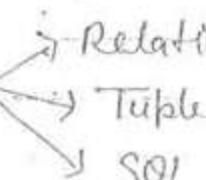


DBMS

(9)

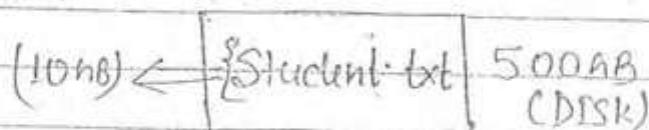
18/08/2011

Contents:-

- ⇒ Introduction & Integrity Constraints.
Referential Integrity Constraints
ER model.
- ⇒ Schema Refinement : [2+2] marks
(Normalization)
- ⇒ Transactions & concurrency control. [2] marks
- ⇒ Query lang 
Relational Algebra
Tuple Relational Calculus
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

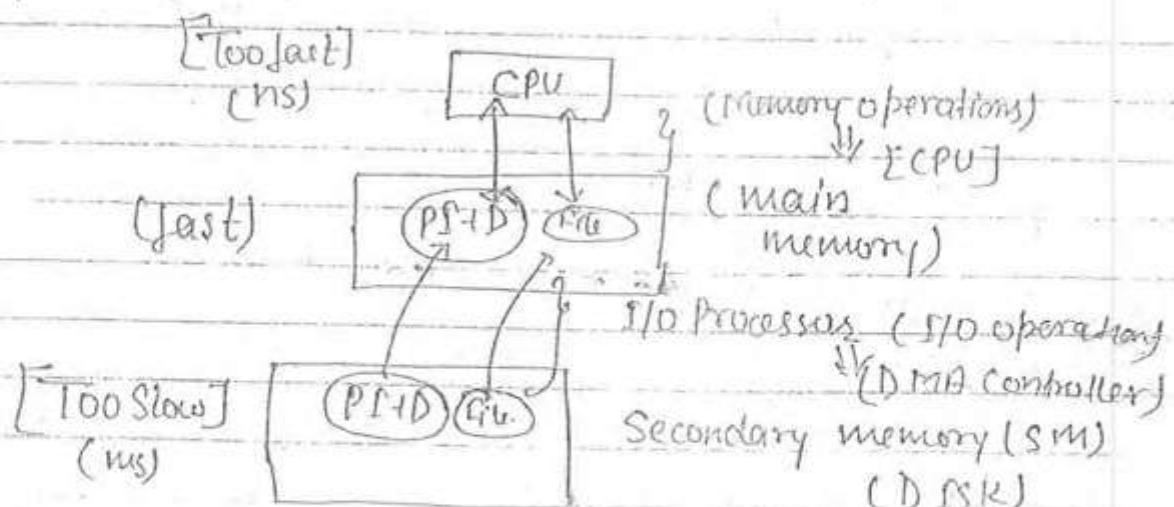
[Index]

	sid	b/w 20 to 25
K ₁	21	B ₁
K ₂	20, 21, 22, 23, 24, 25	B ₂
K ₃	21	B ₃
⋮	⋮	⋮
M	(M << N)	B _H

use stored programming concept in filesystem as well as in DB

Stored Programming Concept :-

Program + Data Required to execute program should be stored in High speed 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

Without Index 500

I/O cost

500

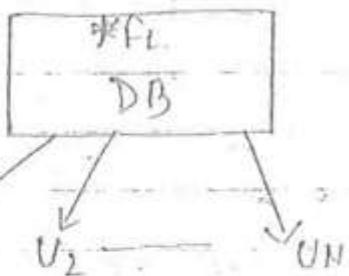
With Index 30 index pages + 500

30 + 1 \rightarrow To access current p

we can't

3) Concurrent Access -

* less concurrency level becoz of file level locks.



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

- operating system lock the file ^{entire} at the time of update.

Drawback of filerlevel Concurrency Control:-

* [file]

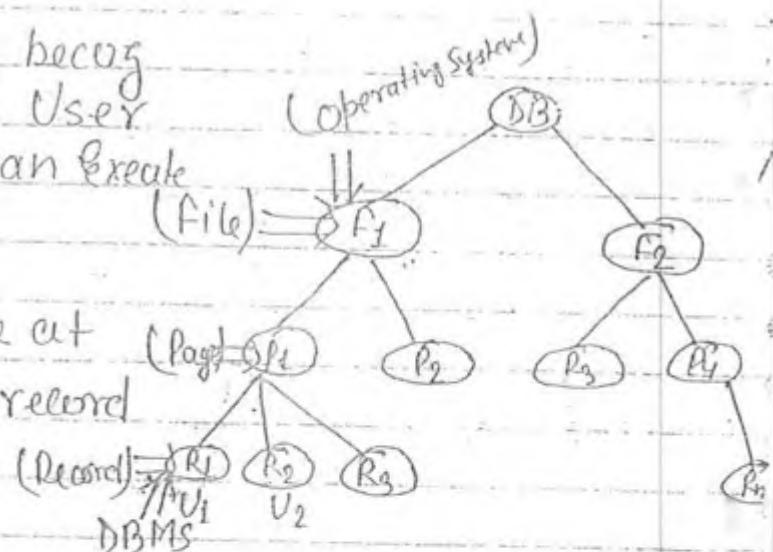
Lock the file	S1 S2 S3 S4 S5	Records
		If user U ₁ ! update S ₁ U ₂ : Read S ₃

Here is no inconsistency
Still file level locks
Not allowed to
execute U₁ & U₂ operations
simultaneously.

Benefit of DBMS in concurrency:-

* Record level locking because of low-level locking User U₁ & U₂ 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 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 every 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, Address)

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

Advantages of DBMS

VIEWS

[Virtual Table]

View StudFac (Sid , Sname, Marks)

StudInfoAdmin (Sid, Sname, Marks)

File

Sid Sname Marks DOB Address PhoneNo

One of the Data Model -

Relational DBMS-

Every data should be stored in proper Tabular formate.

Table-

Collection of rows and columns.

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

Relational instance → Field (OS) Attribute.

(! Set of tuples in the table)

Sid	Name (branch)	→ Field (OS) Attribute.
S ₁	A	CS
S ₂	A	IT
S ₃	B	CS

arity } No. of attributes
(OS) } in the Table.
Domain }

Key ! Sid
key ! Sid 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

(OS)

Relational Schema

Student (Sid Name branch)

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

Codd's Rule-

Every tuple should be differentiate from other tuple.
(OS)

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

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

Key I-

Min. no. of attributes used to differentiate 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 I-

Key forms with two or more attributes.

Simple key I-

Key forms with single attrib.

* A relation may contain more than one key.

Example -

Sid	Sname	Pno	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			

(Unexisting)

(Unknown)

Assume Constraints - No two Students with same DOB & Fname.

Keys - { Sid, Pno, Lno, (DOB Fname) }

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

i.e. If ABC ! key i.e. if (DOB Fname) ! key
then A is not key then

proper subset of ABC's
not allowed as key but
ABC is a key

B	"	"	"	Alone DOB is not key
C	"	"	"	Alone Fname is not key
AB	"	"	"	then only if (DOB & Fname) together uniquely different
BC	"	"	"	the tuple then only they are keys.
AC	"	"	"	

* All the keys collectively called as Candidate Key.

Candidate key - { Sid, Pno, Lno, (DOB Fname) }.

* Candidate key is same as key

i.e. Sid is one of the C-key of Pno, LNo, (DOB Fname) b/c
of these are C-key

* Prime Attribute-

Attributes belongs to Candidate key is called Prime attribute Set (OS) Key attribute set.

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

* Otherwise Non-Prime attribute Set (OS) 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

⇒ At most one primary key for each relation

Let Sid ! Primary key

Alternative keys

⇒ More than one alternative keys allowed for each relation then { PPno, Uno, (DOB, Fname) }
All

Candidate key

(Primary key) (Alternative key)

Primary key

Alternative key

→ Primary key not allowed Null Value.

→ Allowed Null Value

Null: Unknown (or)
unexisted Value.

Index Srl	Sid ↑ (integer)	Sname ↑ (integer)	PPho	Lno	Dob	Ru
	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	

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 goes for 3rd constraints i.e. to access data.

Numerical values gain more priority to design Primary key.

t. 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 notnull is same as primary key.

Table Creation Command-

CREATE TABLE Student

Sid Varchar(10) Primary key,

Sname Varchar(30) NOT NULL,

PPho 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 differentiate tuples of the relation.

* Difference between superkey & candidate key is that ^{candidate key} should not be minimum.

Example:- Sid

Sid Sname

PPho

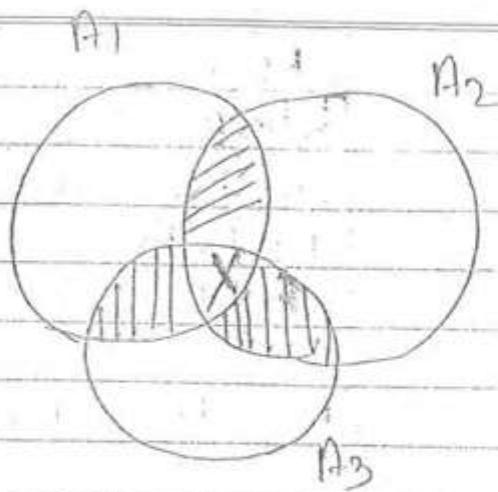
PPho Lno

Superkeys

(Sname DOB) Not Super keys Because it is not uniquely determine the tuples of Relation.

* Every candidate key is ~~also~~ Super key. converse is not true.

e)



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

note

Ques 2) R is the relation set $R(A B C D E)$ & Candidate key $\{A, BC\}$. Then how many super key are possible.

Sol" 1— Because of $A \rightarrow 2^{5-1} = 2^4 = 16$
 " " $BC \rightarrow 2^{5-2} = 2^3 = 8$
 Common $\rightarrow 2^{5-3} = 2^2 = 4$

$$\text{Total : } 16 + 8 - 4 \\ = 20$$

Ques 3) Relational schema with n attributes $(A_1, A_2, A_3, \dots, A_n)$

then How many super key are possible??

L

* Referencing Relation i.e (Enrolled)

1) Insertion - (May cause Referential 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
20/5

(Ques)

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.

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

Solⁿ - (5,2), (7,2), (9,5) are deleted.

Pg 30)

(Ques) geg! greater than or equal to

$(x,y) \in \text{geg}$ only if $y \geq x$

CREATE geg

Ulb integer not null,

Ub integer not null,

primary key Ulb,

Foreign key (Ub). References geg, on delete cascade.

Which of the following is possible if a tuple (y,z) 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 subtract 1 bcoz
∅ is not a super
key)

FOREIGN KEY

(All student Info) Foreign key (Students Enrolled Course)		
(deletion prohibited)	Student (Referenced relation)	Enrolled (Referencing relation)
S_{id}	Student	Enrolled
S_1	Shame	St_{id}
S_2	John	C_{id}
S_3	John	Fee
S_4	John	S_1
		C_1
		C_2
		C_3
		C_4

