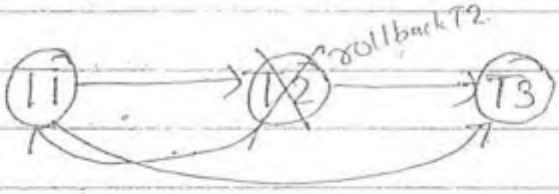
a) Assume $TS(T_i) < TS(T_j)$

[b] [1] If Transaction T_i required resources

that is holded by Tj, Then Ti allowed to wait until Tj unlocks.

[ii] if Transaction Tj required resource that is hoi by Ti, then Rollback Tj. (to remove starvation) Restart with same Timestamp value.

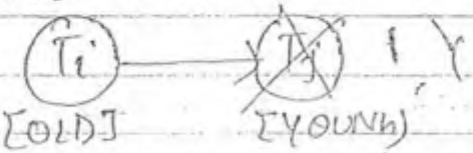
⇒ Deadlocks are not possible there becoz we are not allowed to form cyclic dependency.



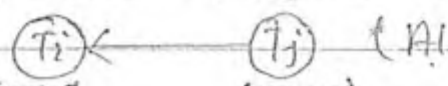
[2] WOUND-WAIT Protocol:-

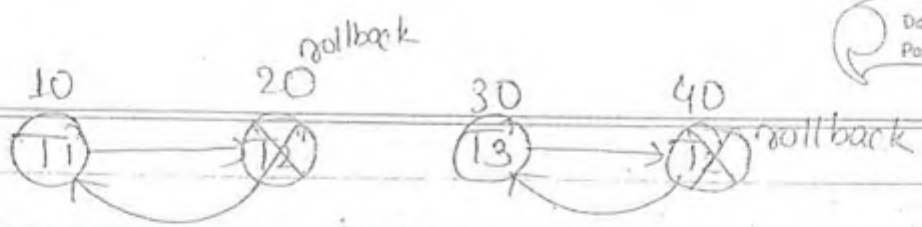
a) Assume TS(Ti) < TS(Tj)

b) [i] If Transaction Ti required resource that is holded by Tj, then Rollback Tj (to remove starvation) Restart with same Timestamp value.

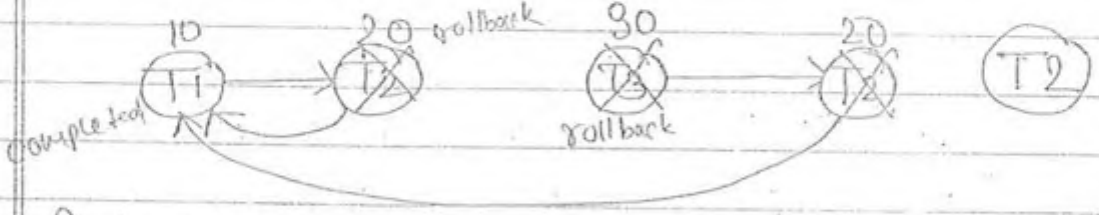


[ii] If Transaction Tj required resource that is holded Ti, then Tj allowed to wait until Ti unlocks.





i.e. if we take the transaction with new time-stamp value then starvation is possible (i.e. we go for infinite rollback)



But if we take the rollback transaction T2 with same time stamp value again then starvation is removed from that schedule. Even if again transaction required resource T1 then starvation is possible but only for finite time i.e. till the transaction T1 does not completed. So, T2 is not rollback for infite time. So, Starvation is removed.

→ Concurrency problems → RW, WR, WW

→ Classification of schedule → Recoverabutity, Serializabulity ***

→ Protocols :— ***
- locking
- Timestamp

Indexing & File Organization :-

Database files Divided into Blocks or pages.
Page divided into records.

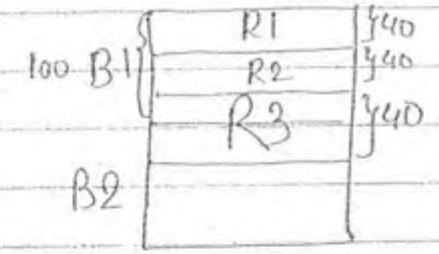
i.e. Database file collection of Blocks.
Each Block collection of Records.

Records are stored in a Block in Two ways :-

[1] Spanned Organization :-
[2] Unspanned Organization.

[1] Spanned Organization :-

Record can be spanned b/t the blocks.



Block Size = 100 byt
Record size = 40 byt

Block factor :-
No. of records per block.

= Block Size / Record Size

= 100 / 40 = 2.5

Advantage of Spanned organization :-

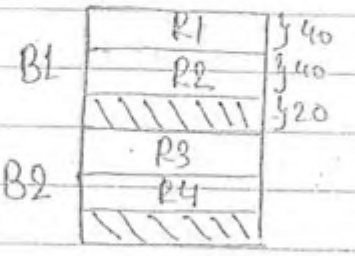
⇒ No memory leaks (OS) No internal fragmenta

Disadvantage :-

⇒ More I/O cost.

[2] Unspanned Organization:-

Entire record should be accomodate in Single block.



Block Size = 100 bytes.
Record size = 40 bytes

$$\text{Block factor} = \left\lfloor \frac{\text{Block Size}}{\text{Record Size}} \right\rfloor$$

$$= \frac{100}{40} = 2$$

Disadvantage:-

⇒ Internal fragmentation Possible.

Advantage:-

⇒ less I/O cost.
because Secondary memory → Main memory (block by block)

⇒ I/O cost:-

No. of blocks required to transfer from Secondary Memory to Main memory to access any record.

Search key:-

Attribute set used to access database record.

Select *
From Emp
where Empno = X
(Empno: search key)

Here we use Empno. to select the data or search the data.

Select *
From Emp
where ppno = y
(ppno: search key)

Search key is a key use to compare the data

Emp DB File

Eno	Ename	rating	PPno	
5			9	B1
10			6	
15			3	
21			1	B2
23			4	
35			8	
40			5	B3
45			7	
55			2	
				BN

ORDERED FILE:-

DB records physically ordered based on Search key attribute.

Select *
From Emp
where Eno = X
Search key
(Ordered File)

Select *
From Emp
where ppno = Y
Search key
(UN-ORDERED FILE)

Worst Case I/O cost:-

If the file is ordered file: I/O cost :- $\lceil \log_2 n \rceil$

If the file is un-ordered: N: blocks

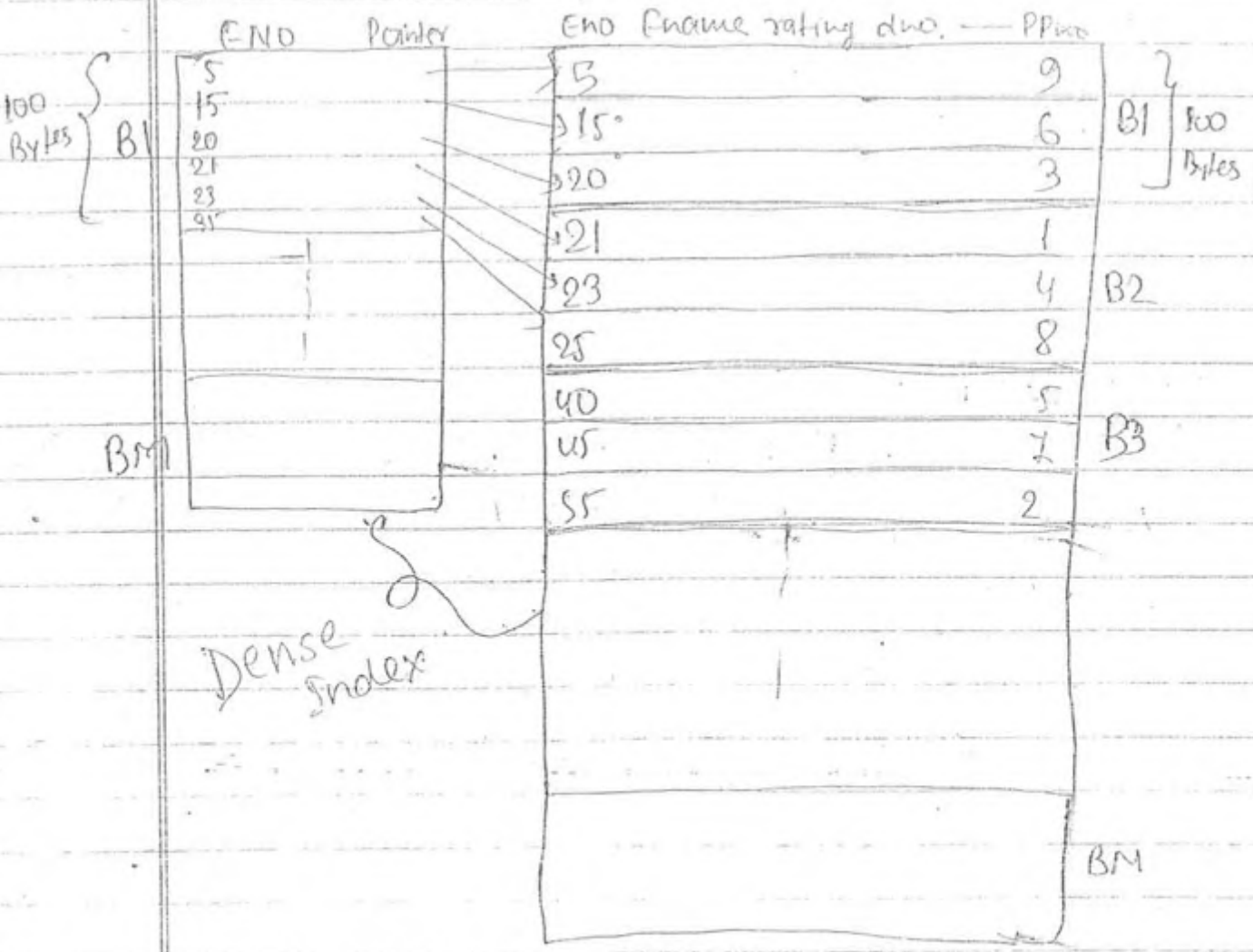
To reduce the I/O cost from $\lceil \log_2 n \rceil$ to N we go for Indexing:

INDEXING:-

Idea:- [To reduce I/O cost]

Size of DB Block = Size of Index block

Index file (Emp DB file)



⇒ Entry of the Index file :- Two fields
< Search key , Pointer >
Ex :- < ENO , Pointer >

⇒ Entry size of Index file << Entry size of DB Re

⇒ No. of entries per block = Block factor
as per above diagram block fack for
Index file = 6
DB file = 3

⇒ "Block Factor" of Index file >> Block factor of DB file

⇒ Total No. of Index file Block << Total No. of DB File Blocks.