(S_{id} ! Primary key) (St_{id} ! Primary 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 STUDENT,
 foreign key(Cid) REFERENCES COURSE)
```

Example for foreign key reference the same table:-

Foreign key	
E_id	Sup_id
E1	Null
E2	E1
E3	E2
E4	E3

Here Sup_id reference the E_id means supervisor must belong to E_id i.e. no ~~other~~^{one} member not belongs to E_id is in Sup_id i.e. every member of Sup_id must belongs to E_id.

* Referential key is also called foreign key.

Referential Integrity Constraints:-

* Referential 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 ~~DELETE~~ NO ACTION (default)

(deletion from Referenced Relation is restricted if Referential integrity violation occurs.)

b) ON DELETE CASCADE

(deletion happens from both tables).

* ON DELETE CASCADE never prohibits deletion

If foreign key is on the same table then

Eid	Sup-id
E1	Null
E2	G1
E3	G2
E4	G3

If we delete E1
using ON
DELETE
CASCADE

it will delete
more of the
information from
the table.

c) ON DELETE SET NULL

try to set Null Value in the Referencing Relation foreign key attribute if success then allowed to delete from Referenced Relation (Otherwise not allowed).

* Before delete the tuple check in the referencing relation that putting null value in that relation is allowed or not. If it is allowed then after delete the tuples from Referenced Relation otherwise not possible. **Foreignkey**

Example:-

		ON DELETE SETNULL
<u>Cid</u>	<u>Sup_id</u>	
G1	NULL	
G2	G1NULL	
G3	G2NULL	
G5	G1NULL	
G4	G3NULL	

* If Foreign Key is not null attribute or primary key then ON DELETE SETNULL 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 updatation
- b) ON UPDATE CASCADE
- c) ON UPDATE SET NULL & less used updatation.

- c) a tuple (z, w) with $w < z$ is deleted.
- d) Deletion of (x, y) is prohibited.

Solⁿ :-

(HED)

<u>Lb</u>	<u>Ub</u>
1	2
2	3
3	4
4	4

$$(x, y) \quad (z, w)$$

$$y \\ (4, 4) \\ (3, 4) \\ (2, 3) \\ (1, 2)$$

then possibility is that $w <$

- * In S/W design
 - Requirements
 - Design "UML" Model i.e. Diagrammatic Representation of Entire S/W flow.

ER Model - (Entity Relationship Model)

- * High Level DB Design..
- * It is Diagrammatic Representation of Entire DB Design.

Entity -

An object which can be differentiated 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

S₁

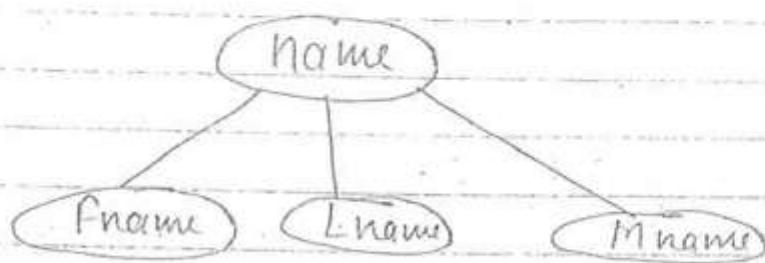
S₂

A
C/C++
C++/Java

(Name)

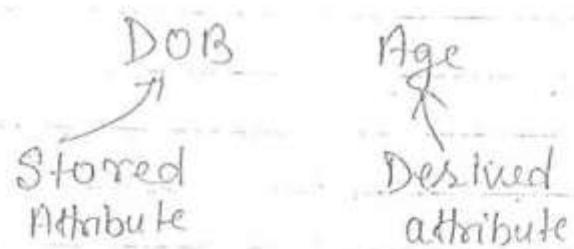
(Name)

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



Derived Attribute-

Value of derived attribute derived from other stored attribute.



$$\text{Age} = \text{Current Date} - \text{DOB}$$

It is represented as:

- * → One
- Many
- ==> Total participation

19/08/2011

Relationship Set:-

Used to relate two or more entity sets.

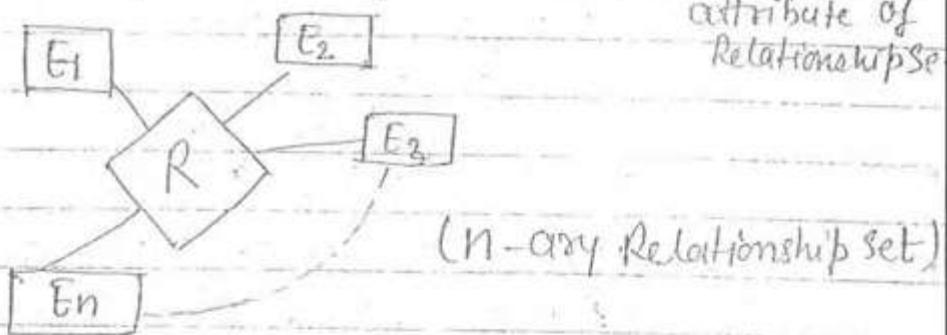
Symbol:-



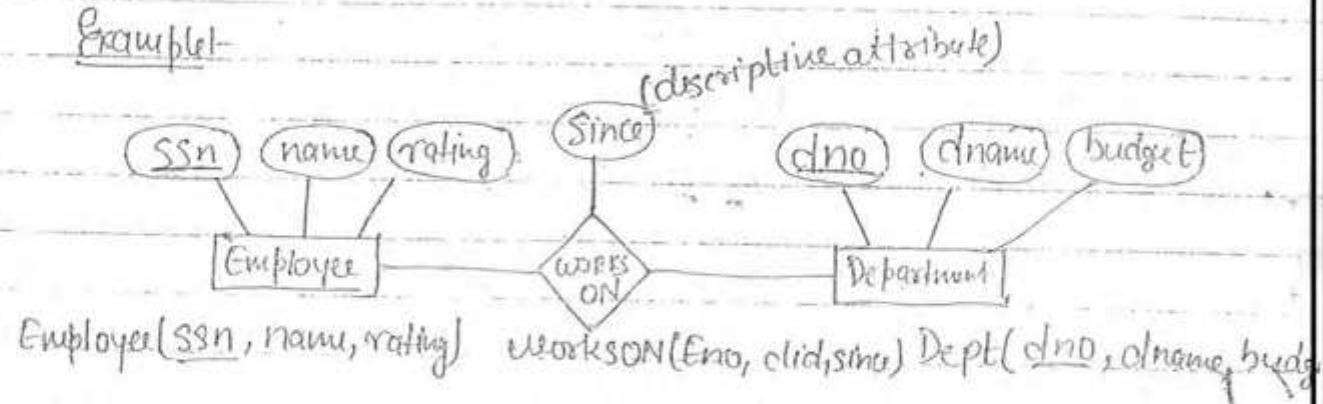
represents the Relationship

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

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



Example:-

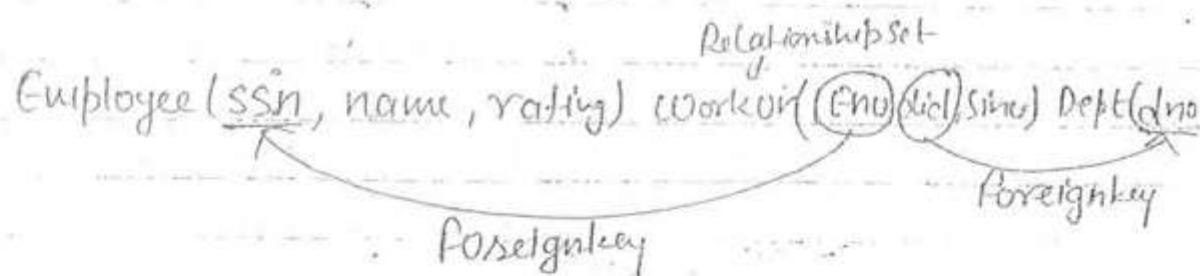


Constraints of Entity Set

- 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),
dno varchar(15),
sdate date,

primary key (_____))

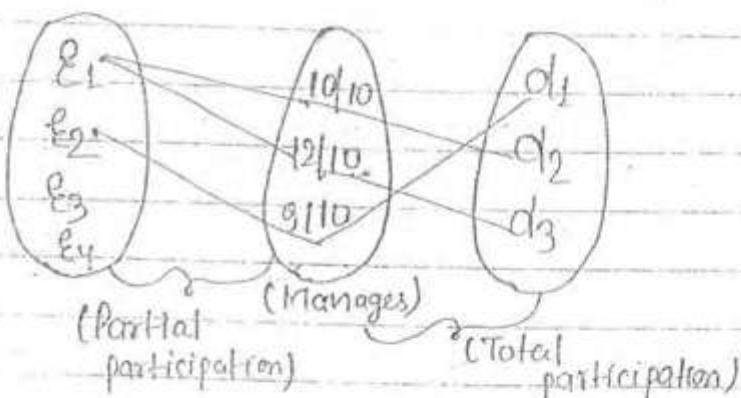
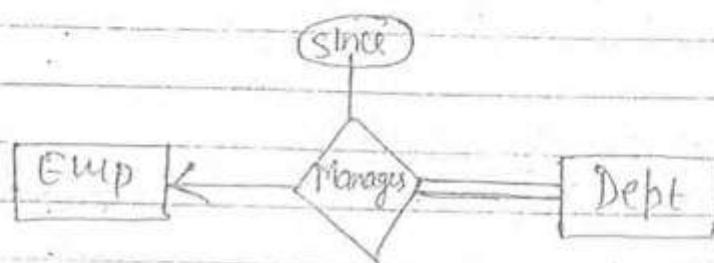
FOREIGN KEY (ENO) Reference Employee ON DELETE CASCADE

FOREIGN KEY (DNO) 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 related with relationship set is called Total Participation.

Foreign key does not depends on cardinality only primary key does only depends on cardinality.

2) Cardinality:-

How one Entity Set Entities relate with Other Entity set.

These are four types Cardinality! -

a) One to One:-

We do not allow Eno & did as dupl i.e. both the entity should be key for One to One cardinality.

Eno	did
e ₁	d ₁
e ₂	d ₃

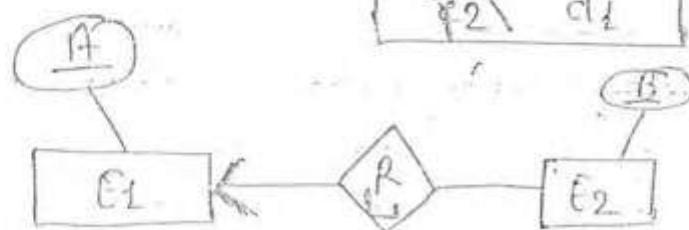


Candidate keys:
A, B

b) One to Many:-

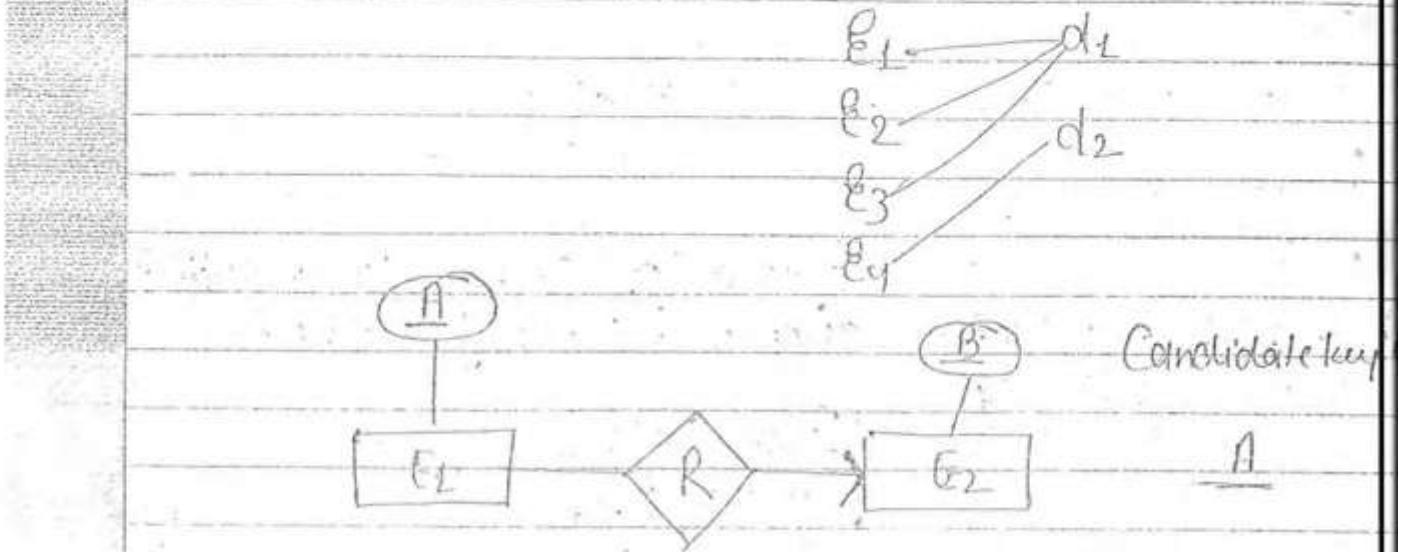
In one to many cardinality one should be primary key.

Eno.	did
e ₁	d ₂
e ₂	d ₃
e ₂	d ₁



Candidate key of R
B

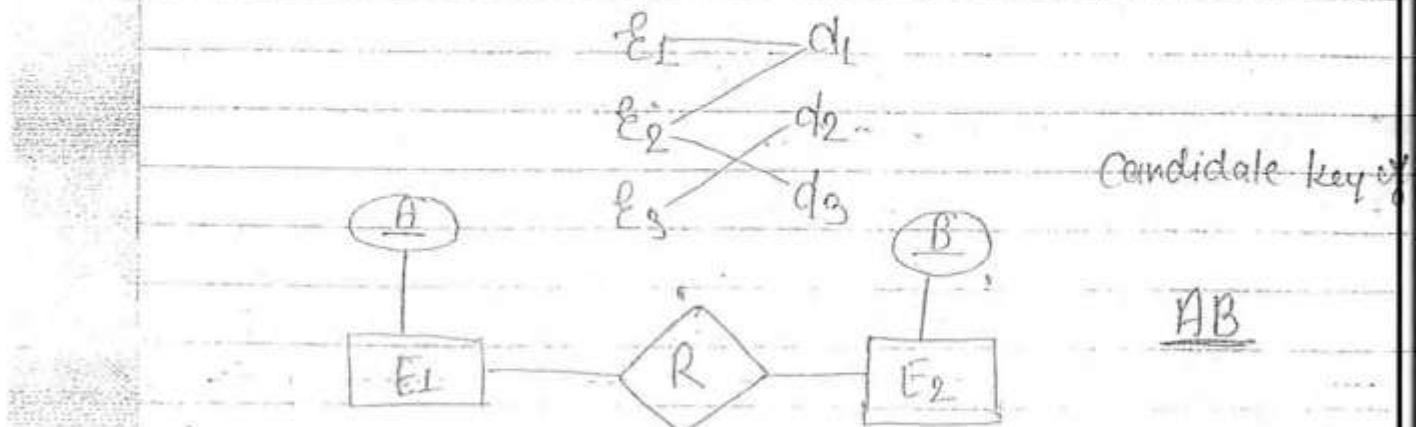
c) Many to One - In Many to One Eno should Primary key.



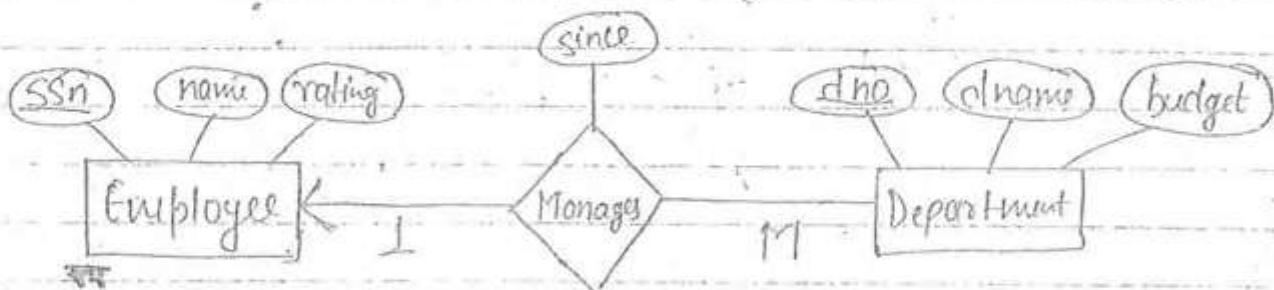
d) Many to Many (M:M & N:N)

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

Here in both Eno & d duplication is possible.



Example:-



Employee(SSN, name, rating)

Manages(SSN, dno, since)

Dept(dno, dname, budget)

Primary key : dno

foreign key : SSN
dno

After merging the managers table & Department Table:

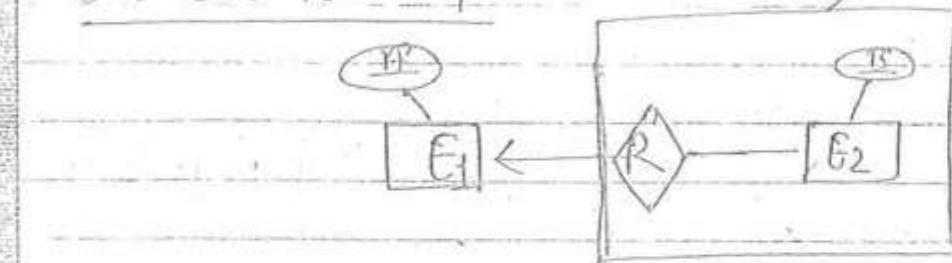
DeptManages(dno, dname, budget, SSN, Siha)

foreign key (SSN) Reference Emp.

Minimization of ER Model:-

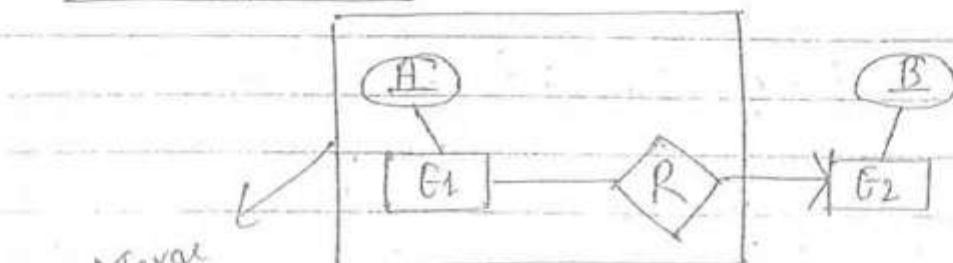
If foreign key attribute set of Relationship set R is also primary key of R and references Entity set E. Then R & E can be Merge into Single Table.

In One to Many -



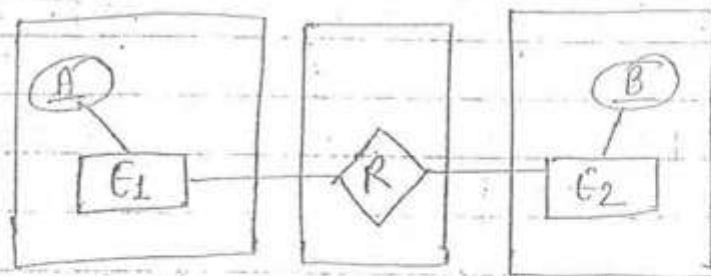
Merge these table

In Many to One -



Merge
these
tables

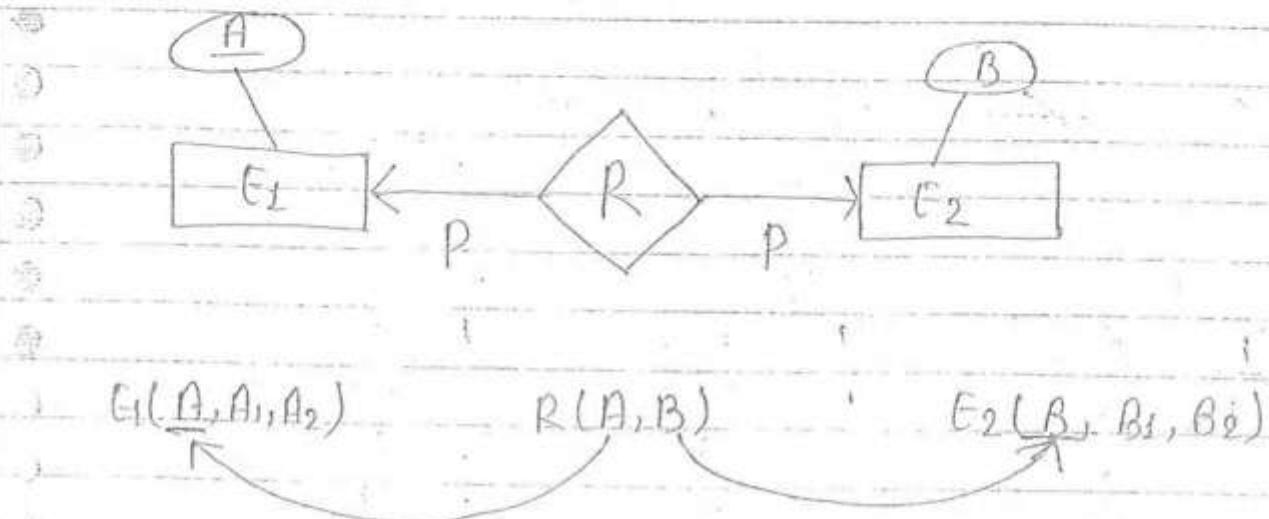
Many to Many -



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

* Participation does not matter while minimizing the One to many, many to one or many to many but it does matter while minimizing one to one.

One to One :-



A	A ₁	A ₂		A	B		B	B ₁	B ₂
1	a ₁	a ₂		1	b ₃		11	b ₁	b ₁
2	a ₁	a ₂	Partial	2	b ₂	Partial	12	b ₁	b ₂
3	a ₃	a ₄					13	b ₁	b ₂
4	a ₄	a ₄							

$(G_1 R)$

A	A ₁	A ₂	B
1	-	-	13
2	-	-	12
3	-	-	Null
4	-	-	Null

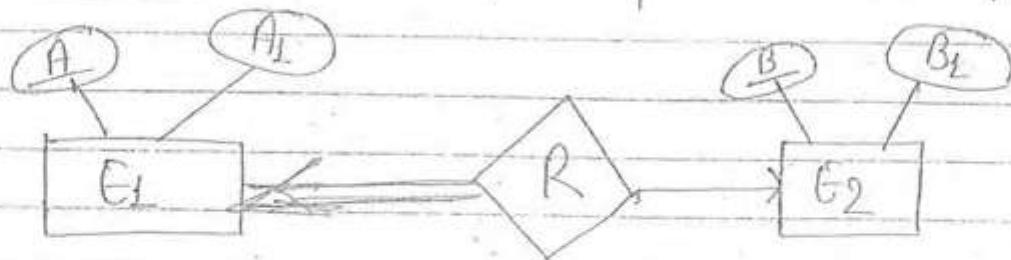
$R G_2$

B	A	B ₁	B ₂
11	Null	-	-
12	2	-	-
13	1	-	-

A: Primary key B: alternative key

* If partial participation exists then we can minimize it either in $(G_1 R)$ or $(R G_2)$.

Assume- Total participation exists b/w G & R.



E ₁		R		E ₂	
A	A ₁	A	B	B	B ₁
1	a ₁	3	13	11	b ₁
2	a	2	11	12	b ₁
3	a ₂	1	12	13	b ₂
Null	Null	14	.	14	b ₂

A	A ₁	R	B ₁
1	a ₁	12	b ₁
2	a ₁	11	b ₁
3	a ₂	13	b ₂
Null	Null	14	b ₂

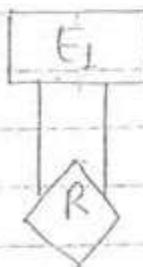
B: Primary key
A: Alternative key

* 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.

*

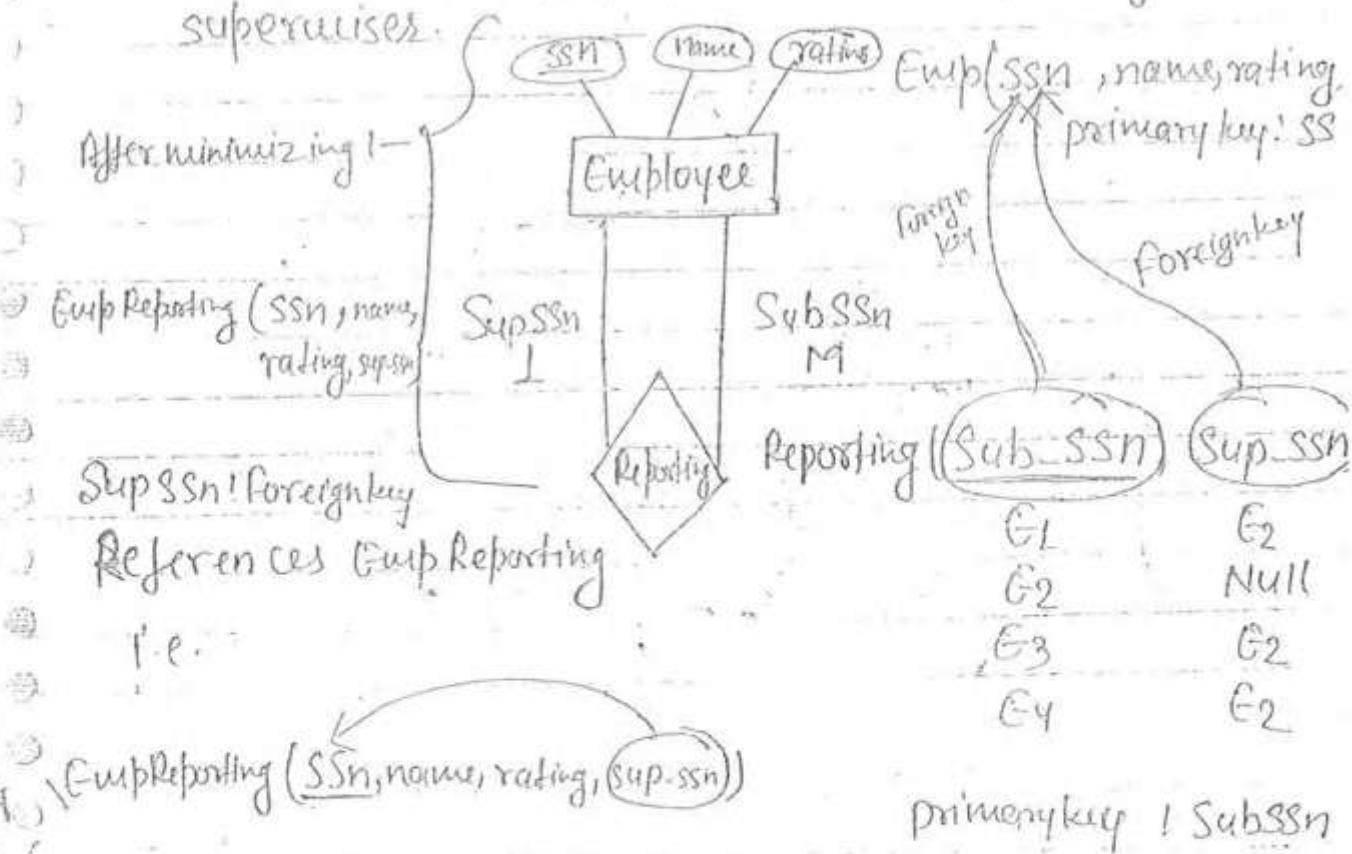
Self Referential Relationship Set:-

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



Eg:- An employee can supervise multiple subordinates & each sub-ordinate reporting to single supervisor.

After normalizing:-



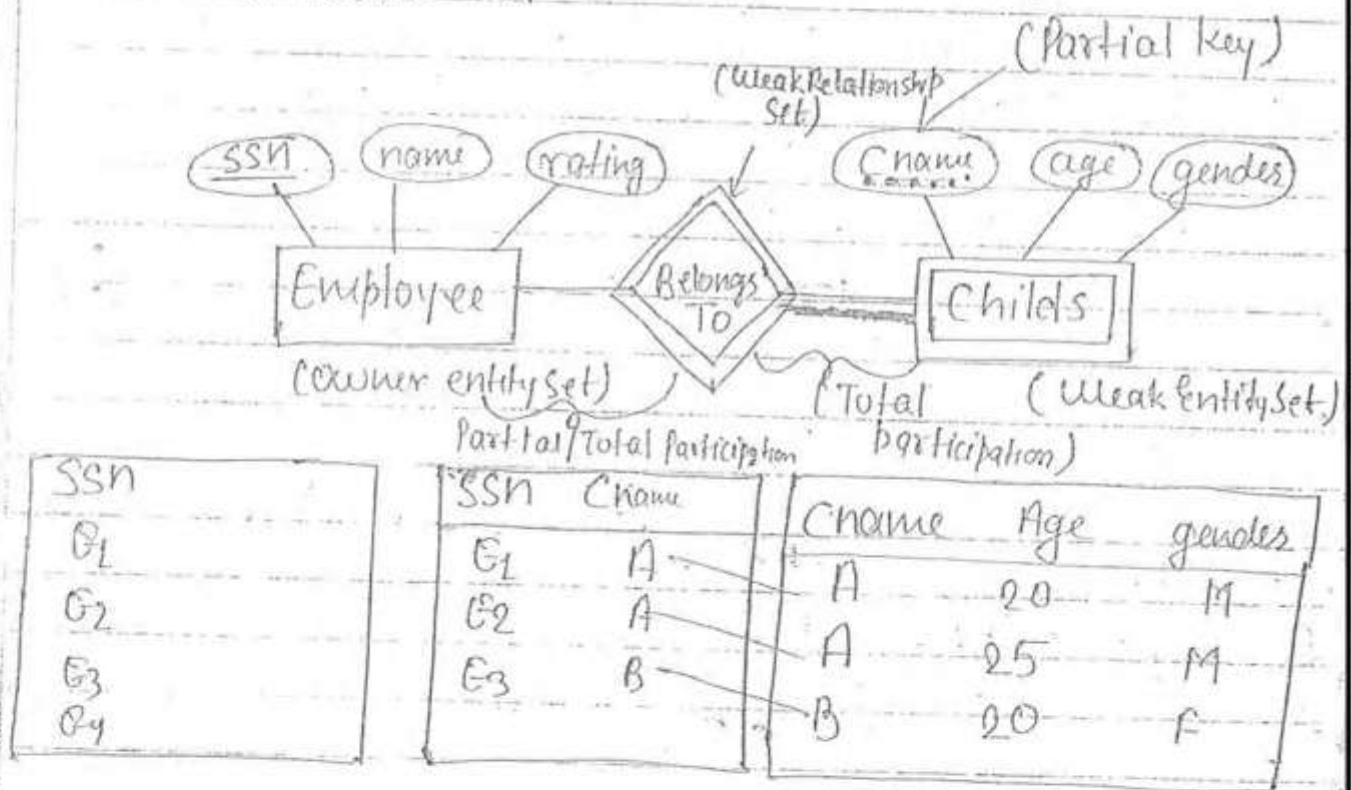
Weak Entity Set -

- 1) Entity set with no proper candidate key.

If is represented by -

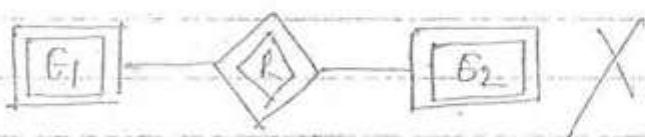


- 2) Tuples in the weak entity set may not possible to differentiate using their own attribute set.



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

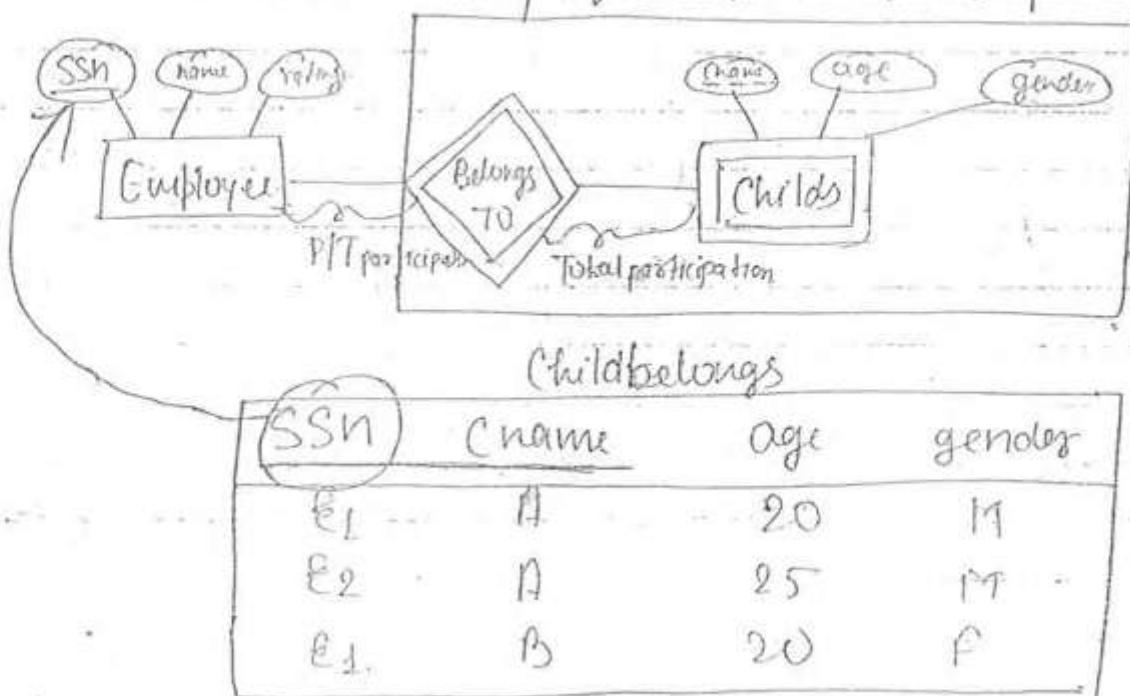
- 4) Two weak entity set not possible to relate with relationship set.



- 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 combined with weak entity & weak relationship set.



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

* Max no. of Tables or min. no. of tables \Rightarrow ?

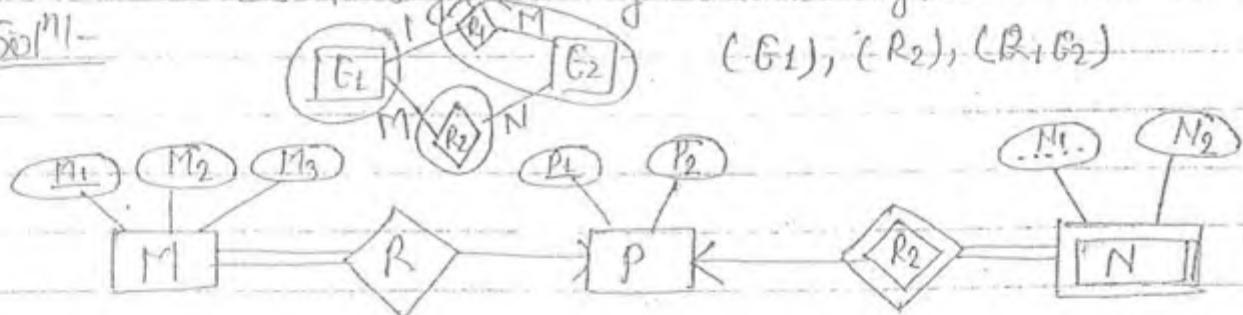
Ques

E₁ & E₂ are Entity Sets

R₁ & R₂ Relationship sets

Relating b/w E₁ & E₂ with 1:M & M:N cardinality
how many minimum no. of tables required to minimize the following ER diagram.

Soln:-



1) minimum no. of tables needed to represent M, N, P,
R₁, R₂

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 Anomaly-

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

Example- Because of deletion of S₁ we lose course C₃ details.

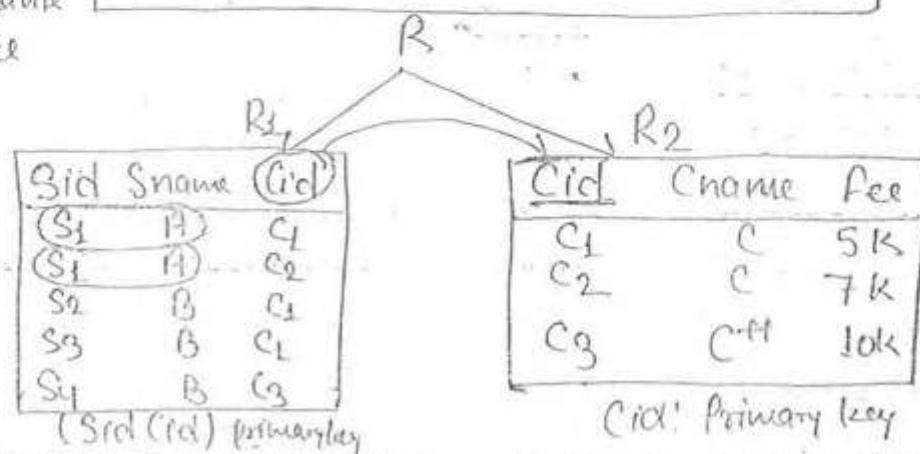
⇒ The only way to elimination of Redundancy is decomposition of the Relation.

Decomposition of the Relation-

Splitting the Relation into two or more sub-relation.

C_{id} → Cname
S_{id} → Sname
C_{id} → Fee
Also
S_{id}C_{id} → Sname
→ Cname
→ Fee

R	Sid	Sname	C _{id}	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 ^H	10K



* To put relationship between two decomposition 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 decomposable 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 relation R.

$$R_1 \bowtie R_2 \equiv R \text{ (lossless Join)}$$

$$R_1 \bowtie R_2 \supseteq R \text{ (lossy Join)}$$

R	$\boxed{\begin{array}{c} S_1 C_1 \\ S_1 C_2 \end{array}}$	$R_1 \bowtie R_2$	$\boxed{\begin{array}{c} S_1 C_1 \\ S_1 C_2 \\ S_1 C_3 \end{array}}$
-----	---	-------------------	--

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

2) Dependency Preserving Decomposition -

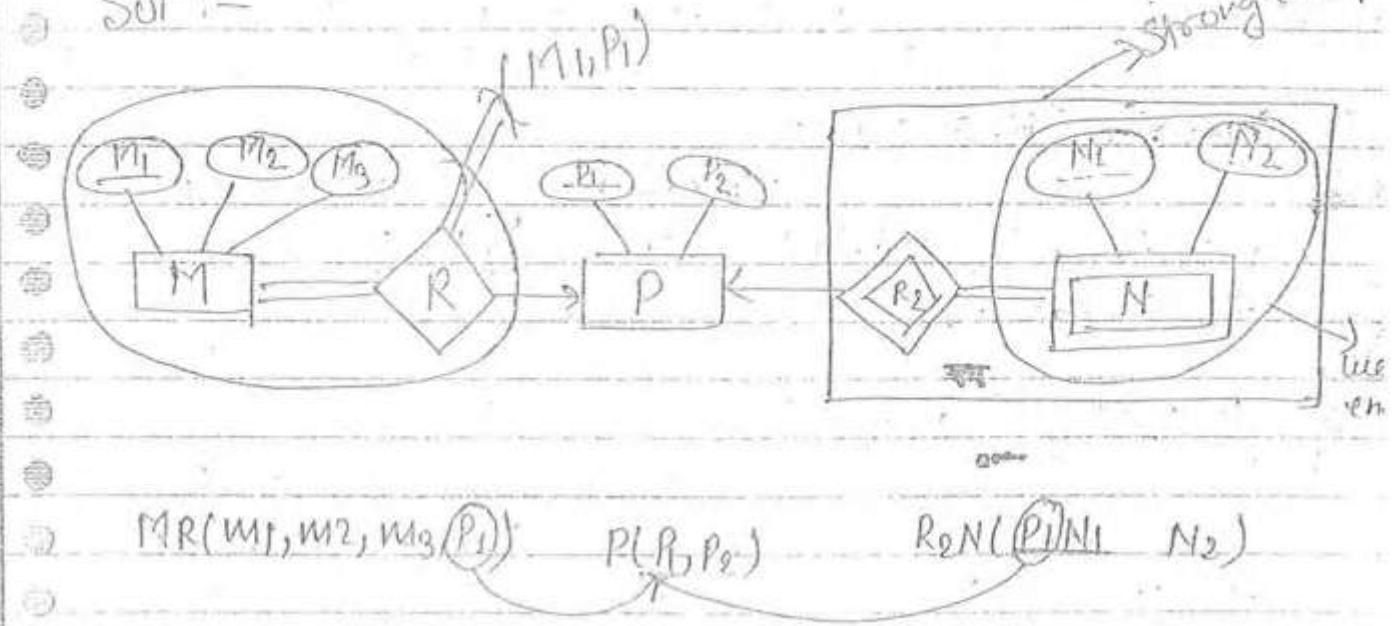
Because relation of decomposition, you should not lose any dependency.

Example -

$$\begin{aligned} Cid &\rightarrow Cname \\ Srd &\rightarrow Sname \\ Cid &\rightarrow fee \end{aligned}$$

All factors of the dependency of relation also easily offer decomposition into two tables.

Soln:-



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	Cname	Fee
S1	A	G	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 bcoz of Redundancy I - (Anomalies)

1) Update Anomaly :-

Update of single value
results all duplicate copies update which
is too costly.

2) Insertion Anomaly :-

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

If we want to
insert (C4, 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

Sid	Sname	Cid	Cname	Fee
S1	A	C1	C	5K
S1	A	C2	C	7K
S2	B	C1	C	5K
S3	B	C1	C	5K
S4	B	C3	C4	10K
XX	XX	C4	Java	15K

Dummy
data

primary key (Sid Cid)

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 $Cid \rightarrow Cname$
and if $t_1.Cid = t_2.Cid$ C
 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 not 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-

I) 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

$\left. \begin{array}{l} Sid \rightarrow Sid \\ Sid.Sname \rightarrow Sid \\ Sid.Sname \rightarrow Sid.Sname \\ Sid.Sname \rightarrow Sname \end{array} \right\}$

* Trivial dependency are always valid because it exist according to the definition we check only at is subset of any set then always trivial dependency occurs. Do not check for the data of the table.

2) Non Trivial Functional dependency - ($X \rightarrow Y$ where $X \cap Y = \emptyset$)

$X \rightarrow Y$ is non-trivial only if

- a) $X \cap Y = \emptyset$
- and b) $X \rightarrow Y$ should satisfy functional dependency definition.

Example-

Cid \rightarrow Cname

Sid \rightarrow Sname

Sid.Cid \rightarrow Sname

} Non trivial functional dependency

3) Semi - non trivial functional dependency -

Combination of trivial & non trivial funⁿ dependency is called semi - non trivial funⁿ dependency.

Example-

Sid \rightarrow Sid.Sname

\rightarrow Sid \rightarrow Sid (Trivial)

\rightarrow Sid \rightarrow Sname (Non trivial)

Properties of funⁿ dependency -

\Rightarrow Reflexivity -

If $X \supseteq Y$ Then $X \rightarrow Y$ is reflexive PD

* Trivial funⁿ dependency are reflexive funⁿ dependency.

\Rightarrow Transitivity-

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

\Rightarrow Augmentation-

If $X \rightarrow Y$ Then $XZ \rightarrow YZ$

Example- If $Sid \rightarrow Sname$

then $Sid Cid \rightarrow Sidname Cid$

\Rightarrow Union of Functional Dependency-

If $X \rightarrow Y$ & $X \rightarrow Z$

Then $X \rightarrow YZ$

but converse is not true i.e.

If $A \rightarrow C$
 $B \rightarrow C$
then $AB \rightarrow C$

not allowed

\Rightarrow Splitting of Functional Dependency-

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

* Commutative properties not hold by functional dependency
 $X \rightarrow Y$ Then $Y \rightarrow X$ is not possible

* Associative properties is also not hold by functional dependency
i.e. $A \rightarrow BC \neq AB \rightarrow C$

Application of Functional dependency

- 1) Identify the Redundancy in the Relation
- 2) Identify the candidate key of the Relation.

* $X \rightarrow Y$ non trivial dependency with X is not candidate key, attribute set (X, Y) suffering from Redundancy.

Example:- If $R(ABC)$ & if A ! Candidatekey

A	B	C	$B \rightarrow C$
1	2	a	
2	2	a	
3	2	a	
4	3	b	
5	3	b	

Ques)

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

$$\begin{array}{llll}
 \checkmark A \rightarrow B & \checkmark B \rightarrow A & \checkmark C \rightarrow A & XAB \rightarrow C \\
 XA \rightarrow C & X B \rightarrow C & \checkmark C \rightarrow B & \checkmark BC \rightarrow A \\
 \checkmark A \rightarrow BC & X B \rightarrow AC & \checkmark C \rightarrow AB & \checkmark AC \rightarrow B \\
 \cancel{AB} \rightarrow C & \cancel{BC} \rightarrow A & & \\
 \cancel{AC} \rightarrow B & & &
 \end{array}$$

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

Attribute closure (X^+) -

Set of attribute functionally determined by 'x':

Ex:- $R(ABCD)$

$\{A \rightarrow B, B \rightarrow C, C \rightarrow D\}$

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

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

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

$$D^+ = \{D\}$$

Note
2006

Ques $\{AB \rightarrow CD, AF \rightarrow D, DE \rightarrow F, C \rightarrow G, F \rightarrow E, h \rightarrow A\}$
Option is false -

$$\begin{array}{ll} \text{(i)} \quad (EF)^+ = \{A(CDEFH)\} & CF^+ = \{CFHED\} \\ \text{(ii)} \quad (BG)^+ = \{ABCDHG\} & BH^+ = \{BHGACD\} \\ \text{(iii)} \quad (AF)^+ = \{ACDEFH\} & AF^+ = \{AFFED\} \\ \text{(iv)} \quad (AB)^+ = \{ACDFH\} & AB^+ = \{ABCDAH\} \end{array}$$

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 superkey may be a candidate key or may not be candidate key.

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

AB ! Super key.

if $\begin{cases} SA^+ = \text{Not determines all attributes of } R \\ SB^+ = \text{Not determines all attributes of } R \end{cases}$

Then AB ! minimal Super key (Candidate key)

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

$$\{A \rightarrow B, B \rightarrow C, C \rightarrow D\}$$

Identify Candidate key

Solⁿ1- $(A)^+ = \{A B C D\}$

$\Rightarrow A$! Superkey $\{A\}$: Candidate key

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

(Ques) $R(A B C D E)$

$$\{AB \rightarrow C, C \rightarrow D, B \rightarrow E\}$$

Solⁿ1- $(AB)^+ = \{A B C D E\}$

$\Rightarrow AB$! Super key

Check for minimal superkey -

$$A^+ = \{A\}$$
 not superkey

$$B^+ = \{B E\}$$
 not superkey

$\Rightarrow AB$! Candidate key

(Ques) $R(A B C D E)$

$$\{AB \rightarrow C, C \rightarrow D, B \rightarrow EA\}$$

Solⁿ1- $AB^+ = \{A B C D E\}$

AB ! Super key

Check for minimal superkey

$$A^+ = \{A\}$$

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

$\Rightarrow AB$ Superkey but not candidate key
 $\{B\}$! Candidate key

Ques) R(ABCDEF)
 $\{AB \rightarrow C, C \rightarrow D\}$

Solⁿ1- $(AB)^+ = ABCD$

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

$$(ABCE)^+ = ABCDFG$$

$(ABCE)$: Candidate Key

Ques) R(ABCDEF)

$$\{A \rightarrow BCDEF, BC \rightarrow ADEF, B \rightarrow F, D \rightarrow E\}$$

Solⁿ1-

$$A^+ = \{ABCDEF\}$$

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

B⁺ = BF X not superkey

C⁺ = C X not superkey

Candidate key = {A, BC}

* if $X \rightarrow A$ (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.

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

$\{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A\}$

Soln -

$$\begin{aligned}A^+ &= ABCD \\B^+ &= ABCD \\C^+ &= ABCD \\D^+ &= ABCD\end{aligned}$$

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

(Ques) $R(A B C D E F)$

$\{AB \rightarrow C, C \rightarrow D, D \rightarrow E, E \rightarrow F, F \rightarrow B\}$

$$\begin{aligned}(AB)^+ &= \{ABCDEF\} \quad AB \text{ is super key} \\A^+ &= \{A\} \times \\B^+ &= \{B\} \times \\(AF)^+ &= \{ACDEFB\} \quad \text{not super key} \\A^+ &= \{A\} \times \\F^+ &= \{FB\} \times\end{aligned}$$

Candidate Key = $\{AB, AF, AE, AD, AC\}$

$$\begin{aligned}(AE)^+ &= \{AEFBCD\} \\E^+ &= EFB \times \\(AD)^+ &= \{ABCDEF\} \\D^+ &= DEFBA \times \\(AC)^+ &= \{ABCDEF\} \quad c^+ = CDEFB \times\end{aligned}$$

Ques) $R(ABCDEFHJI)$

$$\begin{cases} AB \rightarrow C \\ A \rightarrow DE \\ B \rightarrow F \\ F \rightarrow GH \\ D \rightarrow IT \end{cases}$$

$$(AB)^+ = \{ABCDEFHJI\}$$

$$A^+ = \{ADEIT\}$$

$$B^+ = \{BFHGI\}$$

$$\text{Candidate Key} = \{AB\}$$

Ques) $R(ABDLPT)$

$$\begin{cases} B \rightarrow PT, T \rightarrow L, A \rightarrow D \end{cases}$$
$$B^+ = \{BPTL\}$$

$$(AB)^+ = (BPT \cup LAD)$$

$$A^+ = \{AD\}$$

$$B^+ = \{BPTL\}$$

$$\text{Candidate Key} = \{AB\}$$

Ques) $R(ABCDEFHJ)$

$$\begin{cases} AB \rightarrow C, AC \rightarrow B, AD \rightarrow E, B \rightarrow D, BC \rightarrow A, E \rightarrow G \end{cases}$$

$$(ABDF) = \{ A-B-C-D-E-G-F-H \}$$

$$(B \cap F \cap H)^c = \{A, B, C, D, E, G\}$$

$$(ACM)^t = \{A B C D E F G H\}$$

Candidate Keys = {ABFH, BCFH, ACFH}

(Qw) R(ABCDE)

$\{ A \rightarrow BC, D \rightarrow E, B \rightarrow D, E \rightarrow AG \}$

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

$$(E)^+ = \{A, B, C, D, E\}$$

$$CDH = \{A, B, C, D, E\}$$

$$(BC)^T = \{ A B C D E \}^T$$

$$(A \cap B) = \underline{\hspace{2cm}}$$

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

20/08/11

Ques) $R(ABCDEF)$

$\{AB \rightarrow C, C \rightarrow DE, E \rightarrow F, F \rightarrow A\}$

Ques) $R(ABCDEF)$

$\{AB \rightarrow C, C \rightarrow D, D \rightarrow E, E \rightarrow A, D \rightarrow B\}$

Solⁿ1)

$$AB^+ = \{ABCDEF\}$$

$$A^+ = \{A\}$$

$$B^+ = \{B\}$$

$$FB^+ = \{ABCDEF\}$$

$$F^+ = \{AF\}$$

$$EB^+ = \{ABCDEF\}$$

$$E^+ = \{EA\}$$

$$CB^+ = \{ABCDEF\}$$

$$C^+ = \{ACDEF\}$$

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

Solⁿ2)

$$ABF^+ = \{ABCDEF\}$$

$$(ABD)^+ = ABCDEF$$

$$XADF^+ = \{ABCDEF\}$$

$$(EBF)^+ = ABCDEF$$

$$XEDF^+ = \{ABCDEF\}$$

$$E^+ = EA$$

$$CF^+ = \{ABCDEF\}$$

$$B^+ = B$$

Candidate Key = {ABF, BCF, DA, CF}

(Ques)

$R(ABCDE)$

$\{AB \rightarrow C, C \rightarrow D, D \rightarrow E, E \rightarrow A, D \rightarrow B\}$

$$AB^+ = \{ABCDE\} \quad AB^f = \{ABCDEF\}$$

$$A^+ = \{AB\}$$

$$B^+ = \{B\}$$

$$XAD^+ = \{ABCDEF\}$$

$$EB^+ = \{ABCDEF\}$$

$$E^+ = \{E\}$$

$$D^+ = \{BAC\}$$

$$XDB^+ = \{ABCDEF\}$$

$$\checkmark D^+ = \{DEABC\}$$

$$C^+ = \{CDEABC\}$$

* Candidate key:- $\{AB, EB, D^+, C^+\}$
 If $R(ABC)$ with NO non trivial functional dependency
 $\Rightarrow ABC$! Candidate Key

* If $R(ABCD)$

$$\begin{array}{l} \{AB \rightarrow C, DE \not\rightarrow F, C \rightarrow D \not\rightarrow F, E \not\rightarrow F, D \not\rightarrow F\} \\ \downarrow \\ C \rightarrow D, C \rightarrow E \end{array}$$

then remaining Relation

$$\{AB \rightarrow C, C \rightarrow D\}$$

\Rightarrow In the above we remove those functional dependency whose attribute not belongs into the relation schema attribute
 i.e. $\{DE \rightarrow F, C \rightarrow E, E \rightarrow F, D \rightarrow C\}$

L

Properties of functional dependency set -

1) Membership test -

Example - If $\{AB \rightarrow C, C \rightarrow D\}$

check $B \rightarrow C$ is member of this set

check $AB \rightarrow D$

Soln -

$$B^+ = B$$

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

Check for $AB \rightarrow D$ is membership set

$$AB^+ = ABCD$$

i.e. AB determine D so, it is member
functional dependency of the set.

→ Let F be the functional dependency
and $X \rightarrow Y$ and F.D.

$X \rightarrow Y$ logically (F^+) in P 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:- $F = \{ A \rightarrow B, B \rightarrow C \}$

$$F^+ = \left\{ \begin{array}{l} A \rightarrow A \quad A \rightarrow B \quad B \rightarrow C \quad A \rightarrow C \\ B \rightarrow B \quad AC \rightarrow BC \quad AB \rightarrow AC \quad AB \rightarrow CB \\ AB \rightarrow A \quad AC \rightarrow B \quad | \quad | \\ AB \rightarrow B \quad AC \rightarrow C \quad | \quad | \\ AB \rightarrow AB \quad | \quad | \end{array} \right\}$$

F^+ contains

① $\phi^+ = \phi$

$$A^+ = ABC$$

② $B^+ = BC$

$$C^+ = C$$

③ $\begin{cases} (AB)^+ = ABC \\ (BC)^+ = BC \\ (AC)^+ = ABC \end{cases}$

④ $\{(ABC)^+ = ABC\}$

$\emptyset \rightarrow \emptyset$	①
$A \rightarrow \emptyset \quad A \rightarrow B \quad A \rightarrow AB \quad A \rightarrow BC$	⑧
$A \rightarrow A \quad A \rightarrow C \quad A \rightarrow AC \quad A \rightarrow ABC$	⑧
$B \rightarrow \emptyset \quad B \rightarrow B \quad B \rightarrow C \quad B \rightarrow BC$	④
$C \rightarrow \emptyset \quad C \rightarrow C$	②
$AB \rightarrow \emptyset$	③
$BC \rightarrow \emptyset$	④
$AC \rightarrow \emptyset$	⑧
$ABC \rightarrow \emptyset$	⑧
	43

Here F^+ logically equivalent to F

$R(AB)$

$F, \{ A \rightarrow B, B \rightarrow A \}$

Identify Total dependencies in F^+

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

$$A^+ \supseteq AB$$

$$\begin{aligned} A \rightarrow \phi & \quad A \rightarrow A \quad A \rightarrow B \\ & \quad A \rightarrow AB \end{aligned}$$

$$B^+ = AB$$

$$\begin{aligned} B \rightarrow \phi & \quad B \rightarrow A \quad B \rightarrow B \\ & \quad B \rightarrow AB \end{aligned}$$

$$AB^+ = AB$$

$$\begin{aligned} AB \rightarrow \phi & \quad AB \rightarrow AB \\ AB \rightarrow A & \quad AB \rightarrow B \end{aligned}$$

13

3) Equality b/w FD sets :-

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

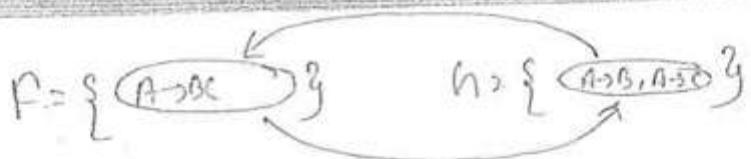
b) F & H said to be Equal only if

[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



$F \text{ covers } h$	$h \text{ covers } F$	
T	T	$\Rightarrow F = h$
T	F	$\Rightarrow F > h$
F	T	$\Rightarrow h > F$
F	F	$\Rightarrow F \neq h$ (No comparison value)

Ques) $F = \{ A \rightarrow C, AC \rightarrow D, E \rightarrow AD, E \rightarrow H \}$
 $h = \{ A \rightarrow CD, E \rightarrow AH \}$

$F \text{ COVER } h$ is true $A^t = ACD, E^t = EAH$
 $h \text{ COVER } F$ is true $AC^t = ACD, A^t = ACD, E^t = EAH$

Ques) $F = \{ AB \rightarrow CD, B \rightarrow C, C \rightarrow D \}$
 $h = \{ AB \rightarrow C, AB \rightarrow D, C \rightarrow D \}$

$F \text{ COVER } h$ is true
 $h \text{ COVER } F$ is false

$\Rightarrow F > h$

Properties of Decomposition -

1) Lossless Join Decomposition -

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

In general $R_1 \bowtie R_2 \bowtie \dots \bowtie R_n \supseteq R$

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

Then lossless join decomposition

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

Then lossy join decomposition

Example - $R(ABC) \rightarrow$ decomposed into $R_1(AB)$ $R_2(BC)$

R			R_1	R_2
A	B	C	A B	B C
1	1	2	1 1	1 2
2	1	1	2 1	1 1
3	2	1	3 2	2 1

$$\pi_{AB}(R)$$

$$\pi_{BC}(R)$$

π : Projection

To Select attribute of Relation

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

\hookrightarrow : Selection operator

Select the tuples those are satisfied predicate condition

SECOND NORMAL FORM:-

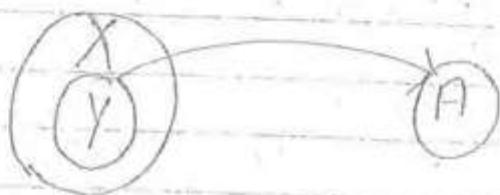
Let R be the Relational Schema. It's in 2NF only if

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

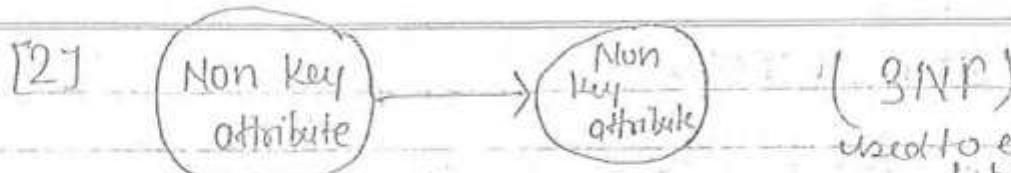
A: Non-Prime
attribute

Non-trivial FD is partial dependency: only if
it satisfies
that the
relation
is in
2NF.

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

need to check the relation is in 2NF. If yes, use the
above condition for 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 redundancy is not possible.

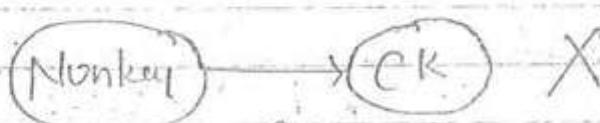


Example:-

$R(ABCD E)$. $\{AB \rightarrow C, C \rightarrow D, B \rightarrow E\}$

[2] case

[1] case



$R(ABCD)$

Example:- $\{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A\}$

Candidate key $\{A, B, C, D\}$.

This dependency
is not possible.

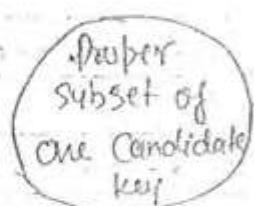


Example:- $R(ABCDE)$

$\{AB \rightarrow C, C \rightarrow D, B \rightarrow E, C \rightarrow A\}$

$C \cdot K \{AB, CB\}$

[3]



proper
subset of
other candidate
key

(BCNF)

used to eliminate this type of redundancy

Example:-

$R(ABCD)$. $\{AB \rightarrow CD, D \rightarrow A\}$

[2] case

Relation consists Redundancy bcoz of single valued dependency:

If $X \rightarrow Y$: Non Trivial functional dependency

with X is not super key. Then redundancy always possible.

Example:-

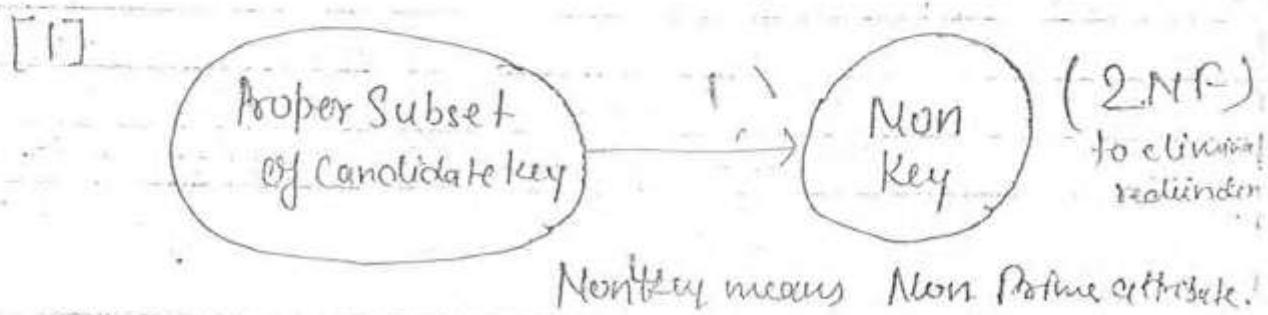
	X	Y
Not Super Key	x_1	y_1
	x_2	y_1
	x_1	y_1

Or in the other 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 bcoz of single value dependencies.)

Non-Trivial FD which results Redundancy :-



By Converting the above relation in 1NF! we get

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

Redundancy

(Sid \rightarrow Sname)

Sid (Name) \rightarrow Sname

(Sid (Name)) : Primary key

No multivalued Attribute (1NF)

\Rightarrow Problem of 1NF!

It Results high redundancy.

Ques) How to convert Relation into 1NF if
 $R(A, B C, D)$

A: Primary key

C, D: Multivalued Attributes

Not in 1NF

Solⁿl -

$R(\underline{A C D}, B)$! (1NF Relation)

primary key = ACD

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

Dependency 1 - $A \rightarrow B$ (Before the 1NF conversion relation)

$ACD \rightarrow B$ (After relation is in 1NF)

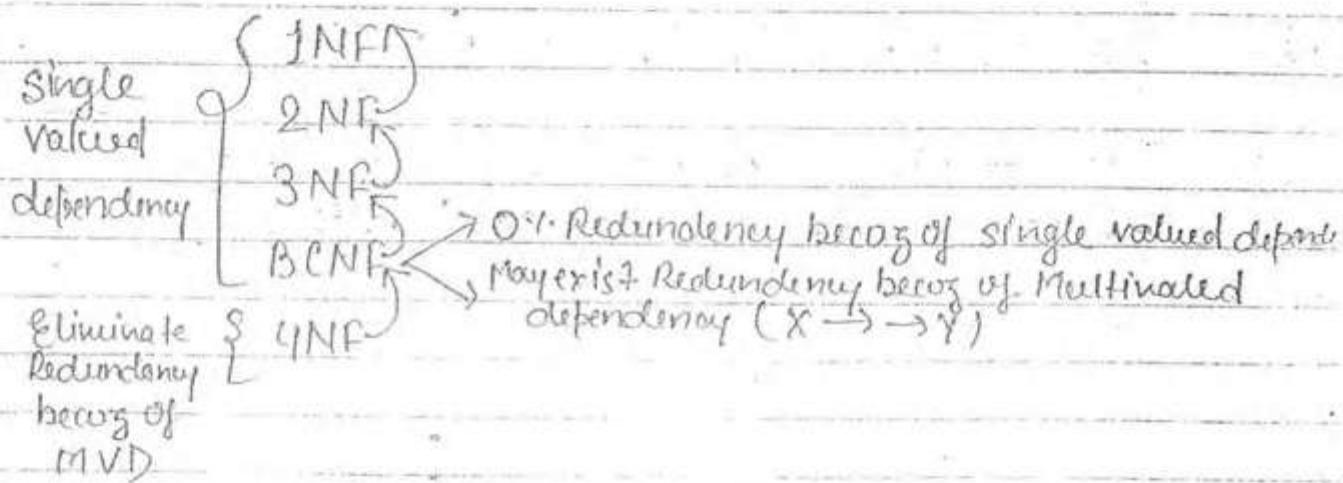
21/08/11

NORMAL FORMS:-

(Used to eliminate/reduce Redundancy)

(Set of Rules to perform decomposition)

Types of Normal Forms:-



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
S1	A	DB/DM
S2	B	DM/DM
S3	A	DB

Multivalued attribute

(NOT in 1NF)

Primary key : Sid

Check for correct decomposition

Given) $R(ABCD \sqsubseteq FGH \sqsubseteq I)$

$\{AB \rightarrow C, A \rightarrow DE, B \rightarrow F, F \rightarrow GH, D \sqsubseteq IJ\}$

$D_1 = \{ABC, ADE, BFH, DJ\}$ {thus
AB \rightarrow C A \rightarrow DE B \rightarrow F F \rightarrow GH D \rightarrow IJ } two

$D_2 = \{ABCDE, BFH, DJ\}$ {dependency preserving
AB \rightarrow C A \rightarrow DE B \rightarrow F F \rightarrow GH D \rightarrow IJ }

$D_3 = \{ABCD, DE, BF, GH, DJ\}$

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

4) Union (U) :- $R_1 \cup R_2 = R_2 \cup R_1$ ✓

5) Intersection (I) :- ✓
 $R_1 \cap R_2$

6) Set Difference (-) :- ✗

7) (Division) :- ✗

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

8) Natural Join (N) :- ✓

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

Supplier		Catalog	
Sid		Sid	Prod
S1		S1	S1
S2		S1	S2
S3		S1	S3
1		1	1
S ₂₀₀		S ₁	S ₂₀₀

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

Min no. of tuples in sup \bowtie catalog = ... (because
(below Sid is foreign key of supplier Sid S., Sid refers to
minimum 100 tuples is necessary.)

(because
minimum 100 tuples is necessary.)

[Q4]

$$\pi_{\text{row}} \left(\sigma_{\text{SId}} \left(\pi_{\text{PId}} \left(\sigma_{\text{col=Ref}} \left(\pi_{\text{Catalog}} \right) \right) \right) \right) \bowtie \pi_{\text{SId same}} \left(\pi_{\text{Supplies}} \right)$$

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

Check for Commutativity

1) π (Projection) = \times

$$\pi_{A_2}(\pi_{A_1}(R)) \neq \pi_{A_1}(\pi_{A_2}(R))$$

2) σ (Selection) = \checkmark

$$\sigma_{C_2} \left(\sigma_{C_1}(R) \right) = \sigma_{C_1} \left(\sigma_{C_2}(R) \right) = \sigma(R) = \left(\sigma_{C_1}(R) \right) \cap \left(\sigma_{C_2}(R) \right)$$

3) Cross Product = \checkmark

$$R_1 \times R_2 = R_2 \times R_1$$

Ques) $R(A B C D E H)$

$$f = \{ AB \rightarrow C, AC \rightarrow B, AD \rightarrow E, B \rightarrow D, BC \rightarrow A, EH \rightarrow H \}$$

$$D = \{ ABC; ACDE, ADH \}$$

$A B C$	$A C D H$	$A D H$
ABC	ACDH	ADH
ACDE		

$$A^t = A$$

$$B^t = BD$$

$$C^t = C$$

$$AB^t = ABC$$

$$BC^t = ABCDEH$$

$$AC^t = ACBDEH$$

$$A^t = A$$

$$C^t = C$$

$$D^t = D$$

$$AC^t = ABCDH$$

$$AD^t = ADEH$$

$$AH^t = AH$$

$$CD^t = CD$$

$$CH^t = CH$$

$$DH^t = DH$$

$$ACD^t = ACDBEH$$

$$ACH^t = ACBDEH$$

$$CDH^t = ACDBEH$$

$$A^t = A$$

$$D^t = D$$

$$H^t = H$$

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

* All the original functional dependency are preserved in this
it is dependency preserving decomposition

Ques) Check whether decomposition preserving or Not.
 $R(ABCD \in F)$

$$F = \{ AB \rightarrow CD, C \rightarrow D, D \rightarrow E, E \rightarrow F \}$$

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

$R_1(AB)$	$R_2(CDE)$	$R_3(EF)$
—	$C \rightarrow D, C \rightarrow E,$ $C \rightarrow DE,$ $D \rightarrow E, D \rightarrow E,$ $E \rightarrow D$	$E \rightarrow F$ $F \rightarrow E$

Not preserving dependency

$$A^t = A$$

$$C^t = CDEF$$

$$E^t = FE$$

$$B^t = B$$

$$D^t = DEF$$

$$F^t = FE$$

$$B^t = ABCDEF$$

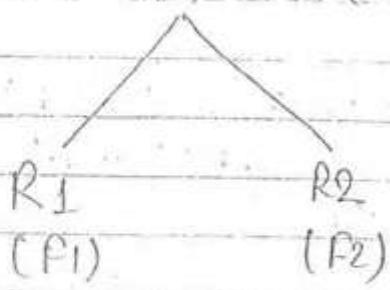
$$E^t = EF$$

$$CD^t = CDEF$$

$$DE^t = DEF$$

$$E^t = ECF$$

$R \leftrightarrow (F)$



which of them is Always false

- a) $F_1 \cup F_2 \supseteq F$
- b) $F_1 \cup F_2 = F$ (Dependency preserving)
- c) $F_1 \cup F_2 \subseteq F$ (Not preserving dependency)
- d) All the above

* If $R_1 \bowtie R_2 \supseteq R$ &
 $F_1 \cup F_2 \subseteq F$ then lossless and preserving
dependency

If $R_1 \bowtie R_2 \subseteq R$ &
 $F_1 \cup F_2 \subseteq F$ then lossy and not preserving
dependency.

Ques) Check whether Dependency are Preserving or not
 $R = (ABCD)$

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

$$D = \{ AB, BC, CD \}$$

U

2) Dependency Preserving Decomposition :-

Let R be the Relational Schema with FD set F decomposed into R_1, R_2, \dots, R_N with FD set F_1, F_2, \dots, F_N respectively

In general $F_1 \cup F_2 \cup \dots \cup F_N \subseteq F$

If $F_1 \cup F_2 \cup \dots \cup F_N = F$

Dependency Preserving

If $F_1 \cup F_2 \cup \dots \cup F_N \subset F$

Not preserving dependency

Ques) $R(ABCD)$

$\{A \rightarrow B, B \rightarrow C, C \rightarrow D\}$

$D = \{AB, BCD\}$

Soln:- Identify FD's of Subrelations

$R_1(AB)$	$R_2(BD)$
$A \rightarrow B$	$B \rightarrow CD$
$(B \rightarrow C, C \rightarrow D)$ $A \rightarrow B$	$C \rightarrow D$ $BC \rightarrow D$ $BD \rightarrow C$

membership test

Appt 1

$A \rightarrow B$

$B \rightarrow C$

$C \rightarrow D$

$$A^f = ABCD$$

$$B^f = BCD$$

$$B^f = BCD$$

$$C^f = CD$$

$$D^f = D$$

$$BC^f = BCD$$

$$CD^f = CD$$

$$BD^f = BCD$$

$F_1 \cup F_2$ covers F Then preserving dependency

a) $R_1(A, B)$ $R_2(B, C)$ $R_3(A, B, D, E)$ $R_4(C, H)$

$AB^t = ABCDCH$

$R_{13}(ABDE)$

$R_2(B, C)$

$R_4(CH)$

$R_2(B, C)$

$R_{134}(ABDEH)$

$C^t = EH$

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

b) $R_1(A, B, C)$ $R_2(A, C, D, E)$ $R_3(A, D, H)$

$AC^t = ACBDEH$

$R_{12}(ABCDE)$

$R_3(ADH)$

$AD^t = ADEH$

is superkey
for R₃ relation

$R_{12}(ABCDEH)$

so it's lossless
because it is decomposed
into single relation

Decomposition is lossless only if N Relations Become Single Relation.

Ques) $R(ABCDEF)$

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

$$D = \{ BE, ACDEF \}$$

2) $R(ABCD\bar{E})$:

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

$$D = \{ ABC, ACDE \}$$

3) $R(ABCD)$

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

$$D = \{ BC, AD \}$$

4) $R(ABCDEF)$

$$F = \{ AB \rightarrow C, AC \rightarrow B, AD \rightarrow E, B \rightarrow D, BC \rightarrow F, E \rightarrow H \}$$

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

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

Solⁿ 1) lossy

$E^+ = \{ E \}$ not superkey so lossy

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

3) $F \cap D = \emptyset$ so lossy

4)

~~lossless~~

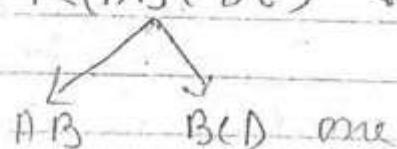
Algorithm for checking decomposition is lossless or not

→ let R be the Relational Schema with FD
Set F decomposed into R_1, R_2, \dots, R_n .
Given Decomposition is lossless only if

$$[1] R_1 \cup R_2 \cup \dots \cup R_n = R$$

(because of decomposition R should not lose any attribute)

i.e. if $R(ABCDEF)$



attribute is lost after decomposition.

and [2] 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

$$\text{I. } R_i \cap R_j \neq \emptyset$$

i.e. $R(ABCDEF)$



i.e. $R_1 \cap R_2 = \emptyset$ not possible.

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

or
 $R_i \cap R_j \rightarrow R_j$ ($R_i \cap R_j$ 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

$$R_1 \bowtie R_2 = \emptyset$$

[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.

Ques) $R(ABCD)$ $F = \{A \rightarrow B, B \rightarrow C, C \rightarrow D\}$
decomposed into $R_1(ABC)$ $R_2(CD)$

Soln:- C is the common attribute check whether C is super key or not

$CF = CD$ C i.e. C is super key of Relation

\Rightarrow decomposition is loss-less decomposition.

* If $R_1(A,B)$ $R_2(C,D)$

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

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

Ques 1- If $R(A,B,C,D)$ decomposed into

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

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) $S \bowtie R = S$

Ques 3) Decomposed into $R_1(A,B)$ $R_2(A,C)$

R_1	A	B
1	1	
2	1	
3	2	

R_2	A	C
1	2	
2	1	
3	1	

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

$$R_1 \bowtie R_2$$

	A	B	A	C
1	1		1	2
2	1		2	1
3	1		3	1
4	1		1	2
5	1		2	1
6	2		3	1
7	2		1	2
8	2		2	1
9	2		3	1
10	2		1	2
11	2		2	1
12	2		3	1
13	2		1	2
14	2		2	1
15	2		3	1

Arity! - Means addition of attribute of Relations

Cardinality! - Means multiplication of rows of relation

R₁ × R₂

(R₁ × R₂)

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

R₁ × R₂ =

$\pi_{ABC}(\sigma_{R_1 \cdot B = R_2 \cdot B}(R_1 \times R_2))$ Residual relation

⇒

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

Extra
tuple
after
decomposition

R₁ × R₂ ⇒ ?

(lossy Join)

if R ₁ (ABC)	R ₂ (BCDE)
$\text{then } R_1 \Delta R_2 = \pi_{ABCDE}(\sigma_{\begin{cases} R_1 \cdot B = R_2 \cdot B \\ R_1 \cdot C = R_2 \cdot C \end{cases}}(R_1 \times R_2))$	

Example:-

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

(Sid Cname) :- Primary key

$\text{Sid} \rightarrow \text{Sname}$ (Partial Dependency) \nrightarrow Not in
 $\text{Sid Cname} \rightarrow \text{Sname}$ 2NF

Decompose into 2NF:-

R1

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

R2

Sid	Sname
S1	A
S2	B
S3	A

$\text{Sid} \rightarrow \text{Sname}$

Candidatekey : (Sid Cname)

Above Decomposition is

losslessJoin

+
Dependency Preserving

+
2NF (NO partial
Dependency)

Example:-

$R(A B C D E F)$

$\{AB \rightarrow C, C \rightarrow D, B \rightarrow EF\}$

Check the above relation is in 2NF or not.

Solⁿ1 -

$\{AB \rightarrow C, C \rightarrow D, B \rightarrow EF\}$

(Partial Dependency)

Candidate Key :- AB

Convert it to 2NF.

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

$R_2 [B \quad E \quad F]$

$B \rightarrow EF$

B! Candidate key

Put remaining attribute in other relation

$R_1 [A \quad C \quad D]$

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

$R_1 [A \quad B \quad C \quad D]$

$AB \rightarrow C$

$C \rightarrow D$

AB! Candidate key

\Rightarrow Lossless + Dependency Preserving + NO PD(2NF)

$R_2 [B \quad C \quad F]$

$B \rightarrow EF$

B! Candidate key

Example:-

$R(A B C D E F)$

$\{ A B \rightarrow C, C \rightarrow D, B \rightarrow E, A \rightarrow F \}$
PD PD

$A B$! Candidate key.

If we put all PD in one relation:

$A B C F$

$B \rightarrow E, A \rightarrow F$ (Again
PD)

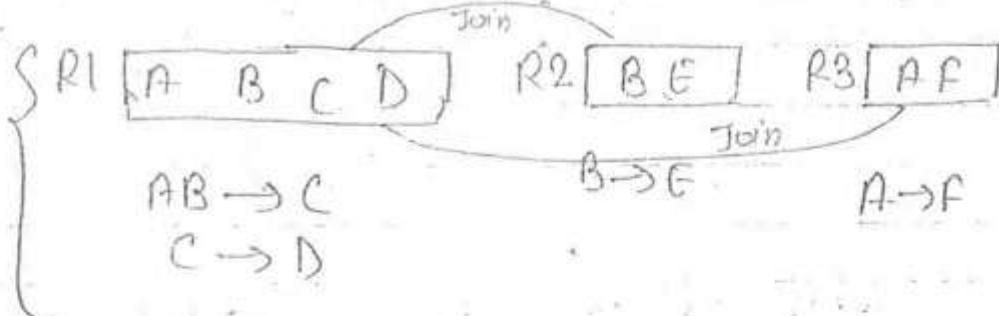
($A B$)! Candidate key

So put all PD in one relation

Dependency
preserving

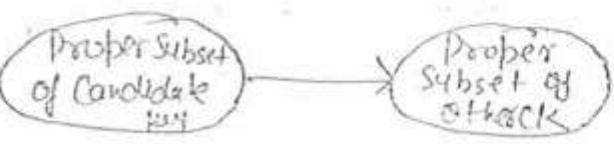
+
lossless join

+
NO PD(2NF)



* \rightarrow 2NF Not free from Redundancy bcoz of

1)  not violate the condition of 2NF

2)  not violate the condition of 2NF

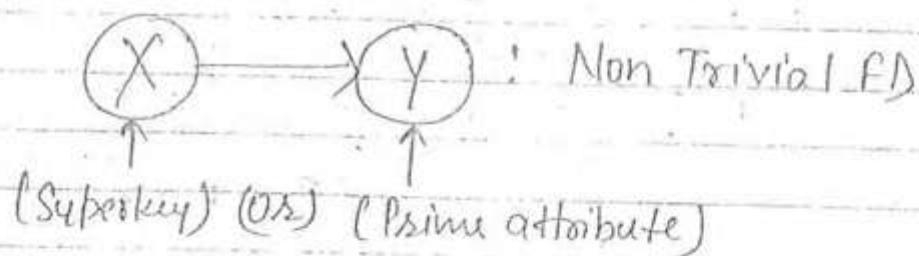
* 2NF has less Redundancy than 1NF.

Third Normal Form

Let R be the relational Schema with $X \rightarrow y$
any Non-trivial FD over R is in 3NF
only if it
satisfies that
the relation
is in 2NF.
There is no
need to
check for
2NF if we
apply these
conditions for
checking that the
relation is in
3NF or not.

a) X is Superkey
(OS)

b) Y is prime attribute.



\Rightarrow 3NF def is also eliminating partial dependencies
we can say that it satisfies the 2NF.

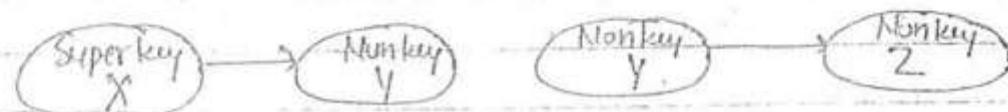
\Rightarrow 3NF eliminates Nonkey \rightarrow Nonkey
dependencies.

3NF - Relation R is in 3NF only if

- 1) R should be in 2NF (no partial dependence)
- 2) Nonkey should not transitively determine by candidate key or Super key.
(No transitive dependency)

Failed i.e.

by 3NF
definition



\Rightarrow Nonkey Z Transitive
determined by Superkey X.

3NF
definition
satisfied



here y is directly
determined by Superkey

Check if it is BNF or not.

(Qn) $R(A B C D E Y)$.

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

Solⁿt - first check R is in 2NF or not

$$\{AB \rightarrow CD, D \rightarrow A, C \rightarrow EY\} \quad \begin{cases} \text{No partial dependence} \\ \text{Candidate key } \{AB, BDY\} \end{cases}$$

so, it is in 2NF.

Check for 3NF

$$\{AB \rightarrow CD, D \rightarrow A, C \rightarrow EY\} \quad \begin{cases} \text{so not is 3NF because} \\ \text{conditional dependency } [D \rightarrow A] \quad X \\ \text{E is functionally determined by} \\ \text{superkey i.e. AB} \end{cases}$$

Decompose into 3NF -

$R_1[A B C D]$

$R_2[C E]$

yet redundancy $AB \rightarrow CD$
err! $\leftarrow(D \rightarrow A)$

$C \rightarrow E$

$\{C\}$! Candidate key

Candidate key: $\{AB, BDY\}$

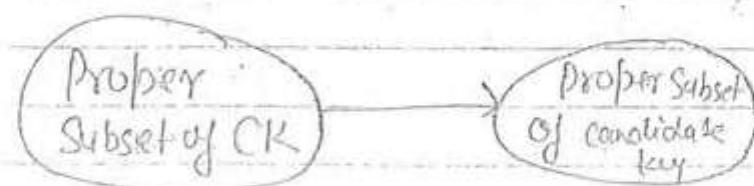