(M << N)

⇒ I/O cost with Index = $\lceil \log_2 M \rceil + 1 \leq \lceil \log_2 N$
or
N

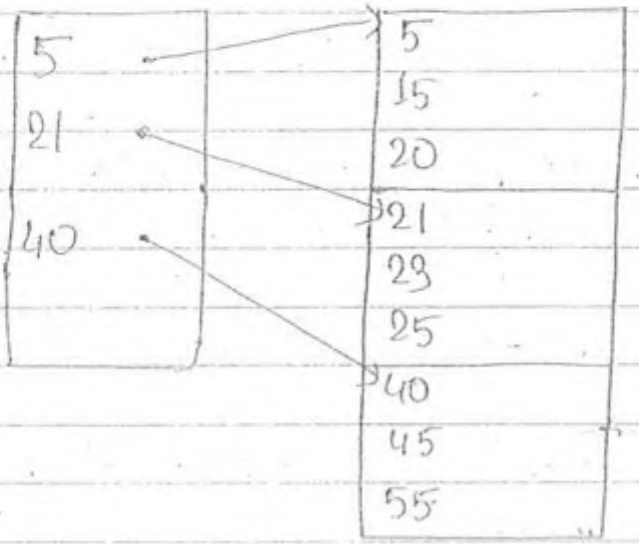
Categories of Index :-

[1] Dense Index :-
For every database record there should be corresponding entry in the Index file. (i.e. one to one mapping between Index file entries & DB file records)

No. of Index File Entries = No. of DB records

[2] Sparse Index:-

For set of DB records there should be single entry in the Index file.



No. of Index File Entries << No. of DB Record

Special Case:-

{No. of Index File Entries} = {No. of DB Blocks}

(Qeee) Blocks are 1000 bytes.
Records are 100 bytes of which 12 bytes takes keys & pointer take 8 bytes.
10000 records exist in the database.
How many no. of Dense & sparse Index blocks?

Block factor of DB file = $\frac{1000}{100}$ = 10 Record/block

No. of DB Blocks = $\frac{\text{No. of records}}{\text{Block factor}}$ = $\frac{10000}{10}$ = 1000 Block.

I/O cost without index ⇒ $\lceil \log_2 1000 \rceil$ = 10 Blocks

Or

N = 1000 Blocks

By using Indexing :-

Entry Size of Index = 12 + 8 = 20 Bytes

Block factor for index file = $\frac{1000}{\text{entry size}} = \frac{1000}{20} = 50$

Dense Index	Sparse Index
No. of Index File Entries – No. of D.B record = 10,000	No. of Index file Entries = No. of DB record = 1000
No. of Block = $\frac{10000}{50}$ = 200 Blocks	No. of Block = $\frac{1000}{50}$ = 20 Blocks
I/O Cost = $\lceil \log_2 200 \rceil + 1$ = 8 + 1 = 9 blocks	I/O cost = $\lceil \log_2 20 \rceil + 1$ = 5 + 1 = 6

Types of Indexes :-

[1] Primary Index :-

Index is said to be primary index if it satisfied following condition :-

(1) Conditions :-

(1) Search key used in the index file should be

used to physically orders the DB file.
[Basically 1st condition says that file should be "Ordered file"]

[2] Search Key should be primary key or Alternative key. [Key]

(2) Both dense & Sparse Primary Index are possible.

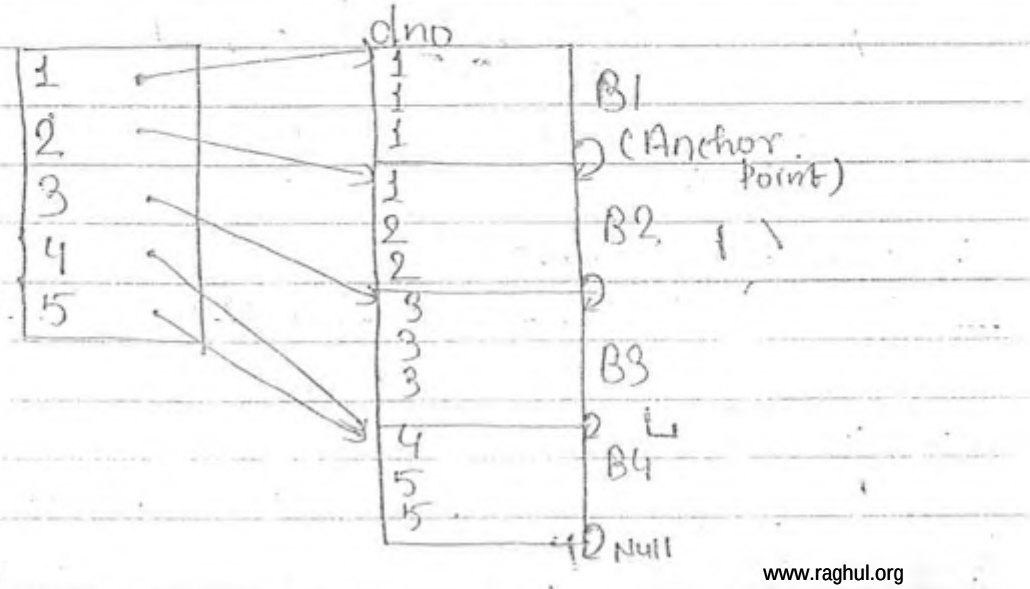
(3) At most one primary Index is possible.

[2] Clustering Index:-

(a) Condition:-

[1] Search key used in the index file should be used to physically orders the DB file. [Ordered File].

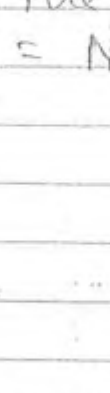
and [2] Search key should be Non-key [Non-KEY].



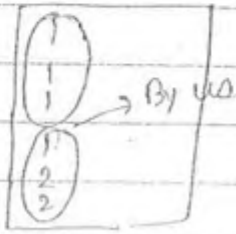
* Ordering of file is done on only one search attribute.
* For one database either we go for primary index or clustering index but not both.

* Block Anchors are required in clustering index:.

* If every employee comes under the same department then I/O cost = 1+N = N



Select *
From Emp
where dno = 1



(b) Always sparse.

(c) At most one clustering index is possible

if $\lceil \log_{10} N \rceil + 1$ { Select * From Emp where Eno = x } 50+

Or

N { Select * From Emp where pno = x } 50+

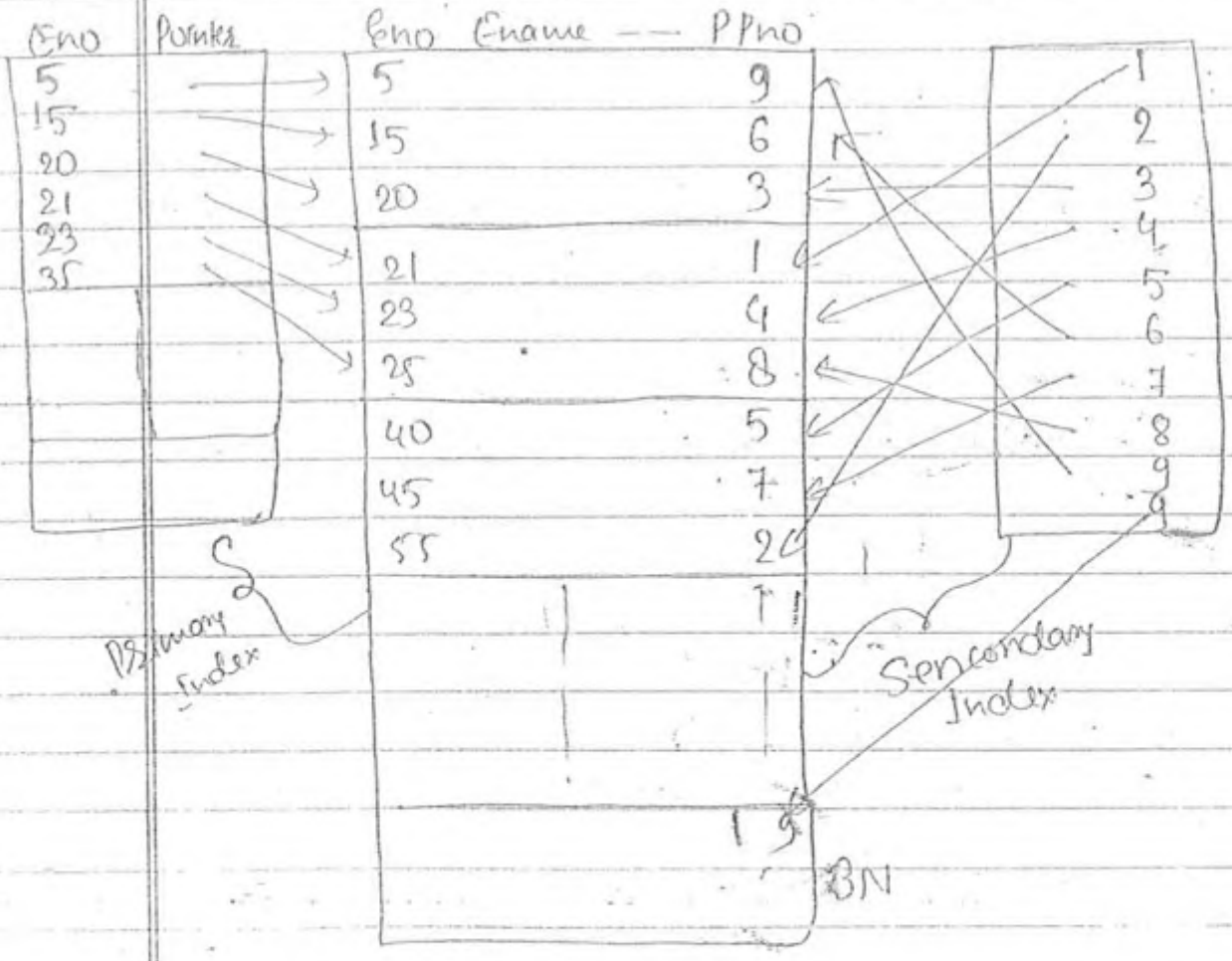
[8] Secondary Index:-

Secondary way of accessing the data if primary index or clustering index already exist.