Now there is no violation of 3NF.

So, it is lossless join

Dependency preserving

f
3NF

* 3NF is also not free from Redundancy bcz



is allowed which is the cause of redundancy.

* but Redundancy 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 Redundancy that are formed because of Functional Dependency.

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

Example:-

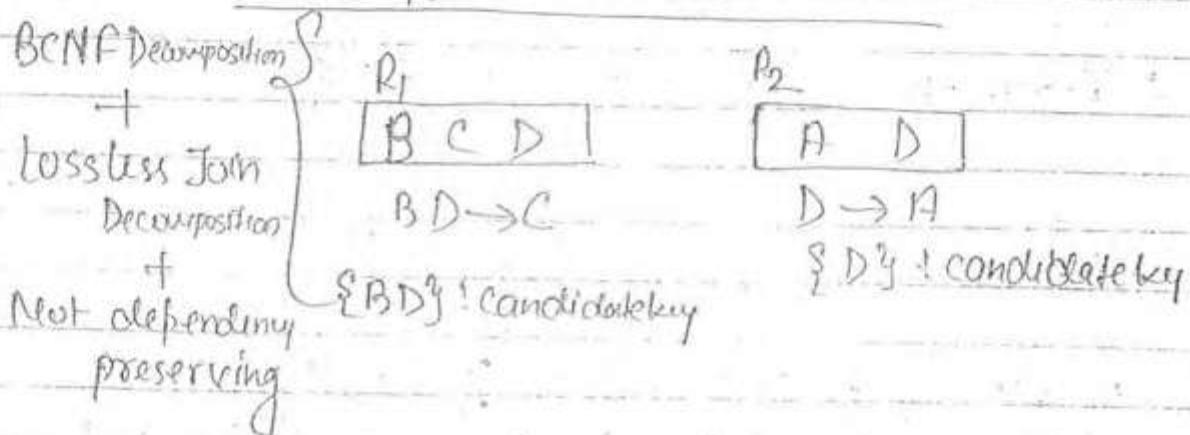
$R_1(A B C D)$
 $\{AB \rightarrow CD, D \rightarrow A\}$

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

{ AB, BD }	✓	✓ (2NF)
Candidate key	✓	✓ (3NF)
	✓	✗ (4NF)

The Relation R is not in BCNF.

Decompose into BCNF -



- * BCNF sometime violates the dependency preservation.
- * Because of BCNF decomposition sometime the original relation R candidate key may lost. That results failure of the dependency preservation.

- * If there is any dependency exist like



- then there must be dependency preservation is violates.

Example:-

$R(ABCDE)$

$R(AB \rightarrow C, C \rightarrow D, B \rightarrow E)$

Candidate Key (AB)

$\boxed{A B C D}$

$AB \rightarrow C$

$(C \rightarrow D)$

$\{A B\}$

Partial Dependency

$\boxed{B E}$

$B \rightarrow E$

$\{B\}$

Decomposit

in 2NF

$\boxed{A B C}$

$AB \rightarrow C$

$\boxed{C D}$

$C \rightarrow D$

Decompositio

in 3NF

We not just 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 equal to 3NF decomposition of R Then Dependency preservation possible.

Example:-

$R(ABCDE)$

$R(AB \rightarrow C, C \rightarrow D, B \rightarrow E)$

$R1 \boxed{ABC}$

$AB \rightarrow C$

$R2 \boxed{CD}$

$C \rightarrow D$

$R3 \boxed{BE}$

$B \rightarrow E$

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	2 NF	3 NF	BCNF
1) O/I Redundancy	High	< 1NF	< 2NF	O/I Redundancy bcz of Single valued function dependencies (Redundancy possible because of MVD)
2) Lossless Join	Always	Always	Always	Always
3) Dependency preserving decomposition	Always	Always	Always	Some Relations may not possible to decompose into BCNF by pres- erving dependen-

* 3NF is more accurate than BCNF.
Even BCNF is more restricted than 3NF but
3NF is more applicable than 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 Depen-
dency. Then 'R' is in BCNF!

bcz $X \rightarrow Y$ non trivial is in BCNF only if X is superkey
 \rightarrow i.e. If nontrivial \rightarrow BCNF
if not nontrivial \rightarrow BCNF

⇒ Binary Relation

Relation with Two attributes always in BCNF

$R(A, B)$

a) $A \rightarrow B$

Candidate key?

A

b) $B \rightarrow A$

B

c) $A \rightarrow B, B \rightarrow A$ A, B

In
BCNF

d) No Non Trivial FD: AB

(Ques) Assume no multivalued attribute

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

$f = \{A \rightarrow BCDEF, BC \rightarrow ADEF, B \rightarrow E, D \rightarrow E\}$

Solⁿ:-

$f = \{A \rightarrow BCDEF, BC \rightarrow ADEF, B \rightarrow E, D \rightarrow E\}$

Candidate key: $\{A, BC\}$ (Not in 2NF)

\boxed{ABCDEF}

$\checkmark A \rightarrow BCDEF$

$\checkmark BC \rightarrow ADEF$

not in 3NF ($D \rightarrow E$)

$\{A, BC\}$ candidate key

\boxed{BF}

$\checkmark B \rightarrow F$

$\{B\}$! Candidate key

\boxed{ABCD}

$A \rightarrow BCD$

$BC \rightarrow AD$

$\{A, BC\}$! Candidate key

\boxed{DE}

$D \rightarrow E$

$\{D\}$! Candidate key

\boxed{BF}

$B \rightarrow F$

$\{B\}$! Candidate key

2NF

+

Not

3NF

-

3NF

+

BCNF

-

BCNF

RB CF ADE
 $A \rightarrow DE$

(One)

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

$\{AB \rightarrow C, A \rightarrow DE, B \rightarrow F, F \rightarrow GH, D \rightarrow IJ\}$

Soln - $\{AB \rightarrow C, A \rightarrow DE, B \rightarrow F, F \rightarrow GH, D \rightarrow IJ\}$

Candidate

Key $\{AB\}$

~~AB~~ ~~BC~~ ~~CF~~ ~~DF~~ ~~GP~~

2NF Decomposition

[A B C]

$AB \rightarrow C$

[B F G H]

$B \rightarrow F$

$F \rightarrow GH$

[A D E I J]

$A \rightarrow DE$

$D \rightarrow IJ$

Consolidate key $\{AB\}$

3NF Decomposition

[A B C]

$AB \rightarrow C$

[B F]

$B \rightarrow F$

[F GH]

$F \rightarrow GH$

[A D E]

$A \rightarrow DE$

[D I J]

$D \rightarrow IJ$

It is also in BCNF

11

4

(Ques) $R(A B C D E F G)$

$\{ A \rightarrow B C D E F G, B C \rightarrow A D E F, B \rightarrow F, D \rightarrow F \}$

R1

$A B C D E F G$

$A \rightarrow B C D E$

$B C \rightarrow A D G$

R2

$B F$

$B \rightarrow F$

R3

$D F$

$D \rightarrow F$

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

$\{ A B \rightarrow C, A C \rightarrow B, A D \rightarrow E, B \rightarrow D, B C \rightarrow A, E \rightarrow H \}$

Solⁿl-

$\{ A B \rightarrow C, A C \rightarrow B, A D \rightarrow E, B \rightarrow D, B C \rightarrow A, E \rightarrow H \}$

R1

✓

✓

PD

✓

Candidate key (ABFH) BCFH, ACFH)

2NF Decomposition

R1

$A B C E F H I$

$A B \rightarrow C$

$A C \rightarrow B$

$B C \rightarrow A$

PD $E \rightarrow H$

R2

$B D$

$B \rightarrow D$

R3

$A D E$

$A D \rightarrow E$

Candidate key $\{ A B F H, B C F H, A C F H \}$

{B}

{A D}

R1! Again it is not in 2NF

R₁

R₂

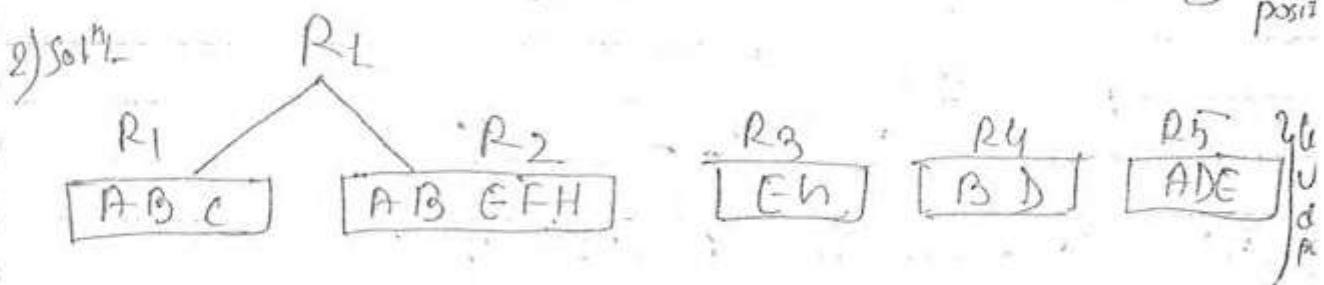
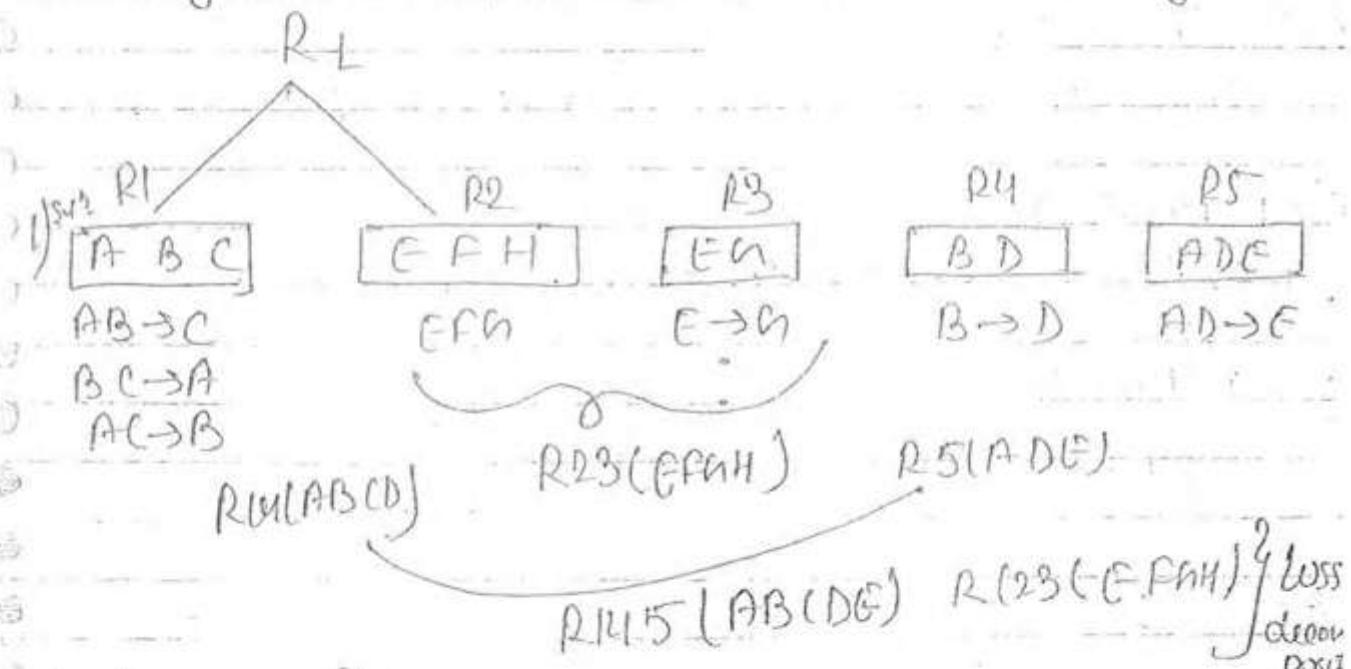
R₃

R₄

R₂₄

NF	R ₁	R ₂	R ₃	R ₄	R ₂₄
1NF	[ABC, CEFH]	[E, h]	[B, D]	[A, D, E]	[ADEH]
	$AB \rightarrow C$	$E \rightarrow h$	$B \rightarrow D$	$AD \rightarrow E$	$AD \rightarrow$
	$BC \rightarrow A$				$E \rightarrow h$
	$A, C \rightarrow B$				
	{ABC, CEFH, ACEFH}				

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



22/08/2011

(Q4) Identify highest Normal Form

[1] R(ABCDEFH)

{ABC \rightarrow DE, E \rightarrow FG, H \rightarrow I, I \rightarrow H, ABCE \rightarrow EF, F \rightarrow G}

[2] R(ABCD(E))

{AB \rightarrow C, C \rightarrow D, D \rightarrow E, E \rightarrow A, D \rightarrow B}

[3] R(ABCDEFH_nH)

{AB \rightarrow CD, D \rightarrow E, E \rightarrow F, F \rightarrow H, C \rightarrow EF, H \rightarrow A, H \rightarrow B, A \rightarrow C}

[4] R(ABCDEF)

{AB \rightarrow C, C \rightarrow D, D \rightarrow E, E \rightarrow F, F \rightarrow A}

[5] R(AB(CD)EF)

{AB \rightarrow C, C \rightarrow DE, E \rightarrow F, F \rightarrow A}

[1] Soln: Candidate key [AB C]

2NF ✓

3NF ✗

BUNF ✗

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

{AB \rightarrow C, C \rightarrow D, D \rightarrow E, E \rightarrow A, D \rightarrow B}

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

3NF! ✓

B(CNF) X

(3) Solⁿ) - Candidate keyl - { AH, HH, FH, CH, DH, CH }.

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

Not in 2NF

so, it is in 1NF

(4) Solⁿ) - Candidatekeyl - { AB, FB, EB, DB, CB }

Not in B(CNF). But it is in 3NF. Because right side of AB → C, C → D, D → E, E → F, F → A consider all Prime attribute. So, it is in 3NF

(5) Solⁿ) - Candidatekeyl - { 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:

$$F: \{A \rightarrow B, B \rightarrow C, A \not\rightarrow C, AB \not\rightarrow C\}$$

$A \rightarrow C$ is eliminated

$$A \rightarrow B, B \rightarrow C \Rightarrow A \rightarrow C$$

$AB \rightarrow C$ is eliminated because

Augmentation Property: If $B \rightarrow C$

$$AB \rightarrow AC$$

$$AB \rightarrow A$$

$$AB \rightarrow C$$

$$\text{then } h \subseteq \{A \rightarrow B, B \rightarrow C\}$$

$$\Rightarrow (F \equiv h)$$

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

Extraneous Attribute -

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

B is extraneous Attribute -

$$f = \{AB \rightarrow C, A \rightarrow C\}$$

$$h = \{A \rightarrow C\}$$

cover h - True

cover f :- $h = \{A \rightarrow C\} \models AB \rightarrow C$ True
 $\models A \rightarrow C$ True

$\Rightarrow A \rightarrow C$ determine all the FD of F's

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

Proof - First check A is extraneous

$$f = \{AB \rightarrow C, A \rightarrow B\}$$

$$h = \{B \rightarrow C, A \rightarrow B\}$$

cover f ! - $f = \{AB \rightarrow C, A \rightarrow B\} \models B \rightarrow C \times$
 $A \rightarrow B \checkmark$

cover f :- $h = \{B \rightarrow C, A \rightarrow B\} \models \overline{AB \rightarrow C} \checkmark$
 $A \rightarrow B \checkmark$

$$f \neq h$$

Take B is extraneous attribute

$$F_2 \{ AB \rightarrow C, A \rightarrow B \} \models A \rightarrow C \checkmark$$
$$\qquad\qquad\qquad \models A \rightarrow B \checkmark$$

$$n = \{ A \rightarrow c, A \rightarrow B \} \models AB \rightarrow C \checkmark$$
$$\qquad\qquad\qquad \models A \rightarrow B \checkmark$$

Flushing b \rightarrow \checkmark
Cover F! \rightarrow \checkmark
($F \equiv n$)

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

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

Proof -

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

Combination of transitivity & argumentation

if $AB \rightarrow C$

$AB \rightarrow BC$ if $BC \rightarrow D$

then $AB \rightarrow D$

∴ "C & d" are used to remove extraneous attribute of right side, "d" & "b" are used to remove extraneous attribute of left side.

(Ques) Minimal cover of

$$F = \{ A \rightarrow BC^*, B \rightarrow C, C \rightarrow B \}$$

Solⁿ - $F_1 = \{ A \rightarrow BC^*, B \rightarrow C, C \rightarrow B \}$ $F_2 = \{ A \rightarrow BC, B \rightarrow C, C \rightarrow B \}$

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

$$F_1 = F_2 = F$$

⇒ Minimal cover is not unique. But all minimal covers logically equivalent.

(Ques) $F = \{ AB(D \rightarrow E, E \rightarrow D), A \rightarrow B, AC \rightarrow D \}$

Solⁿ - $\{ AC(D \rightarrow E, E \rightarrow D), A \rightarrow B, AC \rightarrow D \}$

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

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

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

(Ques) Identify minimal cover -

$$\{ A \rightarrow BC^*, B \rightarrow C, A \rightarrow B, AB \rightarrow C^* \}$$

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

$$\{ A \rightarrow B, B \rightarrow C, AB \rightarrow C^* \}$$

$$\{ A \rightarrow B, B \rightarrow C, A \rightarrow C^* \} \rightarrow \{ A \rightarrow B, B \rightarrow C \}$$

$$\{ A \rightarrow BC, B \rightarrow C^* \}$$

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

(Ques)

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

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

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

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

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

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

L

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

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

$\{A \rightarrow B, A \rightarrow C, B \rightarrow A, C \rightarrow AB\}$ ($A \rightarrow B, A \rightarrow C, B \rightarrow AC, C \rightarrow AB$)
($A \rightarrow B, A \cancel{\rightarrow} C, B \rightarrow A, C \rightarrow AB$)

$\{A \rightarrow B, A \rightarrow C, B \rightarrow A, C \rightarrow AB\}$ ($A \rightarrow BC, B \rightarrow A, C \rightarrow AB$)

$\{A \rightarrow BC, B \rightarrow A, C \rightarrow AB\}$.

Identify minimal covers -

$\{A \rightarrow BC, CD \rightarrow E, E \rightarrow C, D \rightarrow AEGH, AH \rightarrow BD, DH \rightarrow BC\}$

Soln: $\{A \rightarrow BC, \cancel{CD} \cancel{\rightarrow} E, E \rightarrow C, D \rightarrow AEGH, AH \rightarrow BD, DH \rightarrow BC\}$

$\{A \rightarrow BC, E \rightarrow C, D \rightarrow AEGH, AH \rightarrow BD, \cancel{DH} \rightarrow BC\}$

$\{A \rightarrow BC, E \rightarrow C, D \rightarrow AEGH, AH \rightarrow D, D \rightarrow BC\}$

$\{A \rightarrow BC, D \rightarrow AEGH, E \rightarrow C, AH \rightarrow D\}$

$\{A \rightarrow BC, D \rightarrow AEGH, E \rightarrow C, AH \rightarrow D\}$

L

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, ~~cover~~ of the original Relation's FD.

Multivalued Dependency:- (MVD)

Representation of Functional Dependency-

Single Valued FD \Rightarrow FD , \rightarrow

Multivalued FD \Rightarrow MVD $\rightarrow \rightarrow$

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 $\{ACD\}$

→ $\{R(A, B, C, D, E, F)\}$

$\{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A\}$

and E, F are Multivalued attribute

Then Candkey :- A, B, C, D

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

$\{R(A, B, C, D, E, F)\}$ IT No multivalued attributes.

$\{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A\}$

Cand key :- AEF, BCF, CEF, DEF

* If the candidate key determine all the attribute then relation does not contain any multivalued attribute.

Ques. If $R(A, B, C, D, E, F)$

and $\{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A\}$

big 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 1N Then MVD is possible.

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

A: Primary key C, D: Multivalued Attribute

1NF(ACD, B)

ACD: Primary key

[ACD] May suffers with
MVD

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

Multivalued attribute
Selection

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

Teacher is Primary key

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

→ (Single Valued Attribute Relation.)

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

Data loss occurred such that independent attribute (C, B) becomes defining each other which is incorrect.

To avoid above problem i.e. course & book depends we need to add more table.

BCNF ↗

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

(i.e. To Avoid data loss requires to add more tables)
⇒ There exist redundancy data b/w(TC) & (TB)

→ Candidate Key: TCB

⇒ Highest Normal Form of R is BCNF.

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

Example -	Teacher	course	Book
	A	CS/OS/DB/DS/CN	K/N/U

Total 15 tuples

[TC]: repeated 3 times

[TB]: repeated 5 times

Definition of MVDI-

Let R be the Relational schema X, Y are attributes of R and Z is $R - (X \cup Y)$

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

$$1) t_1 \cdot X = t_2 \cdot X = t_3 \cdot X = t_4 \cdot X$$

$$\text{and } 2) t_1 \cdot Y = t_2 \cdot Y \text{ & } t_3 \cdot Y = t_4 \cdot Y$$

$$\text{and } 3) t_1 \cdot Z = t_3 \cdot Z \text{ & } t_2 \cdot Z = t_4 \cdot Z$$

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

Representation of definition in Table form -

R	X	Y	Z		X	Y_1/Y_2	Z_1/Z_2
t_1	x_1	y_1	z_1				
t_2	x_1	y_1	z_2				
t_3	x_2	y_2	z_1				
t_4	x_1	y_2	z_2				

$$X \rightarrow \rightarrow Y$$

After interchanging the tuple we obtain complementation rule

	X	Y	Z		X	Y	Z
t_1	x_1	y_1	z_1		x_1	y_1	z_1
t_2	x_1	y_2	z_2		x_1	y_2	z_1
t_3	x_2	y_1	z_1		x_1	y_1	z_2
t_4	x_1	y_2	z_2		x_1	y_2	z_2

$$X \rightarrow \rightarrow Y$$

$$X \rightarrow \rightarrow Z$$

① Complementation Rule -

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

② Trivial MVD - (Reflexive MVD)

If $X \supseteq Y$ (or) $X \cup Y = R$ Then $(X \rightarrow \rightarrow Y)$ is trivial

Example - R(A,B) $\left\{ \begin{array}{l} A \rightarrow \rightarrow A \\ A \rightarrow \rightarrow B \end{array} \right\}$ Trivial

L

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

[3] Transitivity:-

$$\text{If } X \rightarrow Y \text{ & } Y \rightarrow Z \text{ then } X \rightarrow (Z - Y)$$

|
 ABC |
 CDG |
 DG

If there is no any common attribute

$$\text{then } X \rightarrow Z$$

[4] Argumentation:-

$$\text{if } X \rightarrow Y \text{ and } W \supseteq Z \text{ then } XW \rightarrow YZ$$

Example:-

$$\text{if } A \rightarrow B \text{ then } ACD \rightarrow BC$$

$$\text{if } A \rightarrow B \text{ then } ACD \rightarrow BCD$$

* Every trivial multivalued dependency is also reflexive multivalued dependency

[5] Replication:-

$$\text{if } X \rightarrow Y \text{ then } X \rightarrow Y$$

* (Union / Splitting Not allowed)

Example:-

A	B
1	2
1	2
1	2
1	2
0	2

$$(A \rightarrow B) = (A \rightarrow \rightarrow B)$$

$(A \rightarrow \rightarrow B)$ is trivial
while \exists is not

Ques) All possible MVD

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

If $A \rightarrow \rightarrow B$,

Due to complement

$A \rightarrow \rightarrow C$

* $AC \rightarrow \rightarrow B$

* $AB \rightarrow \rightarrow C$

* $AB \rightarrow \rightarrow BC$ * $AC \rightarrow \rightarrow BC$

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

If $B \rightarrow A$ then $B \rightarrow \rightarrow A$, $BC \rightarrow \rightarrow A$ (Due to arguments
 $BC \rightarrow \rightarrow AC$)

$BC \rightarrow A$ then $BC \rightarrow \rightarrow A$

4NF:-

Relational Schema R is in 4NF
only if

(a) R should be in BCNF

(Non-trivial FD $X \rightarrow Y$ with X: Superkey)

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

Note:-

Superkeys or minimal superkeys as 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

$\{ T \rightarrow\rightarrow C, T \rightarrow\rightarrow B \}$
 \uparrow \uparrow
 Superkey 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

$$\underbrace{T \rightarrow C}$$

Trivial
MVD

$$\underbrace{T \rightarrow B}$$

Trivial
MVD

Prospective Results:-

2NF

Redundancy! X

3NF

X

BCNF

X

4NF

✓

(Redundancy
possible b/w
of MVD)

Lossless ✓

✓

✓

✓

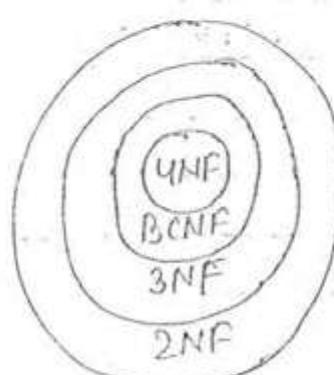
Dependency
Preservation ✓

✓

X

X
(b/w it
is also in
BCNF)

This diagram shows
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 Relation
Answers a question
on R

Query:- Retrieve Student whose marks greater than 60

Relational Algebra

$\Delta, \times, \sigma, \cup, -$
etc.

SQL

SELECT from
WHERE clause

Tuple Relational
Calculus

(First Order logic)
 $\epsilon, \exists, \forall, \vee, \wedge, \neg, \rightarrow$ etc.

Relational Algebra

* Always eliminates duplicate tuple from the result

Ex ij A CS
 | A CS
 | A ST

It gives O/P only A CS
 A ST

Basic Operators

π : Projection

σ : Selection

\times : cross product

\cup : Union

$-$: Minus

δ : Rename

Derived operators

\cap : Intersection

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

\bowtie : Join

$/$: Division

Set Operators

$\cap, \cup, -$: To apply these set operators,
Relations should be Union Compatible.

R & S said to be Union compatible!
only if

a) domain of R & S should be same

* If no. of attribute in R₁ is 2 & no. of attribute in R₂ is 3 then U, ∩, - is not applicable there.

Ex:-

Sid	Sname	X	✓
SI	A	g	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 U, ∩, -.

Ex:-

Sid	Sname	X	✓
	Varchar(2)		Varchar(2)

Sid	Sname	U	✓	Sid	Sname
	Varchar(2)	U	✓		Varchar(2)

R & S cardinalities are m & n each.

RUS : [Range of Tuples] { max(m,n) to min(m,n)}

R ∩ S : " { ϕ to min(m,n) }

R - S : " { ϕ to m }

Cross Product

R & S have Relations Cardinality m, n respectively

and arity n, y respectively

then RXS { cardinality : m * n
arity : x + y }

Example:-

A	B	C
1	2	3
2	2	1

S

B	C	D
2	3	3
2	4	1

RXS

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 another relation or other relation.

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

Cross

* Conditional Join :- (ΔC)

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

$$R \Delta S = \pi_{C, CC_2} (\sigma_{R.A < S.B} (R \times S))$$

Example

$$R \Delta S = \pi_{A, B, C, D} (\sigma_{R.A < S.B} (R \times S)) \rightarrow$$

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

* Natural Join :- (\bowtie)

* It always project distinct column

Example

$$R \bowtie S = \pi_{A, B, C} (\sigma_{R.B = S.B \wedge R.C = S.C} (R \times S)) \rightarrow$$

A	B	C
1	2	3

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

* Left Outer Join :- 

$R \bowtie S$ and Tuples from left Relation those are failed N. Join condition.)

$R \bowtie S$

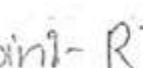
	A	B	C	D
1	1	2	3	3
2	2	2	Null	

* Right Outer Join - 

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

$R \bowtie S$

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

* Full Outer Join - 

$$R \bowtie S = (R \bowtie S) \cup (R \bowtie S)$$

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	Name
C1	-
C2	-
C3	-

Query1- Sid's who are Enrolled some course.

Sid
S1
S2
S3

$\pi_{\text{Sid}} (\text{Enrolled})$

Query1- Sid's who are Enrolled all courses:

O/P:- Sid

$\pi_{\text{Sid}} (\text{Enrolled}) / \pi_{\text{Cid}} (\text{Course})$

* 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 B Rela

✓(Enrolled) —
Sid

π (Emulated) / π (course)
Sicid / cicl

∇ (Enrolled) X (course
S/d S/d Enrolled)

Student	Course
S1	C1
S2	C2
S3	C3
all students	all courses

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

$\pi(\text{Enrolled})/\pi(\text{course})$
Siderid Cid

$$\pi(\text{Enrolled}) = \left[\pi_{\text{std}} \left(\pi_{\text{std}}(\text{Enrolled}) \times (\text{course} - \text{std}) \right) \right] \text{Enrolled}$$

(Universal)
(Every Student
Enrolled Every
Course)

(Actual
Enrollees
Student
who are
enrolled
(some)
receive

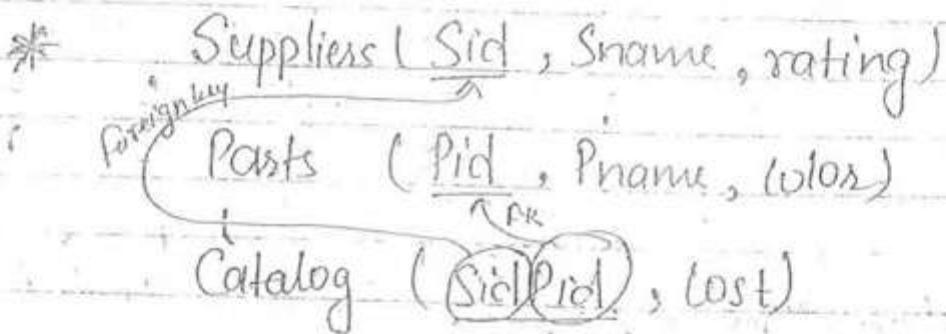
(At least
one year)

Sid	Cid
S2	C3
S3	C2
S3	C3

(complement data)

(Students who are not enrolled in all courses)

23/08/11



Query! - Select Sid's of suppliers whose rating greater than 10.

$\pi_{\text{Sid}} \left(\sigma_{\text{rating} > 10} (\text{Suppliers}) \right)$

Query! - Sid's of suppliers whose rating greater than 10 or who supplied some part.

Supplier

Sid	Rating
S1	5
S2	15
S3	7

Catalog

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

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

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

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

By joining

eliminates S3

because it not

supply any

part because

S3 is not in

Catalog sid but

it is in supplier

sid

Suppliers

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

Catalog

sid	pid	-
S1	P1	-
S2	P1	-
S1	P2	-

$\pi_{sid} (\sigma_{(Supplier \times Catalog)} (some s.sid = c.sid \wedge rating > 10))$

Q2

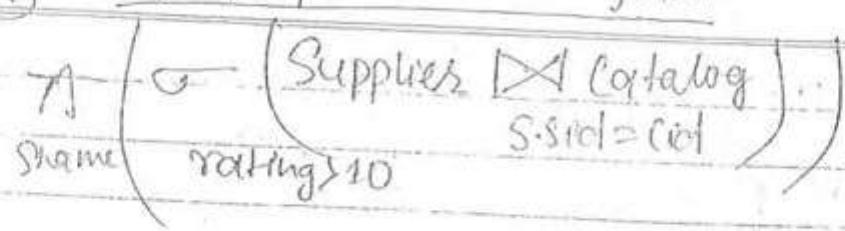
By using natural join

$\Sigma_{(02)} \pi_{name} (\sigma_{rating > 10} (suppliers \bowtie catalog))$

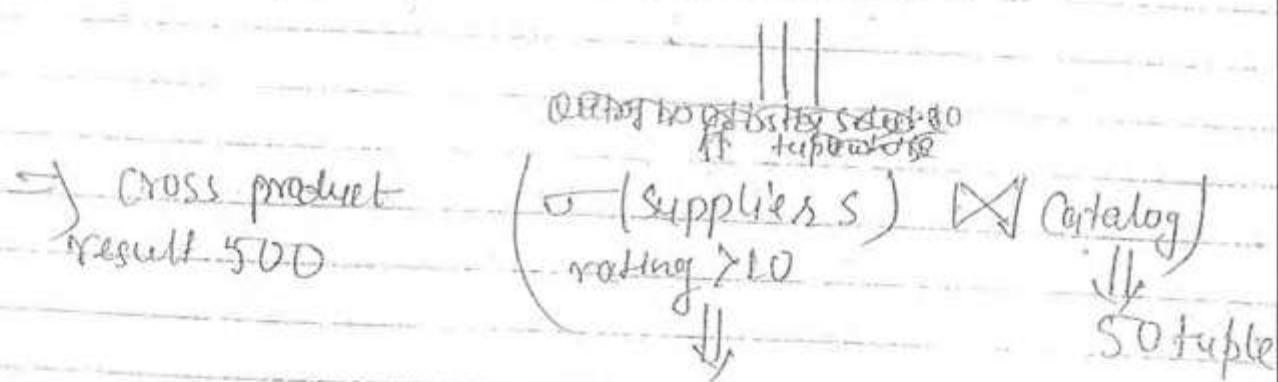
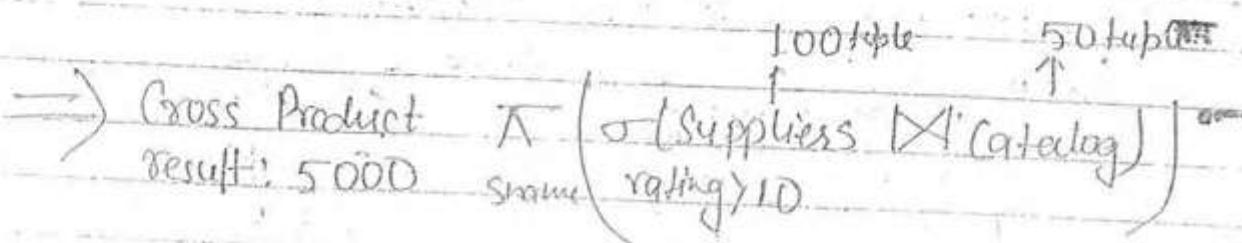
[03]

By applying conditional join
Suppliers \bowtie Catalog
s.sid = c.sid
rating > 10

[Q4] By apply conditional join



[Q1], [Q2], [Q3], [Q4] are different way of writing the same query.



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

Query -

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

π (σ (Suppliers \bowtie Catalog))
 where σ (rating > 10 \wedge cost > 100)

|||

π (σ (Suppliers) \bowtie (σ (Catalog))
 where σ (rating > 10) $\quad \sigma$ (cost > 100)

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 depends for the cost of the relation. so, it decrease the total cost of the whole relation.

*

Let $A \in$ Relation R only.

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

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

④ π (Suppliers \bowtie (Parts))

[01] π (σ (Suppliers \bowtie Catalog \bowtie Parts))
Sname (C01 = Red)

|||

[02] π (σ (Suppliers \bowtie Catalog \bowtie Parts))
Sname (S.Srel = C.Srel \wedge
C.Pid = P.Pid \wedge
Color = RED)

|||

[03] π (name (π (Parts) \bowtie Catalog) \bowtie Suppliers)
Srel (color = RED)

|||

If foreign key constraints is not there then

No. of Tuples in sup catalog = 100
min μ = 4 n = 10 μ = 0

(a) $R(AB)$ $S(BDE)$

with R consist 1000 tuples S consists 9000 tuple
and following dependencies are hold.

$\S B \rightarrow D, D \rightarrow E^Y$

How many max. no. of tuples in RDS.

Solⁿ - Maximum no. of tuples = 1000

Rename operators

Two syntax for Rename operators

1) Rename the Table S (T, Catalog)

2) Rename the column g1 Catalog Catalog
SPC S P C

§ (Catalog)

$1 \rightarrow S, 2 \rightarrow P$

$\{ \text{Spec} \left(\mathcal{S}(T, \text{catalog}) \right) \}$

S P C

* Use of Rename Operator:-

σ (Catalog \times Parts) \equiv σ ($\delta(c, \text{catalog}) \times \delta(p, \text{Parts})$)
Catalog.Pid = parts.Pid \wedge c.pid = p.pid \wedge col2red

\exists σ (Catalog \times $\delta(\text{parts})$)
pid = p1.col1.pid \wedge p1.c

Query1- Retrieve Sid of the Suppliers who supply some red part or some green part.

[Q1]

π_{Sid} (Catalog \bowtie σ (Parts)
col1 = GREEN
col1 = RED)

By using union operators:-

$\{ T_1, \pi_{\text{Sid}} (\text{Catalog} \bowtie \sigma (\text{Parts})) \}$
col1 = RED

$\{ T_2, \pi_{\text{Sid}} (\text{Catalog} \bowtie \sigma (\text{Parts})) \}$
col1 = green

$T_1 \cup T_2$

(Ques) Retrieve "Sid" of the Suppliers who supplied some red part and some green part.

$\Rightarrow [01] \ \exists T_1, \pi_{Sid} (\text{Catalog} \bowtie_{Sid} (\text{parts}))$

$\exists (T_2, \pi_{Sid} (\text{Catalog} \bowtie_{Sid} (\text{parts})))$

$T_1 \cap T_2 \Rightarrow (T_1 \bowtie_{Sid} T_2)$

$\times [02] \ \pi_{Sid} (\text{Catalog} \bowtie_{Sid} (\text{parts}) \underset{\text{color = GREEN}}{\text{---}} \underset{\text{color = RED}}{\text{---}})$

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

* without using the union operator -

$\exists (P1, \text{Parts})$

$\exists (P2, \text{Parts})$

$\exists (C1, \text{Catalog})$

$\exists (C2, \text{Catalog})$

$\pi_{C1.Sid} \left(\sigma_{C1.Sid = C2.Sid} \left((C1 \bowtie_{C1} P1) \underset{\text{color = RED}}{\text{---}} \right) \times (C2 \bowtie_{C2} P2) \underset{\text{color = GREEN}}{\text{---}} \right)$

By using natural join :-

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

Query π_{sid} of the suppliers who supplied some red parts

$$\pi_{sid} \left(catalog \bowtie_{color=Red} (parts) \right)$$

sid of the suppliers who supplied every red parts.

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

$$\pi_{sid} \left(catalog \right) / \pi_{pid} \left(\sigma_{col=RED} (parts) \right) =$$

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

example - catalog

Slot	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.

$$\pi_{pid}^{\text{catalog}} / \pi_{sid}^{\text{supplier}} =$$

$$\pi_{pid}^{\text{catalog}} - \pi_{pid}^{\text{catalog}} \times \pi_{sid}^{\text{supplier}} = \pi_{pid}^{\text{catalog}}$$

These are parts not supplied by eve supplier

Query- Retriene Sid of the Suppliers who supplied at le two parts.

$$\pi_{sid}^{\text{catalog}} \\ \pi_{sid}^{\text{catalog}}$$

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

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

Sid	Pid	Cust	Sid	Pid
S1	P1	-	S1	P1
S1	P1	-	S2	P1
S1	P1	-	S1	P2
S2	P1	-	S1	P1
S2	P1	-	S2	P1
S2	P2	-	S1	D2
S2	P2	-	S1	P1
S2	P2	-	S2	P1
S2	P2	-	S1	P2

$$(\pi_{sid}^{\text{catalog}} \times \pi_{sid}^{\text{supplier}})$$

S1 = Sid
P1 = Pid

$\delta(T_1, \text{Catalog}) \ \ \delta(T_2, \text{Catalog})$

$\pi \left(\sigma(T_1 \times T_2) \right)$

$T_1.Sid$ $T_1.Sid = T_2.Sid$
 $T_1.Pid \neq T_2.Pid$

|||

$\pi_{Sid} \left(\text{Catalog} \bowtie \delta(\text{Catalog}) \right)$
SPC
 $Sid = S.A$
 $Pid \neq P$

Ques4! Retrieve Sid of the Suppliers who supplied at least Three parts.

$\delta(T_1, \text{Catalog}) \ \ \delta(T_2, \text{Catalog}) \ \ \delta(T_3, \text{Catalog})$

$\pi_{Sid} \left(\sigma(T_1 \times T_2 \times T_3) \right)$

$T_1.Sid$ $T_1.Sid = T_2.Sid \wedge$
 $T_2.Sid = T_3.Sid \wedge$
 $T_1.Pid \neq T_2.Pid \wedge$
 $T_2.Pid \neq T_3.Pid \wedge$
 $T_3.Pid \neq T_1.Pid$

Ques4! Retrieve Sid's of suppliers who supplied exactly two parts.

Sid's of Supplies who supplied at least two parts

Sid's of Suppliers who supplied at least three parts

Query1- Sid of the supplier who supplied at most two parts.

$\pi_{\text{Sid}}(\text{Suppliers}) - \{ \text{Sid}'s \text{ of suppliers who supplied at least three parts} \}$

If suppliers Table is not in DB then we take Catalog Table in place of suppliers because then there is no any table who gives 0 parts.

Query1- 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.

$f(T_1, \text{Catalog})$

$f(T_2, \text{Catalog})$

$\pi_{T_1.\text{Sid}, T_2.\text{Sid}}(\sigma_{T_1.\text{cost} > T_2.\text{cost} \wedge T_1.\text{Pid} = T_2.\text{Pid}}(T_1 \times T_2))$

$\cancel{T_1.\text{Sid} \neq T_2.\text{Sid}}$ No need to write this condition because of key constraints because Sid & Pid are primary key. So, Sid must not be repeated.

Ques) Retrieve SId of the supplier who supplied most expensive cost.

SId	Pid	Cust	SId	Pid	Cust	
S1	P1	10	X	S1	P1	10
S2	P1	20	X	S2	P1	20
S3	P2	30	X	S3	P2	30

SId	Pid	Cust	SId	Pid	Cust
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)

Suppliers Info who
doesn't supplied
most exp part

(S2 P1 20)
S1 P2 30)

Suppliers Info
who doesn't supplied
least expensive part

$\sigma(T_1, \text{catalog}) \quad \sigma(T_2, \text{catalog})$

$\sigma(T_3, \pi_{T_1.\text{Sid}, T_1.\text{Did}, T_1.\text{cost}}) \quad (\sigma(T_1 \times T_2) \quad T_1.\text{cost} < T_2.\text{cost})$

$\pi_{\text{Sid}}(\text{Catalog} - T_3)$

→ supplies who supplied least expensive part

$\sigma(T_3, \pi_{T_1.\text{Sid}, T_1.\text{Did}, T_1.\text{cost}}) \quad (\sigma(T_1 \times T_2) \quad T_1.\text{cost} > T_2.\text{cost})$

$\pi_{\text{Sid}}(\text{Catalog} - T_3)$

Or

$\sigma(T_3, \pi_{T_2.\text{Sid}, T_2.\text{Did}, T_2.\text{cost}}) \quad (\sigma(T_1 \times T_2) \quad T_1.\text{cost} < T_2.\text{cost})$

$\pi_{\text{Sid}}(\text{Catalog} - T_3)$

Query:- Retain the supplier who supplied 2nd most expensive part.

$\delta(T_1, \text{catalog}) \quad \delta(T_2, \text{catalog})$

$\delta(T_3, \pi_{T_1 \times T_2} \mid T_1.\text{Sid}, T_2.\text{Pd}, T_1.\text{cost} < T_2.\text{cost}, T_1.\text{cost})$

$\delta(T_4, T_3)$

$\delta(T_5, \pi_{T_3 \times T_4} \mid T_3.\text{Sid}, T_4.\text{Pd}, T_3.\text{cost} < T_4.\text{cost}, T_3.\text{cost})$

$\pi_{\text{Sid}}(T_3 - T_5)$

T3	T5
Sid	Sid
Pid	Pid
Cost	Cost

(Qw) Ensal (scl, cld)

Student (sict, name, sex)

Every course taken by at least one male & at least one female student:

What this relation algebra query represent

$\pi_{\text{enroll}} \left(\pi_{\text{student}} \left(\sigma_{\text{Sex} = \text{F}} \left(\text{sid} \right) \right) \times \pi_{\text{enroll}} \left(\text{cid} \right) \right) - \text{enroll}$

- a) course in which all F students are enrolled
 - b) course in which proper subset of 'F' students are final
 - c) course " " only male students are enrolled
 - d) None

$\pi_{\text{Crd}} \left(\left(\pi_{\text{std}}^q \left(\text{Sex} = f \text{ (Student)} \right) \right) \times \pi_{\text{Crd}} \left(\text{enroll} \right) \right) \rightarrow \text{enroll}$

$\{ \text{Every female student enrolled by } \}$ - $\{ \text{Students who are enrolled some course} \}$

* Take the content of table based on the option given in the question. Discard the option on the basis of the output obtained after solving that question on the table record.

Sid	Sname	Sex	Sid	Cid
S1		F	S1	C1
S2		F	S2	C1
S3		F	S3	C1
			S1	C2
			S2	C3
			S3	C3

S1	C1	S1	C1	S1	C1
S2	X C2	S1	C2	S2	C1
S3	C3	S1	C3	S1	C3
		S2	C1	S3	C1
		S2	C2	S1	C2
		S2	C3	S2	C3
		S3	C1	S3	C3
		S3	C2		
		S3	C3		

Course

C2 \Rightarrow S1 S3

C3 \Rightarrow S2 S3

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

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

$\pi_{Sid}^r (\sigma_{Sex='Female'} (student)) - \pi_{Sid}^r (student \setminus X | f(student))$

$\sigma_{Sex='Female'}$

$X = \text{male}'$

Marks $\leq M$

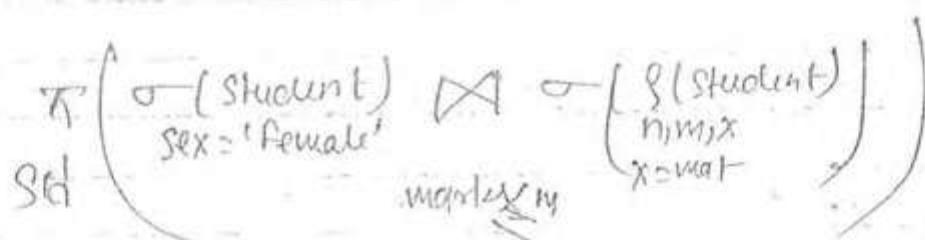
All Female student
16

Student		
Sid	marks	Sex
S1	10	M
S2	20	F
S3	30	F
S1	10	M
S2	20	F
S2	20	F

Student		
n	m	x
S1	10	M
S2	20	F
S3	30	F
S1	10	M
S2	20	F
S3	30	F

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

Solⁿ1-



Sid	marks
S2	20
S4	40
S6	60

n	m
S1	10
S3	30
S5	50

Complement	
>	<=
all	some
>=	<

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

S2
S4

All Female
Student

Female students who scored less than or
equal to some male student)

- a) who scored more marks than some male student
 b) u " " not less than same male student

~~c) u " more marks than all male "~~

d) " " more than equal to " " "

g)

F Student who
 scored more than \rightarrow (All Female) (Female student who
 is equal to all Students) scored less than some
 male students
 \Rightarrow (b), (d) is correct option

* ① $\{ \forall, \geq \} \Leftrightarrow$ (Universal Set) - $(\exists, <^n)$

② $\{ \forall, \geq \} \Leftrightarrow$ (Universal Set) - (\exists, \leq)

③ $\{ \exists, \geq \} \Leftrightarrow$ (Universal Set) - $\{ \forall, < \}$

④ $\{ \exists, \geq \} \Leftrightarrow$ (Universal Set) - $\{ \forall, \leq = \}$

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

\rightarrow All \Leftrightarrow not $<=$ Some

\rightarrow Some \Leftrightarrow not $<$ all

\rightarrow Some \Leftrightarrow not $<=$ all

$R(\underline{P}, R1, R2, R3) \quad S(\underline{P}, S1, S2)$

Ques) Which one of the following Query are equal

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

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

3) $\pi_p(\pi_{P\bar{O}}(R) \cap \pi_{P\bar{O}}(S))$

4) $\pi_p(\pi_{P\bar{O}}(R) - (\pi_{P\bar{O}}(R) - \pi_{P\bar{O}}(S)))$

1) $\pi_p(\sigma_{R \times S})$

$R \cdot P = S \cdot P$

$R \cdot O = S \cdot O$

R		S	
P	O	P	O
1	2	1	2
2	3	2	4
3	3	4	3

2) $\pi_{RP}(\sigma_{\pi_p(R) \times \pi_p(S)})$

$R \cdot P = S \cdot P$

P	1	2	3
1	2	3	1

24/08/2011

SOLI - (Structured Query Language) :-

Sublanguage of SOLI:-

1) DDL (Data Definition Language)

→ modify the schemas of the DB or to manipulate the table, not the content data of the table.

a) Create

b) Drop : (Drop, Table Supplies)

(Entire schema is deleted or we can say the entire table is deleted)

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.

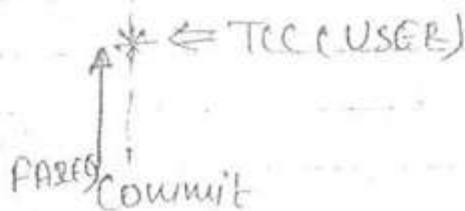
3) DCL (Data Control Language) -

a) Roll back / Abort

b) Commit

c) Transaction consistency (weak point)

Transaction begin



4) Data Query Language -

Used to retrieve data from DB.

- a) Select,
- b) From,
- c) Where,

Query 1

SELECT DISTINCT A₁, A₂, ..., A_N
 FROM R₁, R₂, ..., R_M

Where P

* To eliminate duplicate tuple from result use (DISTINCT)

* whatever the data is comes in From is equal to (R₁ × R₂ × ... × R_M)

* whatever the condition is selected is comes under where

* whatever the attribute specified in the select is equal to the projection.

$\pi_{A_1, A_2, \dots, A_N}^P (R_1 \times R_2 \times \dots \times R_M) \rightarrow$ By default it eliminates the duplicate tuples in the result.

* (SELECT DISTINCT) Equal to projection (π).

* SELECT & FROM clause are mandatory clause with out using these clause we can't create any SQL query.

[5] SELECT [DISTINCT] A₁, A₂, ..., A_N

[6] FROM R₁, R₂, ..., R_M

[7] [WHERE P]

[8] [GROUP BY {Attribute}]

[9] [HAVING condition]]

[10] [ORDER BY {Attribute} [DESC]]

[1] FROM Clause - Cross Product of listed Relations in FROM clause.

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

[3] GROUP BY Clause - Used to group the results (table) based on specified attribute.

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

Ques :- Group by (Branch) ! It group the entire table based on Branch.

O/P:-

Sid	Mark	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 checks for null value
if count is based on entire table it counts all the tuple of the table.

$$\text{Count}(\text{mark}) = 5$$

$$\text{Count}(\ast) = 6$$

$$\text{Count}(\text{DISTINCT marks}) = 4$$

$$\text{SUM}(\text{marks}) = 310$$

$$\text{SUM}(\text{DISTINCT marks}) = 260$$

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

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

* Arithmetic operation with Null results NULL

i.e. $\text{NULL} + 5 = \text{NULL}$

Ques:-

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 is the O/P of the Select Statement.

Ques:-

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

* (AS) (rename operators) :-

Rename for either

attribute or Table.

* "Space" is also taken as rename operator in SQL

Ques 1:- Select min(marks) AS minmarks / (marks) from S

O/P:-

minmarks
50

Ques 2:-

Syntax error

Select Branch, min(marks)

from student.

O/P:- CS 50

It is not able to display O/P in tabular format. So, it cannot be displayed by SQL. So, Syntax Error

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

Query- Select Branch, min(marks)
FROM Student

GROUP BY (Branch).

O/P- CS 50 Here min(marks) is applied on each group of the table.
IT 50
EC 80

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

Query- Select min(marks) O/P- 50
FROM Student 50
GROUP BY (Branch) 80

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

Query- Select Sid, marks O/P
① FROM Student 51 50
② GROUP BY (Branch) 54 70
52 60
53 50
55 80

Query- Select Sid, Avg(marks)
FROM Student } If is not
GROUP BY (Branch) } allowed

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

Syntactically allowed

Query1- Select Sid, ^{branch}marks, Avg(marks)
from Student
GROUP BY (Sid, branch)

Allowed

⇒ If group By clause exists aggregation alone
for every group.

Query1- 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 subquery is written within any 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 50.

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

O/P - Avg SS1	50	CS	1st group	Satisfy the Having Clause condition
60	54	70	CS	
80	55	80	ES	

Example - Employee (name, sex, salary, cname)
SQL Query -