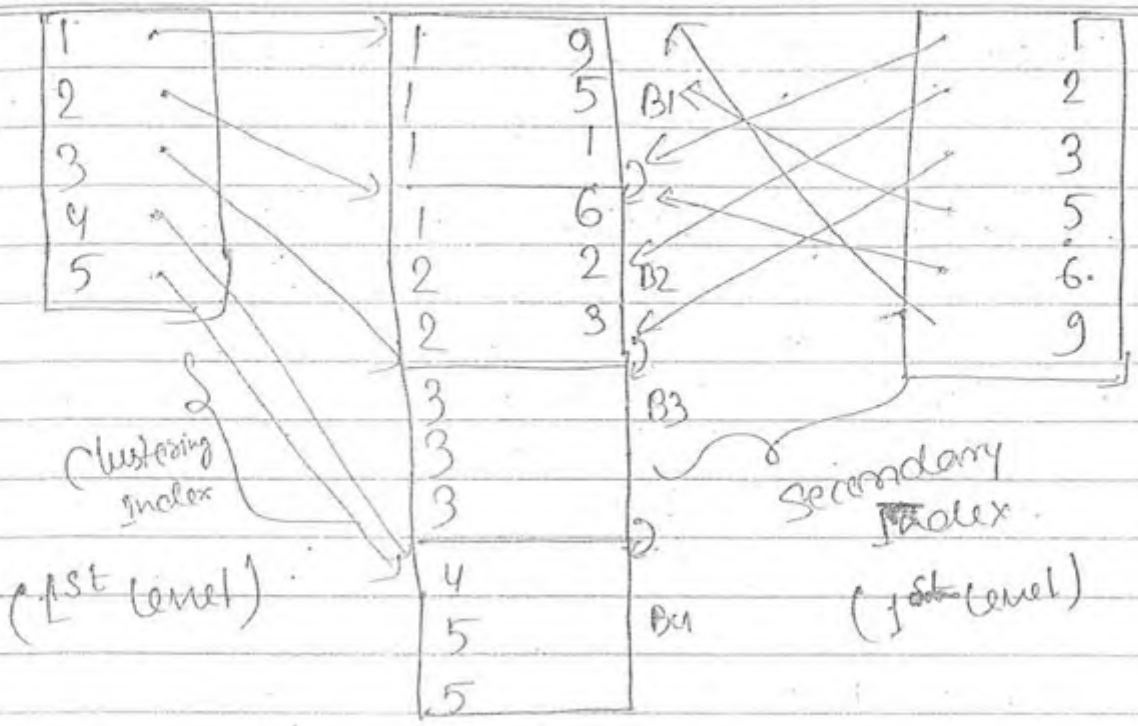
(a) Conditions:-

[1] Database not physically ordered based on search key used in the index file.
[UN-ORDERED FILE]

[2] Search key can be key or Non key.



called 1st level Index.)

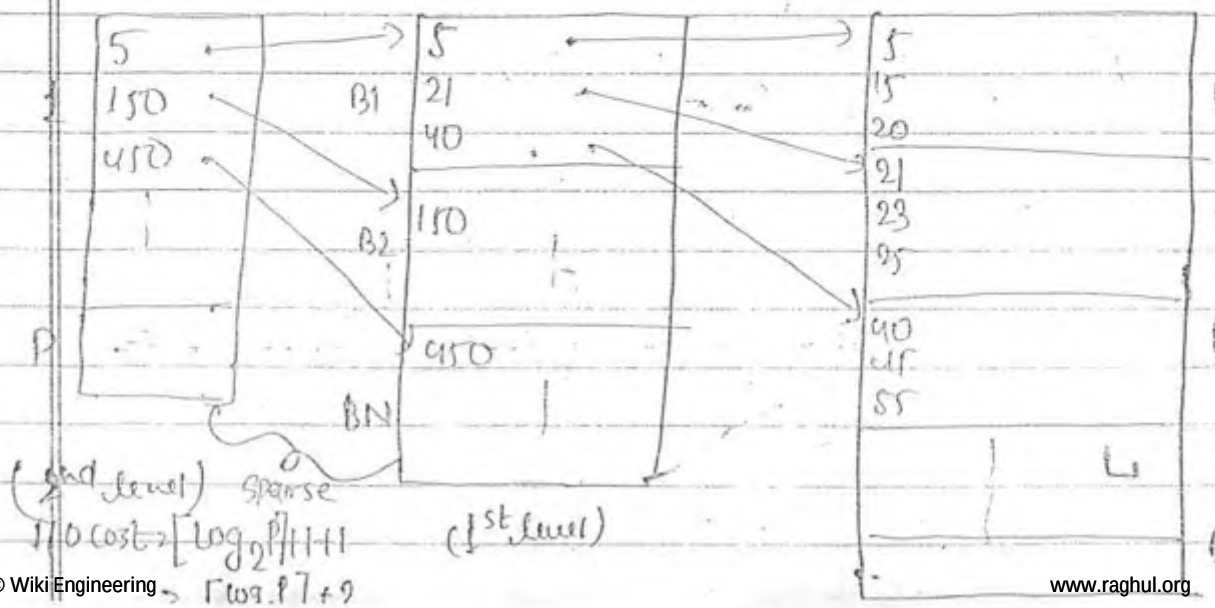


(b) Always Dense Index.

(c) More than one secondary Index possible.

MULTILEVEL INDEX!-

⇒ Index to the Index and so on until last level become single block.



No cost = $\lceil \log_2 P \rceil + 1$

$= \lceil \log_2 97 \rceil + 1$

⇒ Entry size of 2nd level Index = Entry size of 1st level Index

⇒ Block factor of 1st level Index = Block factor of 2nd level Index

⇒ 2nd level or 3rd level or ---- Always Sparse.

⇒ IO cost = No. of levels + 1

Page 55 Que 11)

16384 Records
Record = 32 byte
File Size = 6 byte
Ordered on Non key
Unspanned
Block Size = 1024 bytes
block pointer = 10 bytes
Secondary Index built on key.

Soln:-

16384 Records = 16384, Record Size = 32 byte
Block size = 1024 bytes

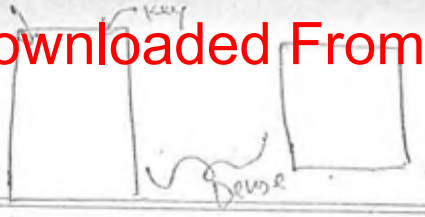
No. of entries in 1st level = No. of entries in D Record
= 16384

Block Size = 1024 bytes
Entry size = 6 + 10
= 16

Block factor = $\frac{1024}{16}$

$\frac{256}{2048}$
$\frac{1096}{0192}$ = 64 Entries per block

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



No. of blocks in 1st level = 256 block

⇒ No. of Entries in 2nd level = 256

Block Factor of 2nd level = Block Factor of 1st level =

No. of Blocks = $\frac{256}{64}$ = 4 Blocks

No. of Entries in 3rd level = 4

No. of block = 1

No. Blocks = {256+4+1} : Blocks

Ques If Ordered on key K

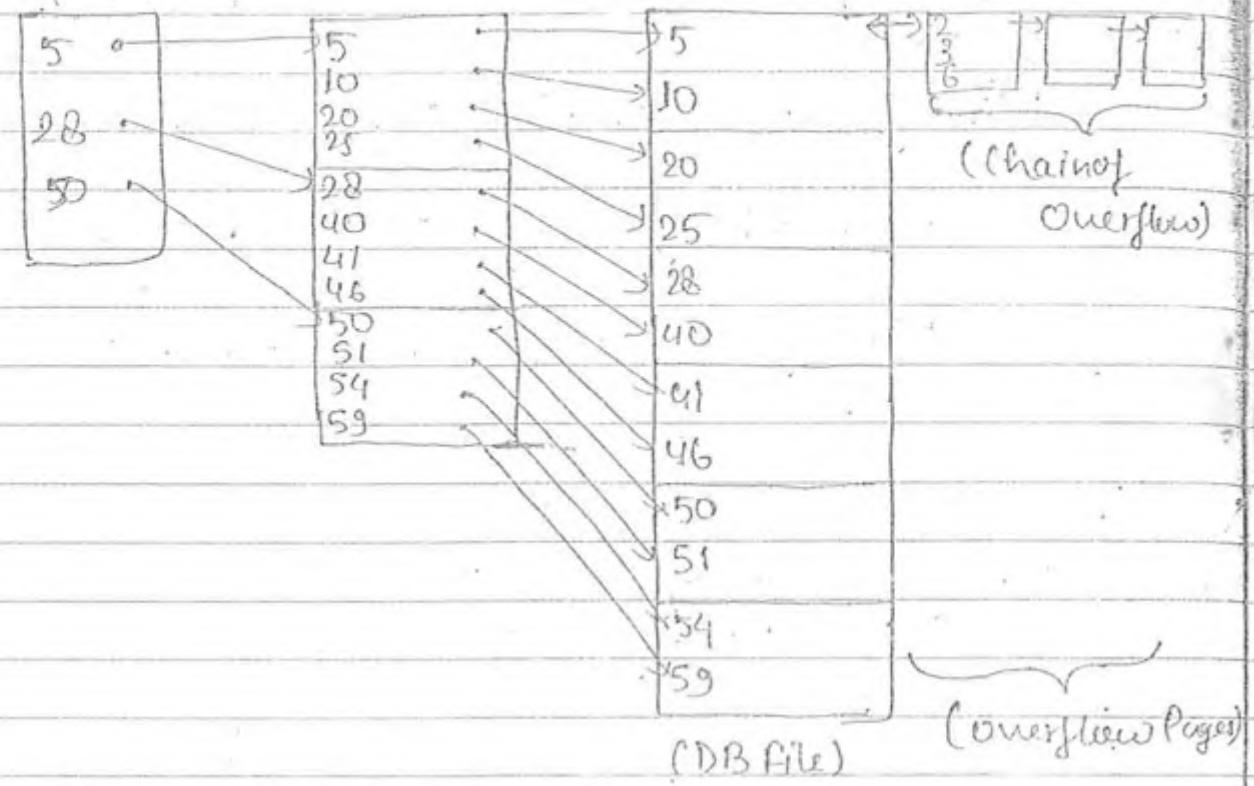
SPARSE Index build on key in the above Ques

then 1) No. of levels = 2

2) No. of Index blocks = 2

1/09/2011

STATIC MULTILEVEL INDEX:-



Limitation of Static Multilevel Index:-

⇒ **Insertion:-**

[1] **Shifting DB:-**
Entire DB file and Index file required to shift because of New insertion. i.e. (Entire Index file changed).