```
① Select DeptName  
② From Employee  
③ Where Sex = 'M'  
④ Group By cname  
⑤ Having Avg(sal) > {Select Avg(salary)  
From Emp  
Company avg salary}
```

What this SQL Query Returns?

"Where clause" condition is applied by "Every table"
"Having clause" condition is applied for every group.

Result

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 column.

[6] DISTINCT Eliminate duplicate tuple from select.

Select { equal to
DISTINCT } RA operators Π (projection).

[7] ORDER By :- default Ascending Order

(If is used
to print the
selected O/P
in the order.)
DESC! for descending order.

SET OPERATORS-

UNION / UNION ALL

INTERSECT / INTERSECT ALL

MINUS (OR) EXCEPT / MINUS ALL

$\pi_{\text{pid}} \left(\sigma_{\text{Color} = \text{Red}}(\text{catalog}) \times \sigma_{\text{Color} = \text{Green}}(\text{parts}) \right) \cup \pi_{\text{pid}} \left(\sigma_{\text{Color} = \text{Red}}(\text{catalog}) \times \sigma_{\text{Color} = \text{Green}}(\text{parts}) \right)$
 C.Pid = P.Pid C.Cid = P.Pid C.Cid > P.Pid

Query 1 - Retrieve 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)

UNION : discards duplicates in Result.

Let us assume

O/P:

1st query

2nd query

	A	B	C
P	A		
P	A		
P	A		
P	B		
P	B		
P	S		
C		V	C
C		V	D
C		V	E

A

B

C

D

E

	A	B
P	A	
P	B	
P	C	

A

B

C

* 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 clauses { ANY; EVERY; }

Text

FROM Student

GROUP BY (branch)

HAVING marks < ANY 5;

HAVING marks > EVERY 20

It is used to select or discard the entire group only.

[Q4]

$$\pi_{\text{new}} \left(\pi_{\text{S1d}} \left(\pi_{\text{S1c}} \left(\sigma_{\text{Parts}} \left(\pi_{\text{P1d}} \left(\pi_{\text{P1c}} \left(\text{col} = \text{Rel} \right) \right) \right) \right) \right) \right) \bowtie \pi_{\text{S1dS1c}} \left(\pi_{\text{Catalog}} \right) \bowtie \pi_{\text{S1dS1c}} \left(\pi_{\text{Supplies}} \right)$$

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

Check for Commutativity-

1) π (Projection) = \times

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

2) σ (Selection) = \checkmark

$$\sigma_{C2} \left(\sigma_{C1} (R) \right) = \sigma_{C1} \left(\sigma_{C2} (R) \right) = \sigma (R) = \left(\sigma_{C1} (R) \right) \cap \left(\sigma_{C2} (R) \right)$$

3) Cross Product = \checkmark

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

4) Union (U) :- $R_1 \cup R_2 = R_2 \cup R_1$ ✓

5) Intersection (\cap) :- ✓

$R_1 \cap R_2$

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 Suppliers \bowtie catalog.

Suppliers		Catalog	
Sid		Sid	Prod
S1		S1	S1
S2		S1	S2
S3		S1	S3
..	
S200		S1	S200

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

Min no. of tuples in sup \bowtie catalog = 100 (because Sid is foreign key of suppliers Sid so, Sid reference minimum 100 tuples is necessary.)

Suppliers Table

* $M \cup N = M \cup N$

M	$\{ A, A, B, B, C, D \}$	UNION ALL	$\{ A, A, B, C, D, D, E \}$	$M \cup N$
				$\{ A, A, B, B, C, D, D, E \}$

UNION ALL ~~TUPLES~~ Duplicates all the tuple of the R/P.

* INTERSECT-

Discards duplicates in result

M	INTERSECT	N
$\{ A, A, B, B, C, D \}$		$\{ A, B, C, D, E \}$

* Common tuple of both the relation with duplicate tuples.

* INTERSECT ALL-

* Maximum no. of common in both the set.

M	$\{ A, A, B, B, C, D \}$	INTERSECT ALL	N	$\{ A, B, C, D, E \}$
				$\{ A, B, C, D \}$

MINUS-

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

A		A	=	
A		A		
A		B		
B	MINUS	C		
B		D		
C		D		
D		E		
A				
A				

MINUS ALL - No. of repetitions in the first relation - no. of
duplicates in 2nd relation.

A		A	=	A
A		A		B
A	MINUS	C		C
B	ALL	D		D
B		D		E
C				
D				
A				
A				

Nested Queries:-

SQL supports query inside query

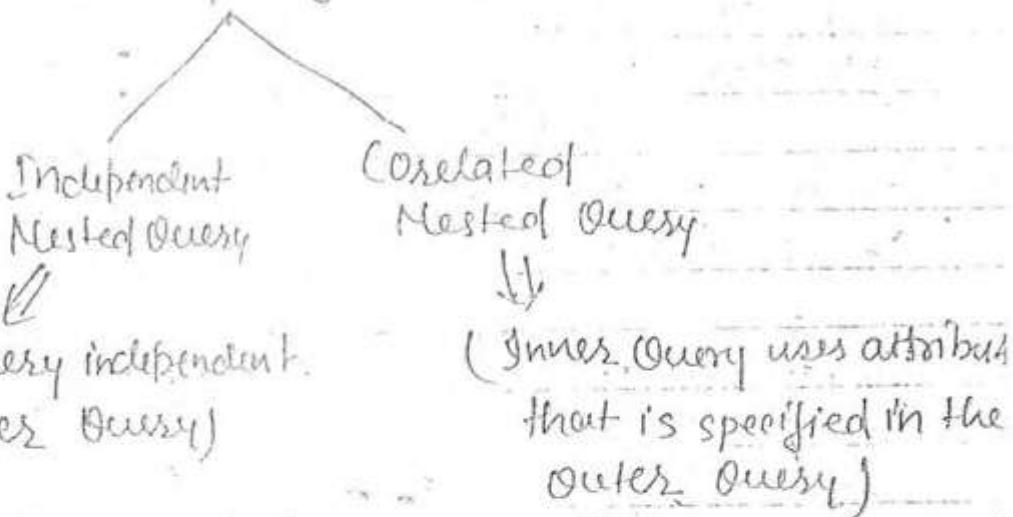
- a) From clause
- b) OR WHERE clause
- c) OR HAVING clause

Example:-