[2] **Overflow Page:-**
Result more I/O cost.
Worst case: O(N) blocks I/O cost

⇒ **Deletion:-**

[1] **Shifting DB:-**
Which results Entire Index file change.

[2] **Leave Space:-**
Some Index block may result of occupied (mean) that wastage of capacity).

Dynamic Multilevel Index:-

All the disadvantages of static multilevel index are addressed here i.e. about 80% to 90% disadvantages of static multilevel index are removed here.

Design Goals:-

[1] Insertion/Deletion:-

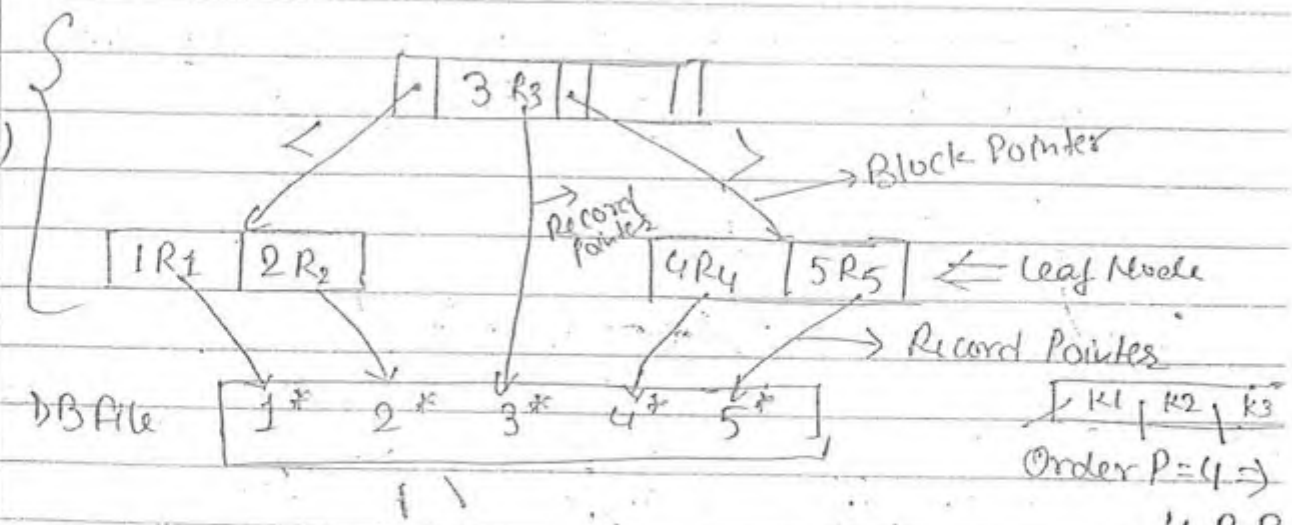
i.e. Insertion/deletion should not result Entire Index file changes.

[2] No overflow pages allowed.

[3] Becoz of deletion No Index Node should be less than 50% occupied.

* B Tree & B+ Tree comes under Dynamic Multilevel Index.

B Tree:-



4 BP
3 Key
3 RP

Record Pointer (OR) Data Pointer (OR) Value Pointer

Block Pointer (OR) Node Pointer (OR) Child Pointer (OR) Tree Pointer

I/O Cost To access any Record =

$$O(\log_p N)$$

N: No. of keys. = 5

P: maximum no. of child pointers in B-Tree Node = 3

To access all records:-

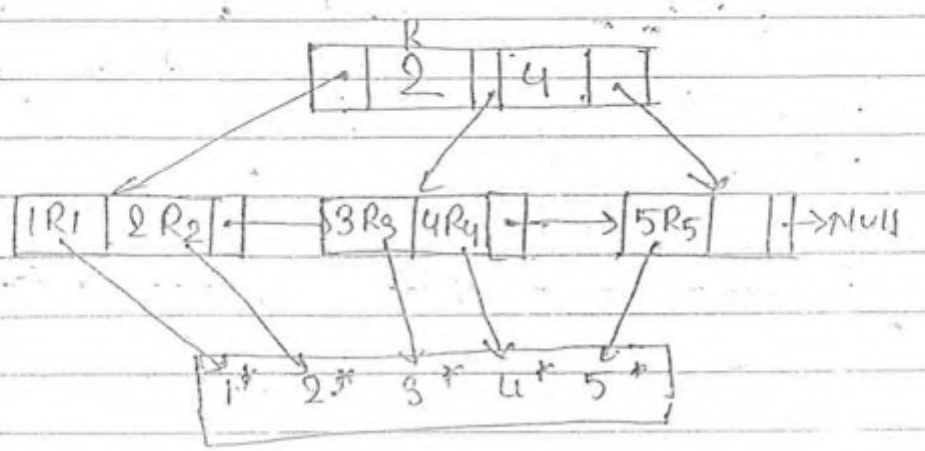
I/O cost = $O(2^M)$.

M = No. of index Nodes.

⇒ Not best Suitable for to access all records of the DB.

To avoide the problem in accessing all records in B tree we goes through B+ Tree.

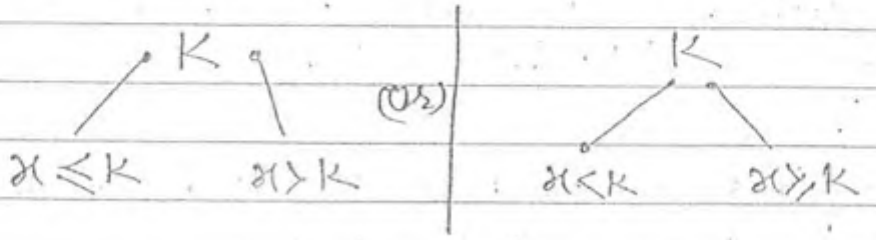
B+ Tree:-



⇒ Every key is these at leaf Node

⇒ Every leaf Node pointed to next leaf node.

- if K is a key then Block pointes point the value whose keys value x follows below pattern i.e.



⇒ I/O cost To access any record = $\Theta(\log_p N)$

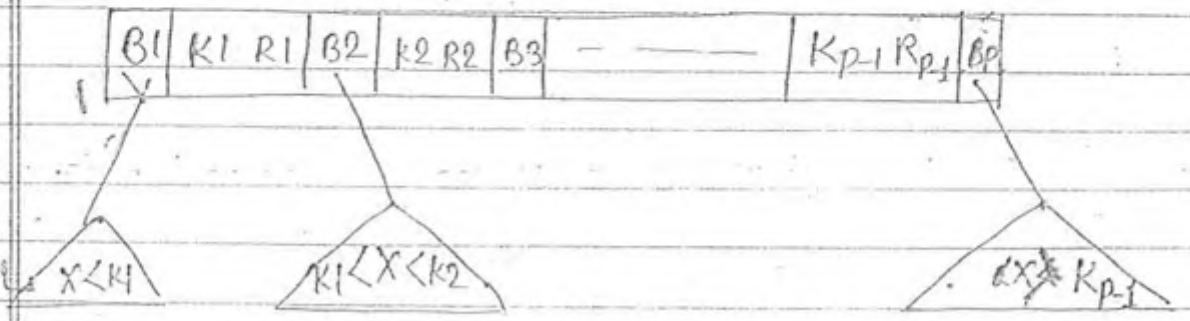
⇒ All records only required leaf Nodes.

B-Tree Definition :-

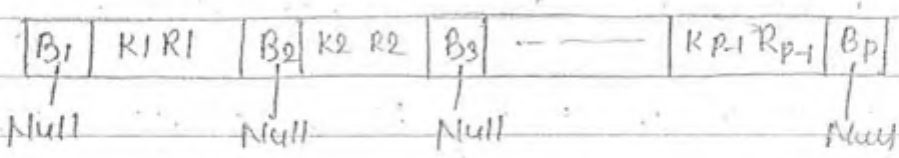
ORDER P = maximum no. of block pointes in B-Tree node

① Structure of Internal Node :-

P Bp + (P-1) keys + (P-1) Record Pointes ≤ Index Block



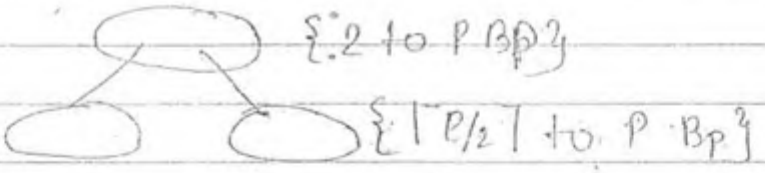
(2) Structure of leaf Node:-



(3) Keys within the node should be in Ascending order $\{K1 < K2 < \text{---} < K_{p-1}\}$

(4) Every Node Except Root should be atleast $\lceil p/2 \rceil$ Block Pointers at most p block pointers. (To ensure the 50% occupency condition).

(5) Root Node can be at least 2 BP at most P Block pointer.



(6) Every leaf node should be same level.

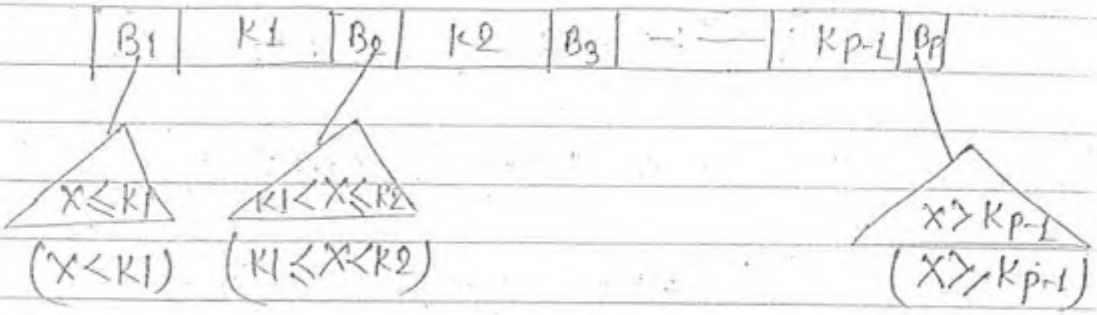
B+ Tree Definition:-

(1) Structure of Internal Node:-

Assume

Order P = maximum no. of Block Pointer in B+ tree internal node

p Block Pointers + (p-1) key ≤ Block



(2) Structure of Leaf Node:-

Order P: maximum no. of keys in B+ Tree leaf Node

P keys + P Record Pointers + 1 BP ≤ Block Size

(3) Keys within the node should be in Ascending order $\{K1 < K2 < \dots < Kp-1\}$

(4) Every Node Except Root should be atleast $\lceil P/2 \rceil$ Block pointers at most P block pointers.

(5) Root Node can be at least 2BP at most P Block pointers.

(6) Every leaf node should be same level.

Q) Internal node B^+ Tree order P = maximum no: of keys

$\therefore$ $(P+1)BP + P\,keys \leq$ Block Size.

Order P: min P keys, max 2P keys in B^+- Tree internal node.

$= (2P+1)BP + 2P\,keys \leq$ Block Size

Order P: min P keys max 2P keys in B-Tree node

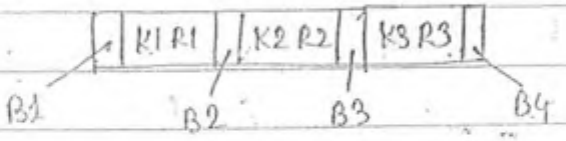
$= (2P+1)BP + 2P\,keys + 2P\,RP \leq$ Block Size

Order P: min P Bp max 2P Block pointers in B Tree Node

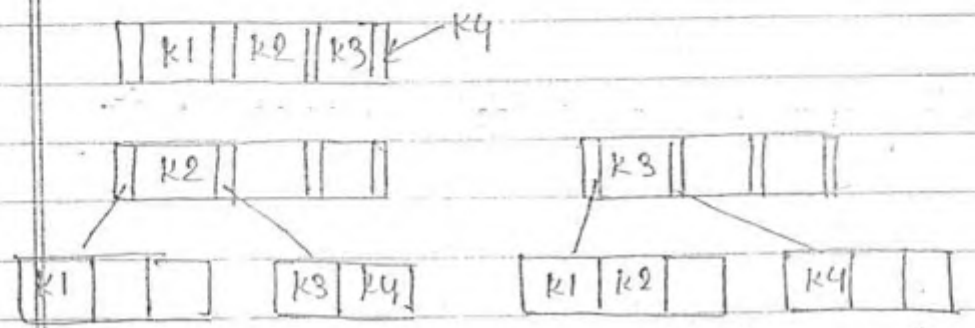
$(2P)BP + (2P-1)$ keys $+ (2P-1)$ record pointer $\leq$ Block size.

Q) Construct B-Tree with order P=4 and following sequence of keys.

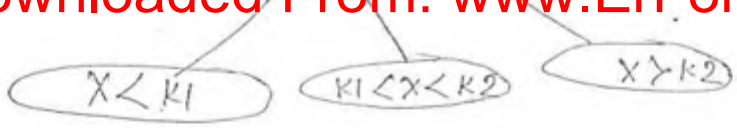
20, 5, 10, 3, 12, 18, 15, 45, 38, 69, 93



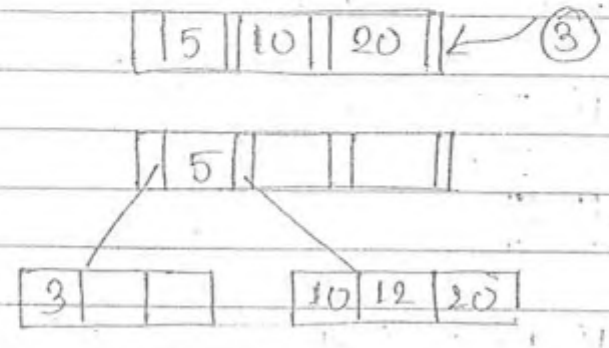
Node Splitting



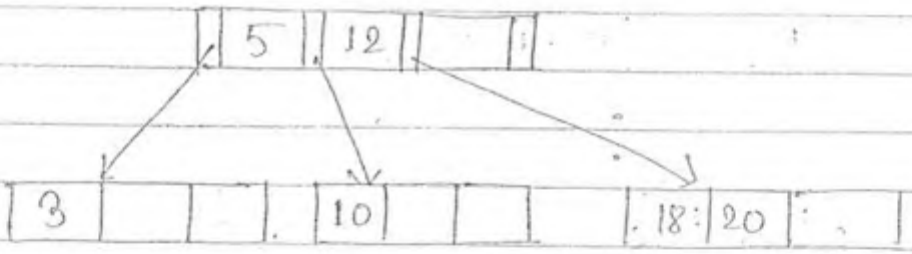
(Left biasing)



Balanced = Height Restricted =
= $O(\log_p N)$
i.e. height must not go higher then $\log_p N$ for balancing the tree.

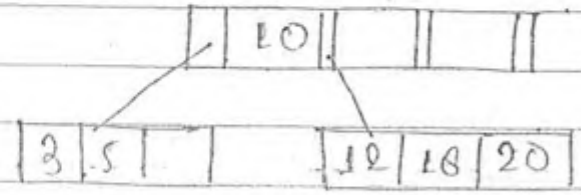


a) Node Splitting :-



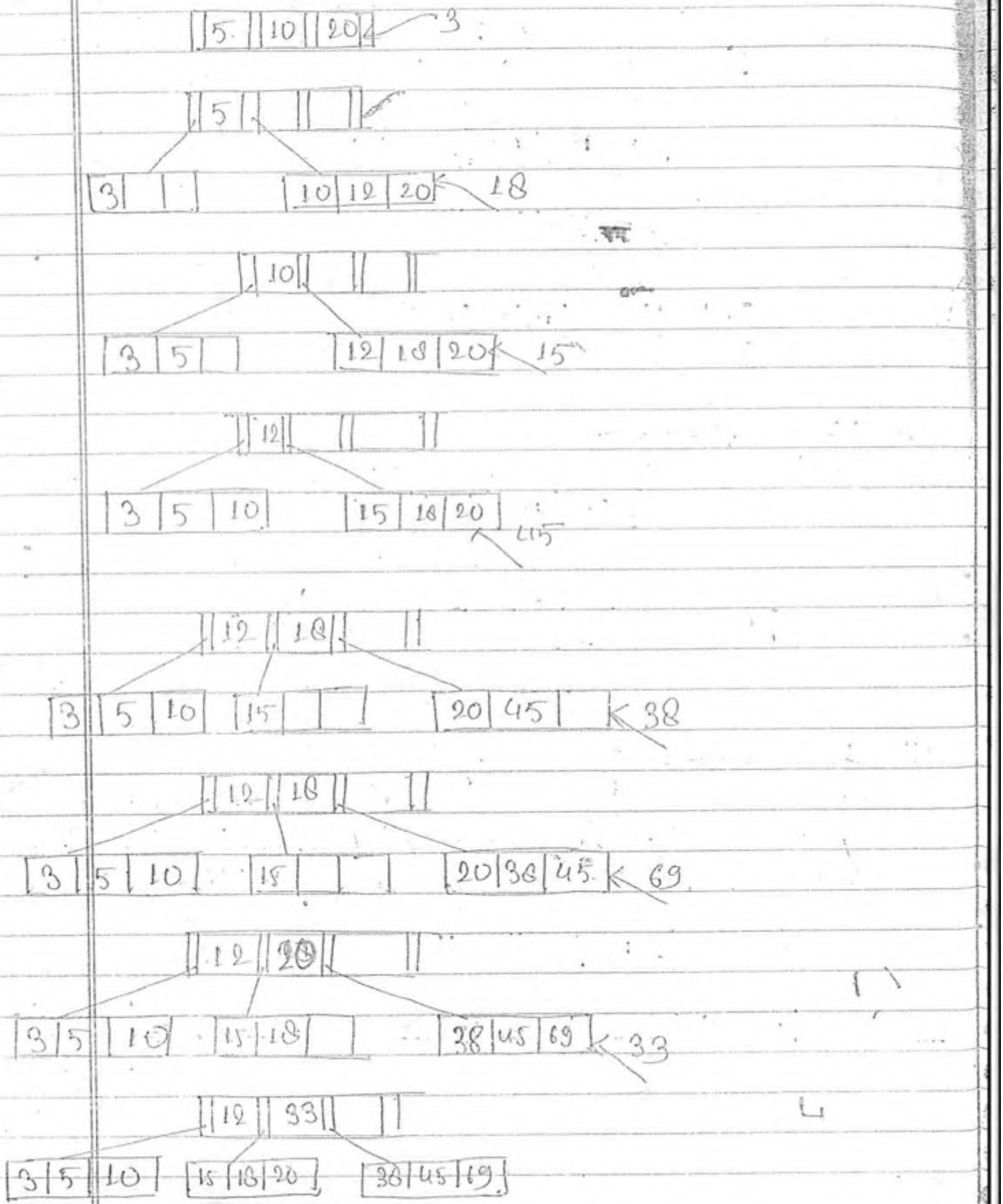
b) Redistribution of the keys :-

If any leaf Node consist some free position we can distribute key to accomodate New key in only Existing Nodes.



Try redistribution if not possible then go for Node splitting

By using redistribution



classmate Date Page

B-Tree:-

ORDER P: maximum no. of Bp in - B Tree Node.