```

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

There are two types of nested query:-



Example:- Retrieve Sid of the supplier 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 (

//

Sid	Pid
S1	P1
S2	P2
S1	P3
S3	P5

) SELECT Pid
FROM Parts
where color = Red

O/P:

S1
S1
S3

Pid
P1
P3
P5

SQl Supports

IN / NOT clause

IN clause - Used compare given tuple value exists in the set of value or 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

//

where Spid = P. Pid

FROM Parts

where color = Red) P

Ques:- $\text{Enpl}(\text{Eno}, \text{Ename}, \text{rating})$

$\text{Dependent}(\underline{\text{Eno}}, \underline{\text{Dname}}, \text{age})$

Retriene employee names who have no dependent.

[a]

SELECT Ename
FROM EMP E, Dependent D
Where E.Eno \neq D.Eno

[b]

FROM Employee
where Eno NOT IN (SELECT Eno
FROM Dependent)

[c] 01 correct but not 02

[d] 02 " " " 01

[e] 01 & 02 correct

[f] 01 & 02 incorrect

Example:-

Eno
E1
E2
E3
E4

Eno	Dname
E1	A
E1	B
E2	A

O/P 01 [E1
E2
E3
E4]

[02] [E3
E4]

Set Comparison operators:-

$x \in \{ \dots \} \rightarrow x$ is in given set or not
if x is in set then it True
Otherwise False.

\Rightarrow op ANY

\Rightarrow op ALL

\Rightarrow op $<, \leq, >, \geq, =, \neq$
(not equal to)

Example:-

$x > \text{ANY} \{ \dots \} \rightarrow y$

returns True if x greater than
any value in the set.

$x > \text{All} \{ \dots \} \rightarrow y$

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 supply
some Red part.

~~Select S.Sid
from SUPPLIER
where Select rating~~

~~FROM SUPPLIER S, Catalog C, Parts P~~

~~where S.Sid = P.Sid and~~

~~P.Pid = C.Pid and~~

~~W.Cat = Red)~~

Select Sid

FROM SUPPLIER

WHERE rating > ANY (Select rating

From SUPPLIERS.CatalogC.Parts

Where S.Pid = C.Sid and

P.Pid = C.Pid and

Colors = RED)

Query: Retrive Sid of the Suppliers whose group average rating is more than the suppliers rating of every group of rating. Age, Manager

Query: Retrive Sid of the Supplier whose age is greater than all

FROM

Query Sid's of suppliers who are highest age.

(Supplier(Sid, Sname, Age))

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 only creates integrity violation but retrieval of any data does not create any integrity violation.

Correlated Nested Queries

Example:-

Select Sid, Sname

FROM Suppliers S

where EXISTS (Select * FROM Catalog C

Where C.Sid = S.Sid)

Suppliers:

Sid
✓ S1
✓ S2
X S3
X S4

Sid	Pref.	Cost
S1	P1	
S1	P2	
S2	P3	

Retrieval Sid, Sname
of the suppliers
who supply some
parts.

* It works as nested join query i.e. we take one tuple from Suppliers and execute the inner query for it. If it is not empty then that tuple of supplier is selected if the inner query return EXISTS (class):- false then supplier tuple is not selected.

Retrieves true if inner query results non-empty.

Note
Ques)

Student (Sid , Sname , marks)

Enrolled (Sid Cid)

No Foreign key, No NULL value in the DB.

(Query) Select Sid

FROM student

Where Sid in (Select sid

from Enrolled)

S1

S2



[Query 2] Select Sid
 FROM Enrolled
 Where Sid in (Select Sid
 from student) S1
 S1
 S2