Height/Level	Min. no. of Nodes	Min no. of B.P.	Min no. of keys
0/1	1	2	1
1/2	2	$2\lceil P/2 \rceil$	$2(\lceil P/2 \rceil - 1)$
2/3	$2\lceil P/2 \rceil$	$2\lceil P/2 \rceil^2$	$2\lceil P/2 \rceil(\lceil P/2 \rceil - 1)$
3/4	$2\lceil P/2 \rceil^2$	$2\lceil P/3 \rceil^3$	$2\lceil P/2 \rceil^2(\lceil P/2 \rceil - 1)$
⋮			
h-1/h	$2\lceil P/2 \rceil^{h-1}$	$2\lceil P/3 \rceil^h$ (minimum null pointers)	$2\lceil P/2 \rceil^{h-1}(\lceil P/2 \rceil - 1)$

Total min. no. of keys in B-Trees with order P & height h

$$1 + 2(\lceil P/2 \rceil - 1) + 2\lceil P/2 \rceil(\lceil P/2 \rceil - 1) + \cdots + 2\lceil P/2 \rceil^{h-1}(\lceil P/2 \rceil - 1)$$

$$1 + 2(\lceil P/2 \rceil - 1)\left\{1 + 2\lceil P/2 \rceil + \lceil P/2 \rceil^2 + \cdots + \lceil P/2 \rceil^{h-1}\right\}$$

$$1 + 2(\lceil P/2 \rceil - 1)\left\{\frac{\lceil P/2 \rceil^h - 1}{\lceil P/2 \rceil - 1}\right\} = 1 + 2\left\{\lceil P/2 \rceil^h - 1\right\}$$

Total min. no. of keys in B-Tree with order P & level l.

$$= 1 + 2\left\{\lceil p/2 \rceil^{l-1} - 1\right\}$$

(Qn) Total min. no. of keys in B-Trees with order P & level l.

a) $1 + 2\left\{\lceil \frac{p}{2} \rceil^{l} - 1\right\}$. {Here levels starts from 0}

b) $p^{l+1} - 1$

c) $p^{2l+1} - 1$

d) $\left\{\lceil \frac{p}{2} \rceil^{2l+1} - 1\right\}$

Height/level	max. no. of nodes	maximum no. of Block Pointer	maximum no. of keys
0/1	1	P	P-1
1/2	P	P^2	P(P-1)
2/3	P^2	P^3	$P^2(P-1)$
3/4	P^3	P^4	$P^3(P-1)$
h/h+1	P^h	P^{h+1} (maximum Null pointers)	$P^h(P-1)$

Total maximum no. of keys in B Tree with Order P & height h.

$$(P-1) + P(P-1) + P^2(P-1) + \cdots + P^h(P-1)$$

$$(P-1)\{1 + P + P^2 + \cdots + P^h\}$$

$$(P-1)\left\{\frac{P^{h+1}-1}{P-1}\right\} = \boxed{P^{h+1}-1}$$

$\Rightarrow$ Total maximum no. of keys in B Tree with order P & level l.

$$\boxed{= P^l - 1}$$

Height of the B-Tree with N keys :-

$$N = 1 + 2\left\{\lceil P/2 \rceil^h - 1\right\}$$

$$\Rightarrow \boxed{h = \log_{\lceil P/2 \rceil}\left\{\left(\frac{N-1}{2}\right) + 1\right\}} \Rightarrow \text{maximum possible height with N keys}$$

$$N = P^{h+1} - 1$$

$$\Rightarrow \boxed{h = \log_P (N+1) - 1} \Rightarrow \text{minimum possible height with N key}$$

* In B Tree :- Minimum Null pointes = $2\lceil p/2 \rceil^h$

Maximum Null pointes = p^{h+1}

* Height of B Tree :- $O(\log_p N)$

where:-

N = No. of keys

P = Orders

Qes Orders P is maximum no. of keys in B Tree node. How many no. of maximum nodes & keys in B-Tree with orders P = 3 and level l = 4

Orders P = 3 l = 4

One block can accomodate :-

P keys + (P+1) Block pointes + P recordpointer ≤ Block

3

level	max no. of nodes	max no. of B.P	max no. of keys
1	1	4	3
2	4	4^2	4*3
3	4^2	16*4	16*3
4	$4^3=64$	64*4	64*3
	85	BP pointed Null	255

Que) Orders P: max no. of Bp in B Tree node.
How many minimum no. of keys & Nodes in B-Tree with orders P=4, Level=4

One block can accomodate:-

$$(P)BP + (P-1).Keys + (P-1)R.P \leq Block$$

Level	minimum no. nodes	minimum no. of Block Pointer	minimum no of keys
1	1	2	1
2	2	$2*\lceil P/2 \rceil = 2*2$	$2*1$
3	4	$4*2$	$4*1$
4	8	$8*2$	$8*1$
	15	Pointed Null	15

page-55
Que 12

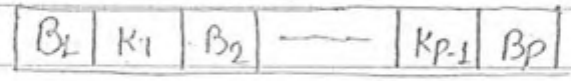
P: maximum no. of Tree pointers in B-Tree Node
Keys = 10 bytes
Block = 512 bytes
data Pointer = 8 bytes
Block Pointer = 5 byte
maximum value of P = ?

$$P*Bp + (P-1)Key + (P-1)RP \leq Block\ size$$
$$P*5 + (P-1)10 + (P-1)8 \leq 512$$
$$23P \leq 512 + 18$$
$$23P \leq 530$$
$$P = \left\lfloor \frac{530}{23} \right\rfloor = 23$$

2/09/11

B^+ Tree :-

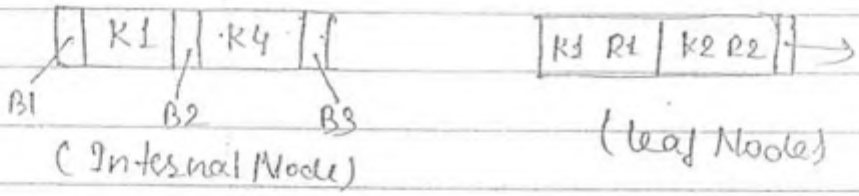
Internal Node :-



P Block Pointer + (P-1) keys

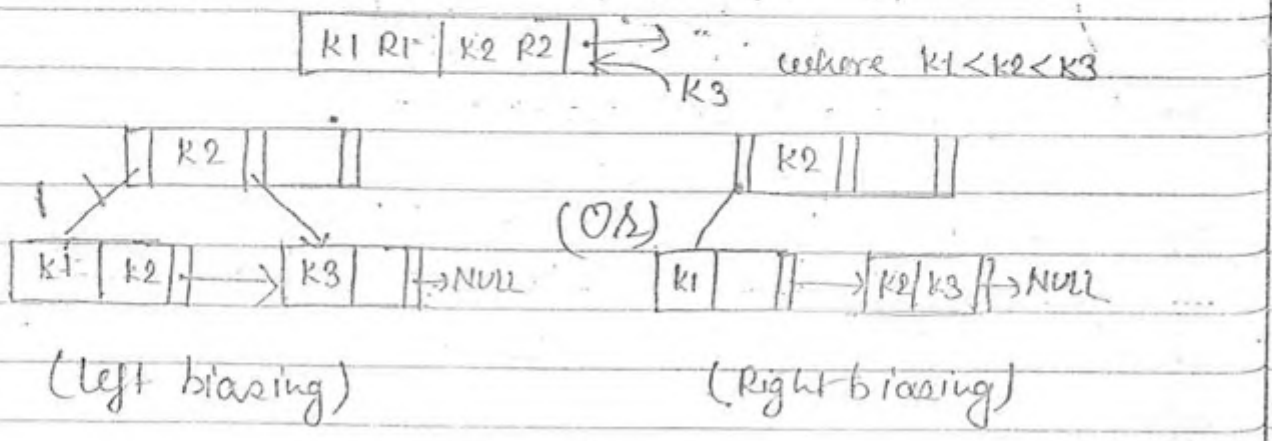
Page 56
Que 16) Order of internal Node = 3 (maximum no. of Tree pointers)

Leaf Node = 2 (maximum no. of data item)



Keys Inserted = 10, 3, 6, 8, 4, 2, 1

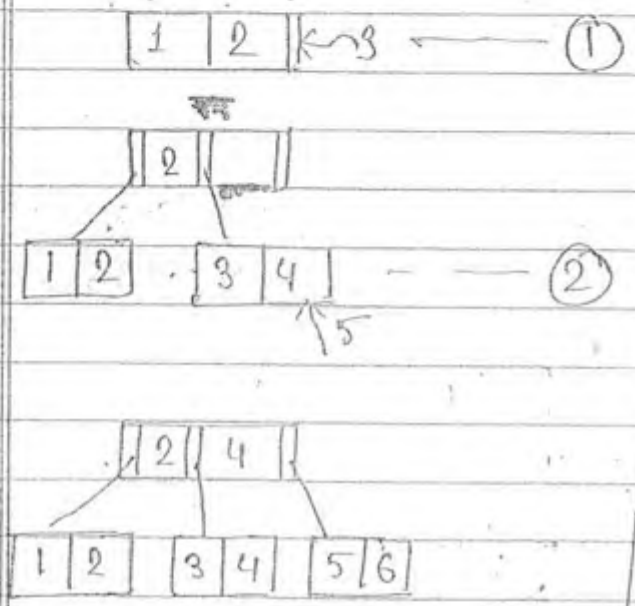
NODE SPLITTING IN B^+ Tree :-



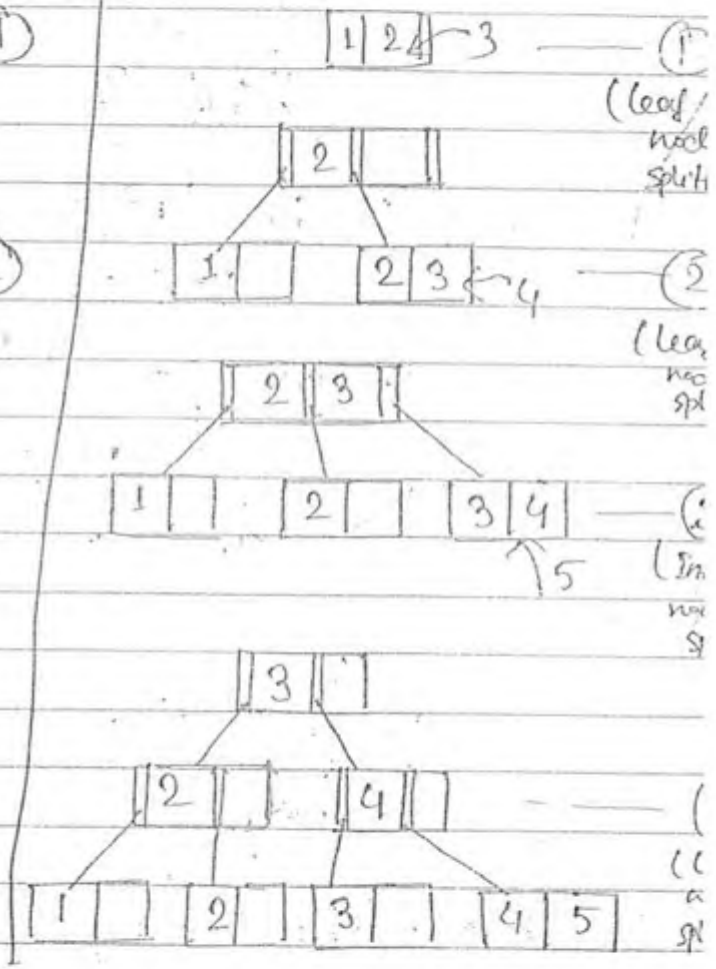
If keys are in Ascending Order

1, 2, 3, 4, 5, 6, ——— , N

By Left Biasing. How many node splitting



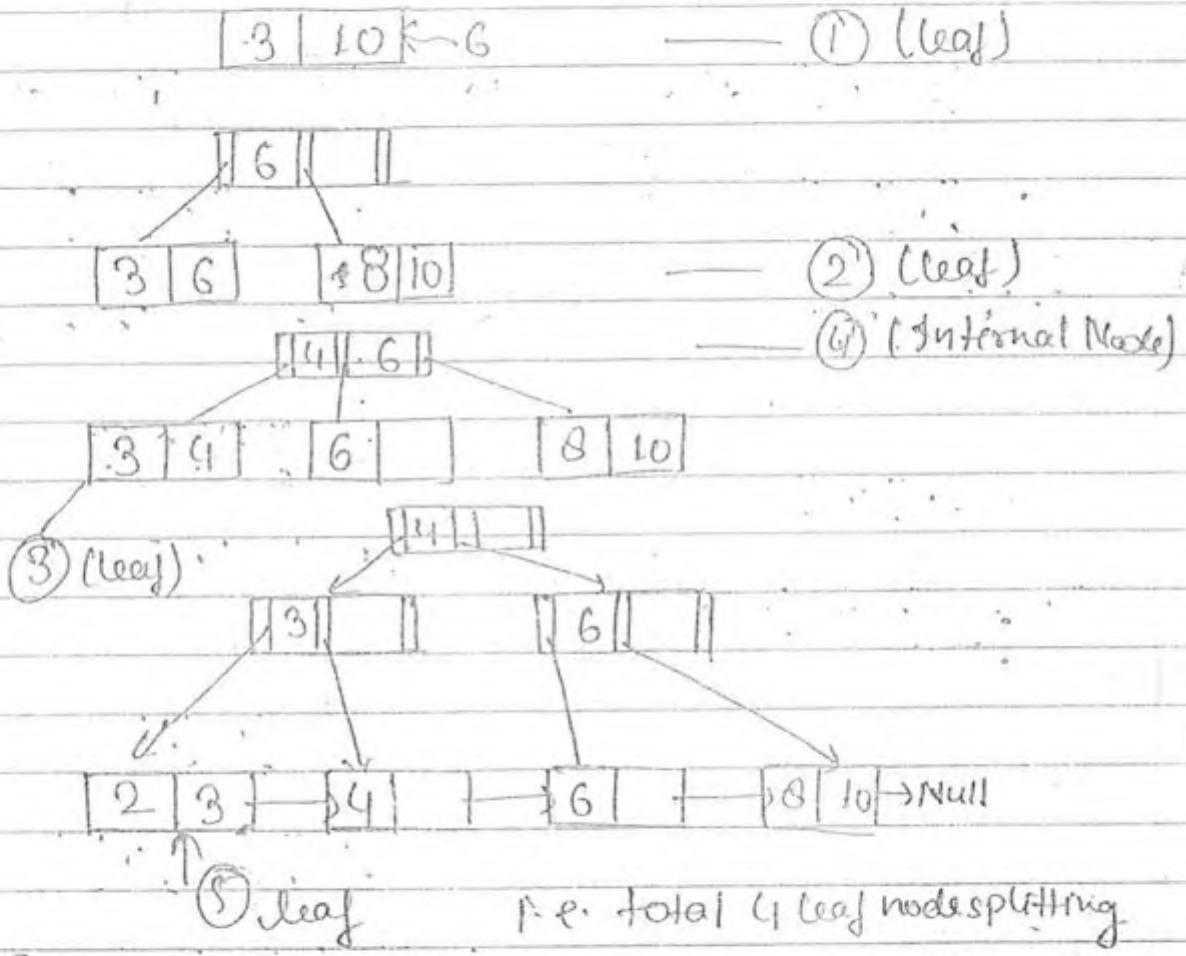
By Right biasing No. of node splitting.



⇒ If keys are in Ascending Order then Right biasing gives the worse Node splitting then the Left biasing i.e. In the above example we see that upto insert 5 element we get 2 node splitting by using left biasing and we get 4 node splitting using right biasing.

⇒ If keys are in Descending order then Left Biasing gives more Node splitting then the right Biasing.

In the above question 10, 3, 6, 8, 4, 2, 1 more than half elements are in decending orders. So, we go through left biasing for obtaining maximum no. of node splitting.

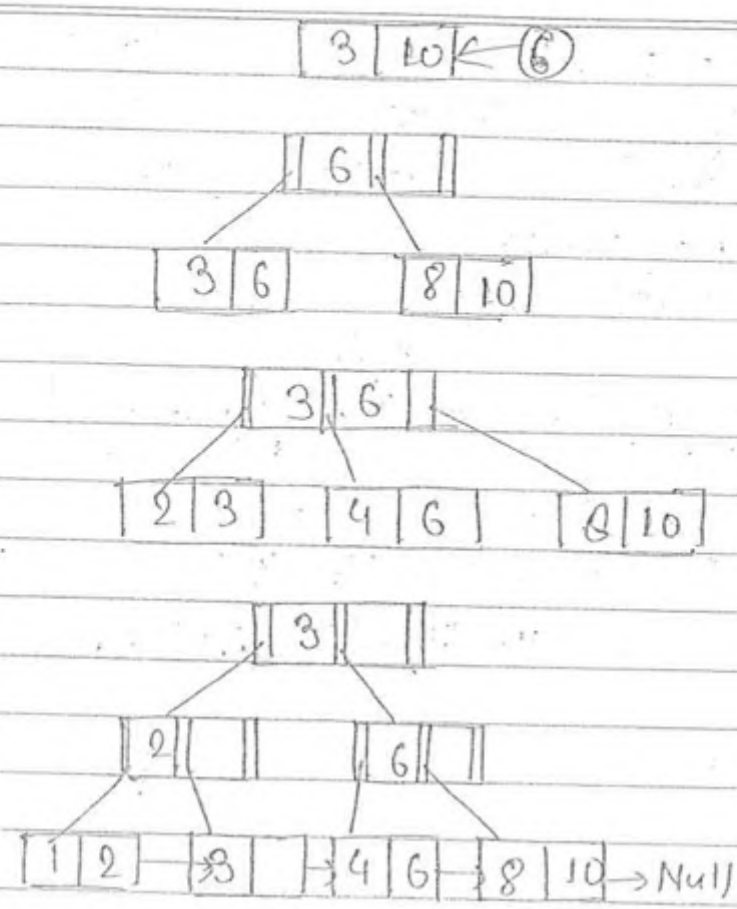


i.e. total 4 leaf node splitting

⇒ Redistribution always gives less node splitting. So, whenever question asked like minimum no. of node splitting we always go through Redistribution method.

By apply redistribution method :-

We get 3 leaf node splitting & 1 internal node splitting.



(Que 5)

Keys = 8 bytes
Block Size = 512 bytes
Index Pointers = 4 bytes

$PBp + (P-1)\,keys \leq Block\ size$

$P \times 4 + (P-1)\,8 \leq 512$

$4P + 8P - 8 \leq 512$

$12P \leq 512 + 8$

$12P \leq 520$

$P \leq \frac{520}{12}$

$P = \left\lfloor \frac{520}{12} \right\rfloor$

Ques) Record = 5,00,000
Record Size = 200 bytes
Search key Size = 15 bytes
Bp = Rp = 5 bytes
Block = 1024 bytes

ORDER of B+ Tree :— No. of pointers

Internal Node: P Bp + (P-1) keys

Leaf Node: (P-1) keys + (P-1) Rp + 1 Bp

$P(5) + (P-1)15 \leq 1024$
$5P + 15P - 15 \leq 1024$
$20P \leq 1024 + 15$
$20P \leq 1039$
$P \leq \frac{1039}{20}$

Ques) $P Bp + (P-1)\ keys \leq Block\ Size$

Ques) Leaf Node Orders P :
Maximum no. of (value, data Rp) pairs it can hold

$P * Keys + P * Rp + 1\ Bp \leq Block\ Size$

Que 13) Largest possible order of Non leaf Node (Internal Node)

$P B_p + (P-1)$ keys $\leq$ Block size

Example: Suppose blocks are sized so that they can either hold 5 record of relation R or be used as B+ Tree Node with 10 keys & 11 pointers. If R has 10000 Records what is the smallest no. of blocks that could be used to store R and a sparse B+ Tree Index on the key of R.

5 record / block = 10 keys & 11 pointers

If Data Block | If Index Block

Index Block [10 key + 11 ptrs]

→ min occupancy (except root) $(\lceil P/2 \rceil B_p)$ [6 Bp + 5 key]

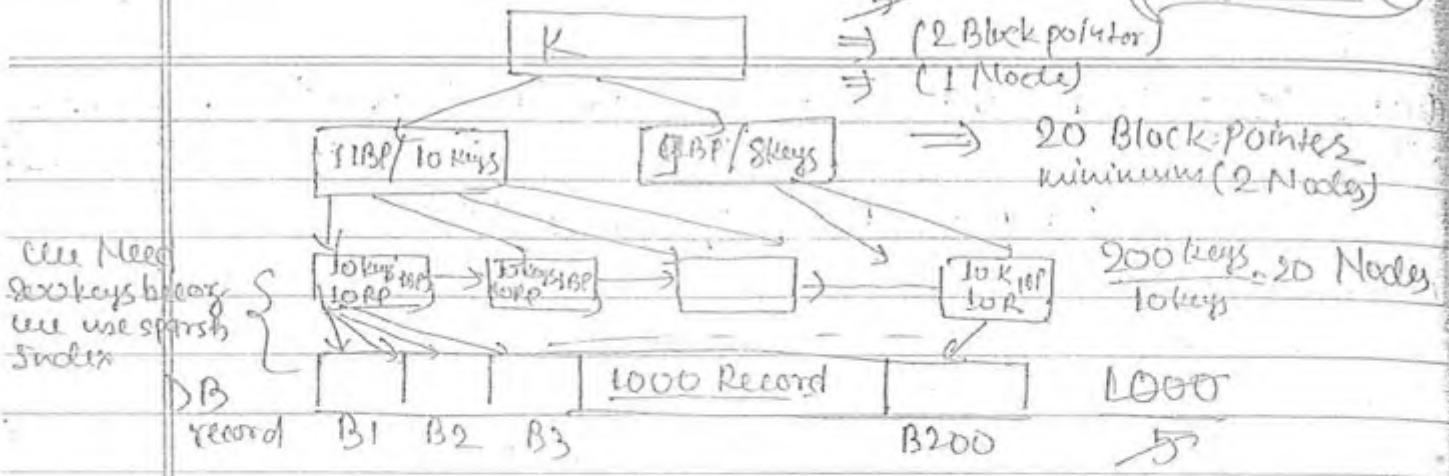
→ max occupancy $(P B_p)$ [10 keys + 11 ptr]

⇒ Smallest No. of Index blocks are possible with only Maximum occupancy per N

⇒ Largest No. of Index blocks are possible with only minimum occupancy per Node

Total no. of block = 1000

⇒ (2 Block pointer)
⇒ (1 Node)



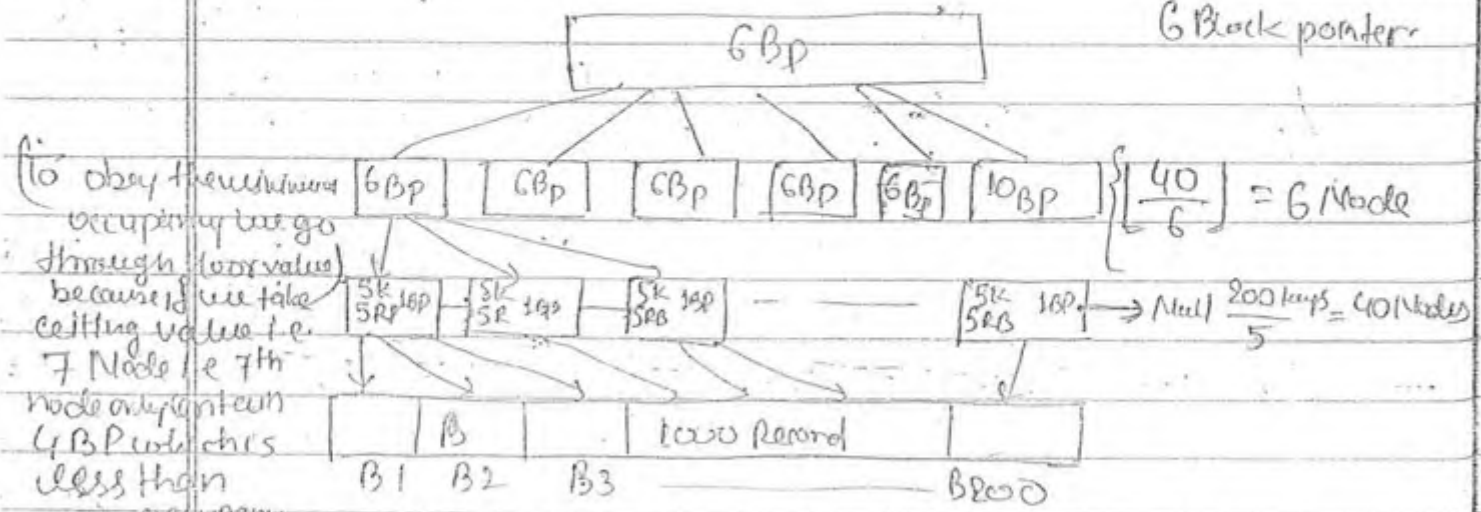
Total No. of Nodes/Blocks -

200 + 20 + 2 + 1 = 223

- 200 → Data Base level
- 20 → 1st Index
- 2 → 2nd Index
- 1 → root level

Largest No. of Blocks + Sparse Index =

Note! Minimum B+ Tree Node capacity.



(To obey the minimum occupancy we go through floor value because if we take ceiling value i.e. 7 Node i.e 7th node only contain 4 BP which is less than min occupancy so we put 6 Node Last Node contain.)

Total No. of Node! — 200 + 40 + 6 + 1 = 247

- 200 → DB Level
- 40 → 1st Index
- 6 → 2nd Index
- 1 → root level

Identify Smallest No: of Blocks for Dense Index!-

10 Block Point = 1 Node.

$\lceil \frac{1000}{11} \rceil = 10$

1000 keys / 10 keys = 100 Nodes

$\frac{1000}{5} = 200$ Blocks

Identify Largest No. of Blocks for Dense Index!-

TUPLE RELATIONAL CALCULUS :-

⇒ It is one of the Non Procedural Query language.

Non Procedural Query language :- Here what to retrieve from DataBase, we least bothered as how to retrieve from DB.

⇒ First Order Logic formula.

⇒ Formate of the Query ⇒ $\{T / P(T)\}$

(It results Tuples T such that P(T) is true)

Example: Retrieve the suppliers whose rating greater than 10.

$$\{S / S \in Suppliess(S.rating > 10)\}$$

(Free Tuple Variable) — Formula

$$\equiv \sigma_{rating>10}(Supplies)$$

Query: Retrieve the Sid's of suppliess whose rating greater than 10.

$$R(\quad) :- \pi_{Sid}\left(\sigma_{rating>10}(Supplies)\right)$$

$$\equiv$$

TR $\{T / \exists S \in Suppliess(S.rating > 10 \wedge T.Sid = S.Sid)\}$

* Variable can projected.

⇒ Bounded Tuple Variable:- Tuple variable set to be bounded only if ∃, ∀ used before Tuple Variable

S ∈ Suppliers (Free Tuple Variable)

∃S ∈ Suppliers (Bounded Variable)

Query Retsieve Sid's of suppliess who supply some red part.

$$\Pi_{Sid}\left(Catalog \bowtie \sigma_{Color=Red}(Parts)\right)$$

$$\{T \mid \exists C \in Catalog (\exists P \in Parts (P.Col = RED \wedge P.Sid = C.Sid) \wedge T.Sid = C.Sid)\}$$

Query Retsieve Sname of the Supplies who supplied Red or green part.

$$\{T \mid \exists S \in Supplies (\exists C \in catalog (C.sid = S.Sid \wedge \exists P \in Parts ((P.Col = RED \vee Red\ Green \wedge P.Pid = C.Pid)) \wedge T.Sname = S.Sname)\}$$

(4) (3) (2,1) (2,3) (4)

Query [Q3] Retsieve Sid's of suppliess who supplied Red & green parts

$$\{T \mid \exists C \in Catalog (\exists P \in Parts ((P.Col = Red \wedge P.Col = Green) \wedge P.Pid = S.Sid \wedge T.Sid = C.Sid)\}$$

(3) (2,1) (1) (3)

Above query is wrong because we can't take intersection

$\{T / \exists C1 \in Catalog (\exists P1 \in Parts (P1.Colour = RED \wedge P1.Pid = C1.Pid) \wedge$

$\exists C2 \in Catalog (\exists P2 \in Parts (P2.Color = Green \wedge P2.Pid = C2.Pid) \wedge (C1.Sid = C2.Sid) \wedge$

$T.Sid = C1.Sid)) \}$

Query 2 Retrieve Sid of the Supplies who supplied Some Red part but not green part.

$\cup : \vee$

$\cap : \wedge$

$- : \neg$

$/ : \forall$

$\{ T / \exists C1 \in Catalog (\exists P1 \in Parts (P1.Color = RED \wedge P1.Pid = C1.Pid) \wedge$ (1)

$\neg \exists C2 \in Catalog (\exists P2 \in Parts (P2.Colour = GREEN \wedge P2.Pid = C2.Pid)) \wedge$ (3, 2)

$T.Sid = C1.Sid) \}$ (1)

Query 1 Retrieve Sid's of the Suppliers who supplied Every Red part.

$$\pi_{Sid, Pid}(Catalog) / \pi_{Pid}(\sigma_{Col=Red}(Parts))$$

$\Rightarrow \forall t(P(t))$ ⇒ Tuples that are for all tuples if t such that P(t) is true.

$\{T/\exists C2 \in Catalog(\forall P \in parts(\exists C1 \in Catalog(P.col = RGD \wedge C1.Pid = P.P \wedge C1.Sid = C2.Sid)) \wedge T.Sid = C2.Sid)\}$

(1) (2) (3) (3) (2) (1)

{For all which is equal to division}

Query) $\{S/\exists(S \in Suppliers)\}$ — Free variable

⇒ O/P: Infinite : retrive tuples thise are not belongs to suppliers table.

⇒ Unsafe Query. (means results of Query Infinite)

$\{T/\exists S \in Supplies \wedge T.Sid = S.Sid\}$

⇒ O/P: Empty

⇒ Safe Query

* Basic Relation Algebra Expressive Powers = Safe TRC Expressive Power

Desk Book

SQL

⇒ ANY IN { }
ANY > ANY
ALL
EXISTS(———) { it only check inner condition is True or False. }

⇒ INNER JOIN : NATURAL JOIN
CROSS JOIN : Cross Product.
UNION JOIN : RUS

A	B
1	2
2	1

⋈ U

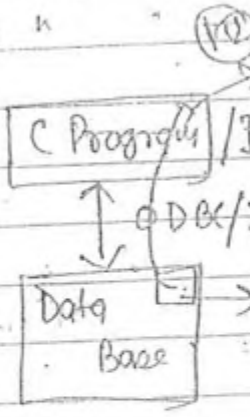
A	B
2	2
2	2

=

A	B
1	2
2	1
2	2
2	2

⋈ = empty

⇒ ODBC :- (Open Data Base Connectivity) (Microsoft)
JDBC :- (Java " ") (Sun Microsystem)



(M1) take less time then M2 to execute the Query

DELTA Page No. Date:

$10^9 \Rightarrow ?$.

$(10)^9 \times \log_{10} 10^9 \, d = 10^9 \times \log_{10} 10^9 * \frac{10}{3000}$ ms.

$= 10^9 \times 9^3 \times \frac{1}{300} \Rightarrow 3 \times 10^7$

$= 3000,000,0$