[Query 4]

```
    Select S.Sid          S1  
        FROM Student S  
    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.

<u>Student</u>
S1
S2
S3
S4

<u>Enrollees</u>	<u>Sid Cof</u>
S1	C1
S1	C2
S2	
S5	

Ques) movie(id, title, yr)
actors(id, name)
Casting(movieid , actorid)

Use SQL Query to return titles of the all 1959 Mon
films.

Solⁿ:-

Select Titles
from Movie, actors a, Casting C
where m.id = movieid and
a.id = actorid and
yr = 1959
name = 'Monroe'

Ques) 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) " " " " fine " "

Title	Price
T1	20
T2	30
T3	20
T4	60

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

Ques) A 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 are not correct both Q1 results correct for some database but both are incorrect

d) Q1 & Q2 returns identical rowset for some database

Output

Cust. Balance	O/P	Rank
C1 10	C5	1
C2 20	C4	1
C3 20	C6	1
C4 30	C3	4
C5 30	C2	4
C6 30	C1	6

C1 10	C1 10	✓ 10 10	✓
C2 20	X C2 20	✓ 10 20 ✓	C1 10
C3 20	C3 20	✓ 10 30 ✓	C1 10 3
C4 30	C4 30	X 20 10	C1 10
		C2 ✓ 20 20	C2 20
		C2 ✓ 20 20	C2 20 3
		C2 ✓ 20 30 ✓	C2 20
		X 20 10	C2 20
		C3 ✓ 20 20	C3 20 3
		C3 ✓ 20 20	C3 20
		X 30 10	C4 30
		X 30 20	
		X 30 20	
		C4 ✓ 30	

⇒ Query 1 is correct only when balance is different
Or distinct but if any two customer of same
balance then query 1 result is incorrect.

⇒ Query 2 is correct except for the 1st rank.

* EXISTS "return True" when inner condition is
 "non empty".
 Similarly

"NOT EXISTS" return True when inner condition
 "false" or "empty". Or return
 False when it is non empty.

Query:-

Select S.Sid

FROM Suppliers S

where NOT EXISTS (Select P.Pid

FROM Parts P

where NOT EXISTS (SELECT C.Sid

FROM Catalog C

where (C.Sref = S.Sid OR

C.Pid = P.Pid))

for (i = 1 to n)

for (j = 1 to n)

for (k = 1 to n)

Suppliers (Sid, Sname)

Parts (Pid, Part, Color)

Catalog (Sid, Pid,

Suppliers	Catalog	Parts
Sid	Sid Pid	Pid
S1	S1 P1	P1
S2	S2 P1	P2
S3	S1 P2	P3
	S3 P2	

(Sid) — { all parts supplied by all suppliers } (enroll)

O/p:- It returns all the suppliers

* This above program is just work as division operator works.

25/08/2011

SOL:-

Supports pattern matching of strings.

"%" \Rightarrow used to denote 0 or more characters

"_" \Rightarrow used to denote Any single character

Example "Names" Starts with R

'R.%'

\Rightarrow Names Starts with 'D' Ends with 'A' and at least 5 characters

'D_ _ _ % A'

(Qn) If we want to retrieve 'MADE%EASY' string.

We use Escape characters ('\\')

pattern characters as like normal character used treat

\Rightarrow 'MADE\\%EASY'

Query Name Starts with 'D' Ends with 'A' and at least 5 characters.

Select *

From Student

where Sname LIKE 'D_ _ _ % A'

LIKE / NOT LIKE -

Used to compare with pattern string.
NOT LIKE 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	Fname	PPno
E1	A	P1
E2	A	NULL
E3	B	P2

Query - Retrieve Employee who have no passport.

Select *
FROM Employee
Where PPno ~~IS~~ NULL

} It is not
compare with
null value

So we use IS / IS NOT clause

⇒ IS / IS NOT - Compare with NULL Values.

Select *
From Employee
Where PPno IS NULL

- There are three truth value in DBMS

→ True \Rightarrow If it is '0'

→ False \Rightarrow If it is '1'

→ UNKNOWN: Comparison with Null value is treated as Unknown.

then it is $\frac{1}{2}$ ($\frac{1}{2}$ is not true, not false,
 $\frac{1}{2}$ is between true
false)

X	OR	Y	Z = max(x, y)	X	Y	Z = min(x, y)
1	1	1	1	1	1	1
1	0	1	1	1	0	0
0	1	1	1	0	1	0
0	0	0	0	0	0	0
0	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	0	$\frac{1}{2}$	0
1	$\frac{1}{2}$	1	1	1	$\frac{1}{2}$	$\frac{1}{2}$

⇒	OR	T	F	U	AND	T	F	U
	T	T	T	T	T	T	F	U
	F	T	F	U	F	F	F	F
	U	T	U	U	U	U	F	U

⇒ NOT UNKNOWN = UNKNOWN

Ques) Relation can contain NULL values. Suppose all comparison wif NULL value are treated false.

which of the following pairs is not equivalent:

a) $x = 5$ $\text{not}(\text{not}(x=5))$

b) $x = 5$ $x > 4 \text{ and } x < 6$

where x is an integer

c) $x \neq 5$ $\text{not}(x=5)$

d) None

If we take Table as

X
NULL
5
6

check one by one in
L.R.S & R.R.S if
the above are true
we obtain the values

a) $x = 5$ $\boxed{5}$ $\text{not}(\text{not}(x=5))$ $\boxed{5}$ x type
val

b) $x \neq 5$ $\boxed{5}$ $x > 4 \text{ and } x < 6$ $\boxed{5}$ x type
val
fals

c) $x \neq 5$ $\boxed{\begin{matrix} x \\ 6 \end{matrix}}$ $\text{not}(x=5)$ $\boxed{\begin{matrix} \text{NULL} \\ 6 \end{matrix}}$

Transaction and Concurrency Control

Transaction:-

Set of logically related operations used to perform unit of work.

Example:-

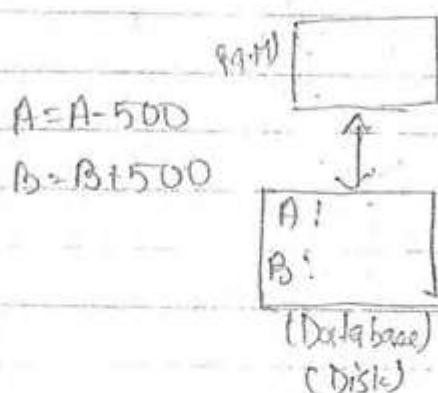
T1: Transfer 500 Rs from Account A to Account B.

Read (A) :-

(or Database) Accessing the data item A from secondary memory to programmed variable.

Write (A) :-

Updates data items A into the database.



begin Trans

R(A)

A = A - 500

W(A)

R(B)

B = B + 500

W(B)

Commit

R(A)

W(A)

R(B)

W(B)

Commit

I/O operations - R(A), R(B), W(A), W(B)

Memory operations - A - 500, B + 500

Commit :- Transaction Executed Successfully.

- * 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 bcoz 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. If it in sequence order not arbitrary.

Example :-

begin Trans.

if A = 1000
500
1000 rollback
A = A - 500

B = 2000
W(B)
R(B)
Failed
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 transactional operations. Used to perform Rollbacking.

⇒ The main purpose of transaction log is Rollbacking.

Undo : operation :-

Database is updated by old values of the log file.

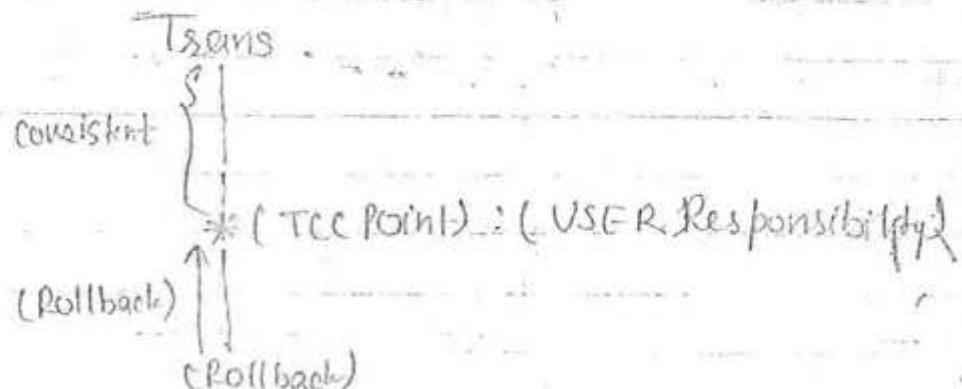
Redo : operation :-

Updating the database by new values.

Database updated with New Value.

⇒ Log file retained in the DB until transaction committed or rollbacking completed, then 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 user 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)

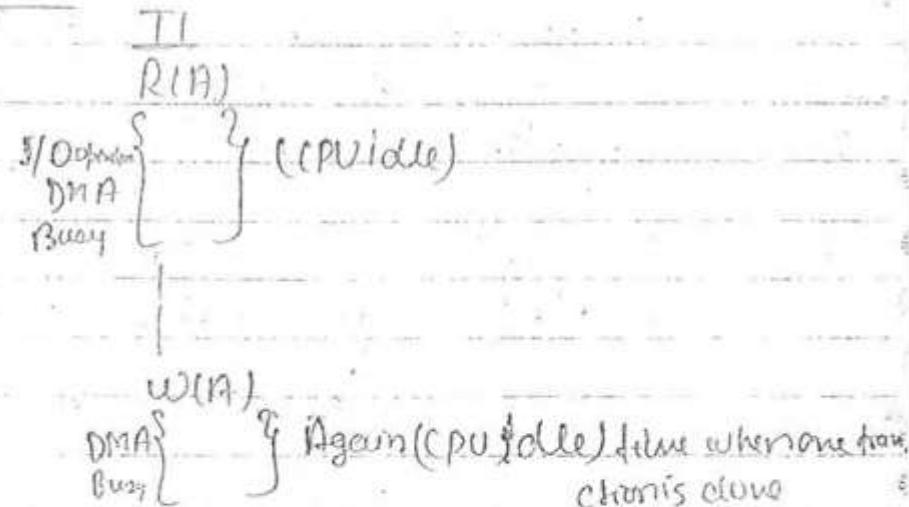
Serial schedule:-

T1	T2	T1	T2
R(A)			R(A) : 1000
W(A)			R(B) : 2000
R(B)		R(A)	3000
W(B)		W(A)	
	R(A): 500	R(B)	
	R(B): 2500	W(B)	
T1:T2	3000	T2:T1	

⇒ If n Trans then
n! serial schedule

⇒ All serial schedules are consistent.

⇒ Cess throughput-



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 transaction

T1	T2	T1	T2
R(A)		R(A)	
W(A)		W(A)	
	R(A) 500		R(A) 500
R(B)			R(B) 2000
W(B)		P(B)	2500
	R(B) 2500		W(B)
	<u>3000</u>		

[consistent] (equal to T1:T2 serial)

Advantage - 1) More throughput

2) Less Response time

3) Better Resource Utilization.

[Inconsistent] $T_1 \rightarrow T_2$ \checkmark
 $T_2 \rightarrow T_1$ \times

Disadvantages

Concurrent execution may result in inconsistency

(Qn) How many total schedules are possible w.r.t T1 & T2 Transaction?

$$\frac{6!}{4!2!}$$

$$\frac{6!}{4!2!}$$

(By Permutation i.e. $6C_4 * 2C_2$)

Assume T_1 & T_2 are m, n operations each. Total possible schedules.

$$= \frac{(m+n)!}{m!n!}$$

Assume T_1, T_2, T_3 are m, n, p operations each. Then Total possible schedule =

$$\frac{(m+n+p)!}{m!n!p!}$$

\Rightarrow if concurrent execution not equal to any serial then it's Inconsistent.

Ex:-

i) Isolation :-

Concurrent execution of two or more transaction should be equal to any serial schedule then it's consistent i.e. we called it as (Serializable Schedule).

then it's caretaken by

\Rightarrow A Concurrency control management component

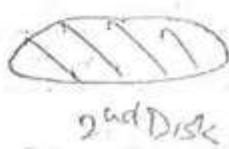
D: Durability

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

[T1 WR Problem] - [Read After Write Problem]

T1	T2
W(A)	

R(A): Uncommitted Read. (R(A) reads data when W(A) is not committed to transaction)

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

T1	T2
R(A)	

||
||
||

T1	T2
1000 R(A)	R(A) 2000
500 W(A)	R(B)
	R(B) 2000
2000 R(B)	
2500 W(B)	

consistent.

We can't know that which problem is exist in this transaction due to inconsistency. i.e. Problem is not just found after seeing the transaction. Firstly we have to check if that transaction is consistent then no problem, but if the transaction is not consistent then after we check, due to which problem it is inconsistent. i.e. problem is checked after checking the transaction is inconsistent.

[2.7 RW Problems [Write After Read Problem]]

T1	T2
R(A)	
	W(A)
	R(B)

T1 performed Read(A)
before commit / rollback of
T1, finds 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 $A = 10$ g'g (dueto
 ↑ R/W prot
 dueto dueto
 transaction trans
 T2 ... T1

Inconsistent (R/W Problem Exist)

A = 10

T1
R(A)

T2

A = 9

R(A)

W(A)

It is called temporary read problem being either there is no any updation in even after some time, if it reads then it get the other value previous. Because there may any other transaction T2 will updated the previous value (Temporary Read Problem) read by the Transaction (T1)

OR

(Phantom phenomena problem)

[S] WW Problem :- [Write After Write Problem]

T1
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 Transaction T1 execution 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)

T1	T2
w(A)	
	w(A)
	w(B)
	w(B)

A = 1000
2000

B = 2000
1000

(Non-Serializable)
Inconsistent
Because of WW Problem

A = 1000
2000

B = 1000
2000

(Serializable)
No concurrent
Execution Problem

* When the transaction is non-serializable then only WR Problem, RW Problem, WW Problem are reasons for non serializable Inconsistency.

* Sometimes WW Problem is LOST UPDATE PROBLEM

⇒ LOST UPDATE PROBLEM -

T1	T2
R(A)	
w(A)	
Rollback	[w(A)]
(FAILED)	

A = ~~0, 10, 20~~ 0

← (lost update problem becos of
T1 Rollback)

* Even if the schedule is serializable lost update problem possible is still possible.

26/08/2011

Problem bcos. 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.

Serializable 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

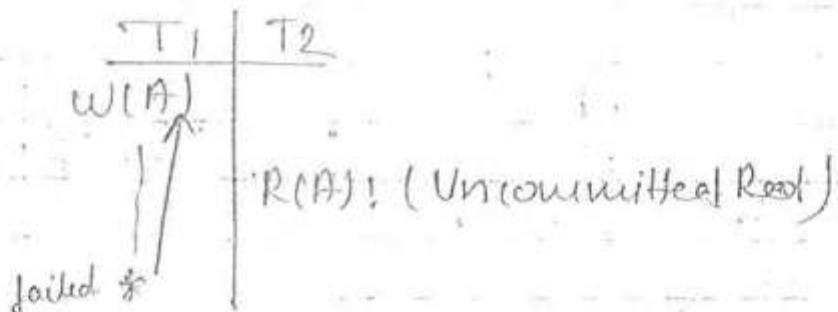
Two classification are possible - Based on -

- * Recoverability
- * Singularity

* 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 depending on other

<u>T1</u>	<u>T2</u>
<u>W(A)</u>	C
↓	↓

NO dependency

b/w Transaction

<u>T1</u>	<u>T2</u>
R(A)	C
↓	↓

<u>T1</u>	<u>T2</u>
W(A)	C
↓	↓

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

[D] Nonrecoverable Schedule -

Rollback of committed transaction.

T1	T2
W(A)	
↑-1	
(Rollback)	
*	
Failed	
	(Nonrecoverable)

T1	T2
W(A)	
	R(A)
	Commit
(Future operation)	Commit
	→ Failed

T1	T2
W(A)	
	R(A)
	Commit
Commit/Fail	

Trans-F: Transaction T2 Ready
 the data item A updated by
 some other transaction T1
 if commit operation
 of T2 before the Commit/fail
 of T1 then schedule is
nonrecoverable.

[i] Recoverable Schedule -

Rollbacking of only uncomm.

transaction.

T1	T2
w(A)	
	R(A)
Commit / Rollback	Commit

T1	T2	T3	T4
w(A)			w(A)
	R(A)	Rollback	R(A)
Commit	Commit	C/R	C/R

If T2 Reads the data item A updated by T1. Then commit of T2 should be delayed until C/R of T1

→ If both the transaction depending on each other then whatever the order of dependency, in each case it is recoverable.

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(B))
	Dependency
	(W(B))
R(B)	
	C1
	C2

(Dependency is recoverable but not present) 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)	
Rollback	W(A)
	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 Rollback :-

T1	T2	T3	T4
W(A)	R(A)		
(rollback)	W(A)	R(A)	
		R(A)	R(A)
(Paused)			

(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 dependencies towards must be roll backed.

⇒ it results
wastage of CPU execution time and I/O operations

[3] Cascadeless Rollbacking Recoverable Schedule

To remove cascading rollbacking problem we use
the cascadeless rollbacking i.e. no transaction is
depending on other transaction.

T1	T2
w(A)	
C/R	
	R(A)

{ Uncommitted Read not allowed
In cascadeless Rollbacking
Recoverable Schedule }

T1	T2	T1	T2	T1	T2
w(A)		R(A)		w(A)	
w(A)		R(A)		R(B)	
w(B)		w(A)		R(C)	
C(B)		w(B)		w(C)	
C1		C1		w(C)	
C2		C2		C1	
(WW Problem)		(RW Problem)		C2	
Is exist		Is exist			

WR Problem not possible here and cascading Rollback are not possible but still WR, 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 avoid simultaneously writes of any two transaction. Due to this WW problem also recoverable removed.

T1	T2
w(A)	
C/R	R(A)/W(A)

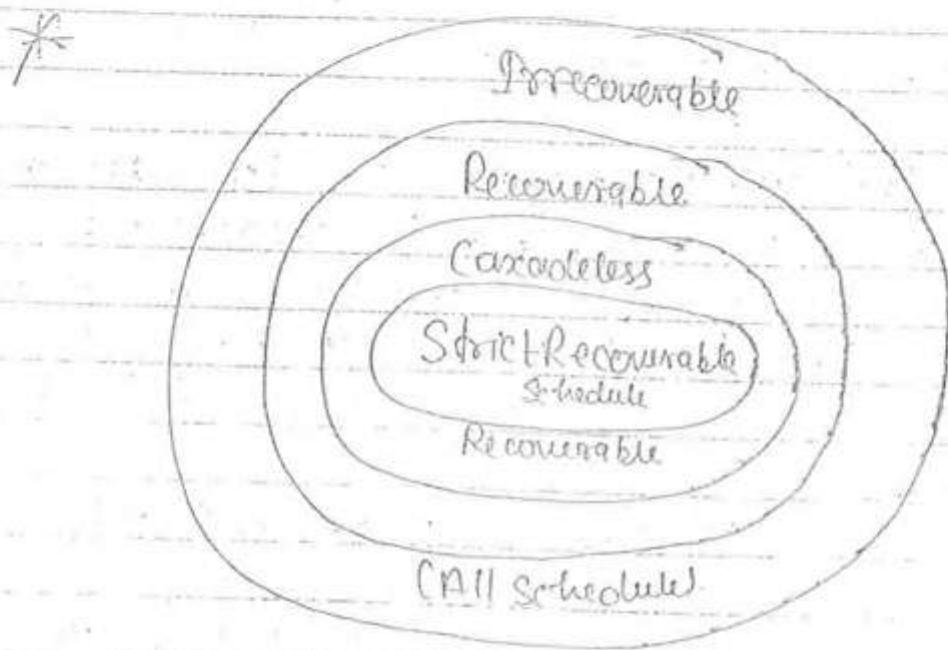
if Trans T1 updates the data item A then some other transaction T2 not allowed to R(A)/W(A) Until C/R of T1.

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 uw, wr ,
 Problem and also last update, Cascading Rollback
 problem.

But rw problem may also exist here.



* Operation details
 $w_1(A) w_2(A) w_2(B) w_1(B) c_1 c_2$
 Transaction No

Ques) SI: $s_1(x) s_2(y) s_1(z) s_3(z) s_3(y) w_1(x) w_3(y) w_2(y)$
 $w_2(z) w_3(y) c_1 c_2 c_3$
 Check which higher recoverability is present
 here.

	T1	T2	T3
1	$s_1(x)$		
2		$s_2(y)$	
3	$s_1(y)$		
4			$s_3(y)$
5			$s_3(y)$
6			$w_3(y)$
7		$s_2(y)$	
8		$w_2(y)$	
9		$w_2(y)$	
10	c1	c2	c3

→ dependency exist

⇒ It is irrecoverable.

(iii) $s_0 w_1(x) w_3(y) (3 s_2(y) w_2(y)) c1 c2$

⇒ Recoverable ✓
Cascadeless ✓
Strict Recoverable ✓

(iv) $s_2 - s_1(x) s_2(y) s_3(x) s_4(y) s_2(y) s_3(y) w_1(x) c1 w_2(y) w_3(y) w_2(y) c3 c2$

$w_1(x)$		
c1		
	$w_2(y)$	
	$w_2(y)$	
	c2	

Cascadeless but
not strict

Ques3: $s_3(x)s_1(x)w_3(x)r_2(x)w_1(y)s_2(y)w_2(y)$ C3 C1 C2

RECOVERABLE ✓

Cascades X

S4: $s_1(x)w_1(x)w_2(x)s_1(x)s_1(y)w_2(y)$ C1 C2

T1	T2	No dependency
R(x)		
w(x)		
	w ₂ (x)	REC ✓
	s ₂ (x)	CAS ✓
s ₁ (y)		Strict X
	w ₂ (y)	
	C1	
	C2	

Ques) T1 operations is T1 T2 operations is T2

R(A)

R₂(A)

w₁(A)

w₂(B)

R₁(B)

C2

w₁(B)

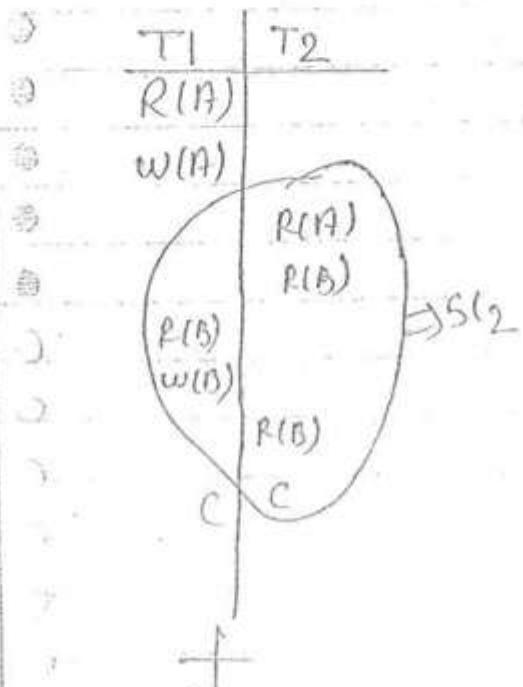
C1

Total no. of schedules: $\frac{8!}{5!} = \frac{8!}{5!3!} = \frac{8 \times 7 \times 6 \times 5!}{5!3!} = 56$

No. of recoverable schedule:-

Example:-

= All schedule - Irrecoverable schedule

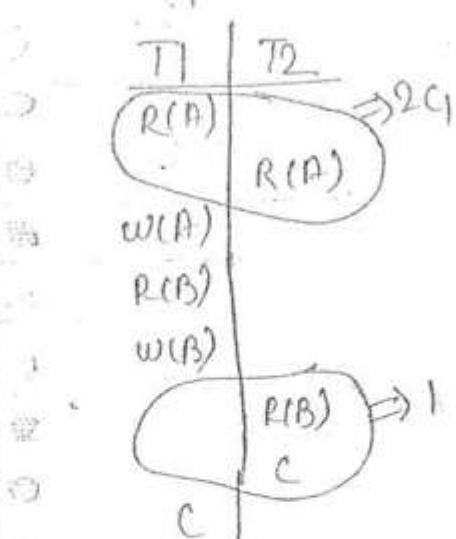


$$= 8c_5 - 5c_2 = 48$$

$$= 8c_5 - 5c_2 - (2c_1 * 1)$$

$$= 46 - 2$$

$$= 44$$



No of Cascadless Recoverable Schedule :-

= (Recoverable Schedule) - (Cascading Rollback)

T1	T2	T1	T2
R(A)		R(B)	
W(A)	R(A)	R(A)	
R(B)	R(B)	R(B)	
W(B)		W(B)	
C		R(B)	
	C		C
		I possibility	

- ⇒ if two schedules are conflict equivalent means that both schedules are logically equivalent.
- ⇒ if schedules are non-conflict equivalent then may or may not be logically equivalent.

Example:-

	T1	T2	T3		T1	T2	T3	
	w(A)				w(A)		w(A)	
A = 16 26 36		w(A)	w(A)					A = 26 16 30
				(S1) A : T3				(S2) A : T3

S1 and S2 are ~~not~~ logically equivalent
bcuz S2 resulted after swapping conflict pairs in S1

So it is not conflict equivalent
even though S1 & S2 are equivalent schedule.

One	T1	T2		T1	T2		
	R(A)			R(A)			
	w(A)			w(A)			
		R(A)			R(A)		
		R(B)			R(B)		
	(R(B))						
	w(B)			w(B)			
			S1			S2	

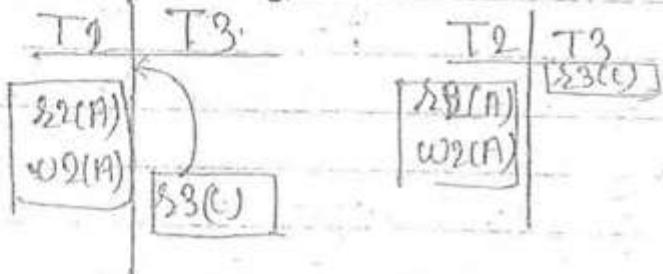
S1 is logically equivalent to S2 bcuz we swapping two consecutive non conflict pair in the schedule

write

S1:	<u>R2(A)</u>	<u>W2(A)</u>	<u>R3(C)</u>	<u>W2(B)</u>	<u>W3(A)</u>	<u>W3(C)</u>	<u>R1(A)</u>	<u>R1(B)</u>	<u>W1(A)</u>	<u>W1(B)</u>
S2:	<u>R3(C)</u>	<u>R2(A)</u>	<u>W2(A)</u>	<u>W2(B)</u>	<u>W3(A)</u>	<u>R1(A)</u>	<u>R1(B)</u>	<u>W1(A)</u>	<u>W1(B)</u>	<u>W3(C)</u>

S3: R2(A) R3(C) W3(A) W2(A) W2(B) W3(C) S1(A)
R1(B) W1(A) W1(B)

Which one of them are conflict equivalent schedules

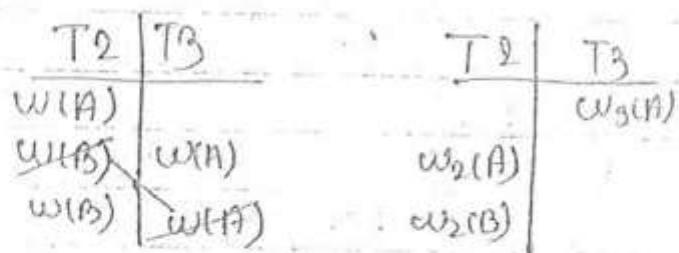


\Rightarrow S1 & S2 are conflict equivalent

S2: R3(C) R2(A) W2(A) W2(B) W3(A) R1(A) R1(B) W1(A) W1(B) W3(C)

<u>R2(A)</u>	<u>R3(C)</u>	<u>W2(A)</u>	<u>W2(B)</u>	<u>W3(A)</u>	<u>R1(A)</u>	<u>R1(B)</u>	<u>W1(A)</u>	<u>W1(B)</u>	<u>W3(C)</u>
<u>R2(A)</u>	<u>R3(C)</u>	<u>W3(A)</u>	<u>W2(A)</u>	<u>W2(B)</u>	<u>W3(C)</u>	<u>S1(A)</u>	<u>S1(B)</u>	<u>W1(A)</u>	<u>W1(B)</u>

conflict pair



Conflict Serializable Schedule-

Schedule is said to be

Conflict Serializable schedule only if there should be conflict Equivalent Serial Schedule.

Q) Check the given schedule is conflict equivalent i.e.

T1	T2	T1	T2
R(A)		R(A)	
	W(A)		W(A)
$T_1 \rightarrow T_2 X$ (Not possible)		R(A)	
		R(B)	
		R(B)	
		R(B)	
			W(B)

between conflict pairs
W(A) & R(A) and between
R(B) & W(B). $T_1 \rightarrow T_2$ &
 $T_2 \rightarrow T_1$ X (Not possible)

only if aware about swap this pair is alone but not possible bcz there is a conflict pair

(T1 followed by T2) is $T_1 \rightarrow T_2 X$ not possible
(T2 followed by T1) is also not possible i.e. $T_2 \rightarrow T_1 X$.

Ques).

T1	T2	T1	T2
R(A)		R(A)	
W(A)			
$T_1 \rightarrow T_2 X$ $W(B) \& R(B)$	R(A)		
$T_1 \rightarrow T_2$	R(B)		
so it's conflict	W(B)		
equivalent because there is no any cycle		R(B)	

between conflict pairs
W(A) & R(A) and between
R(B) & W(B). bcoz equivalent serial schedule is possible

Conflict Serializable Schedule

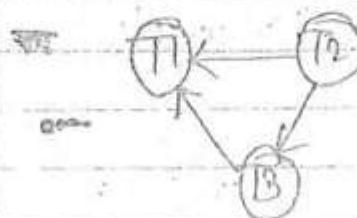
(Ques) $s_2(A) w_2(A) s_3(C) w_2(B) w_3(A) w_3(C) R_1(A) R_1(B) w_1(A) w_1(B)$

$T_2 \rightarrow T_3$ $s_2(A) \& w_3(A)$

$T_2 \rightarrow T_1$ $w_2(A) \& R_1(A)$

$T_2 \rightarrow T_1$ $w_2(B) \& R_1(B)$

$T_3 \rightarrow T_1$ $w_3(A) \& R_1(A)$



$T_2 \rightarrow (T_1 \ T_3)$

$T_2 \rightarrow T_3 \rightarrow T_1$ (conflict)

Serializable
Schedule

(Ques) $s_2(A) s_3(C) w_3(A) w_2(A) w_2(B) w_3(C) R_1(A) R_1(B) w_1(A) w_1(B)$

$s_2(A) \& w_3(A)$ $2 \rightarrow 3$

$w_3(A) \& s_1(A)$ $3 \rightarrow 1$

$w_3(A) \& w_2(A)$ $3 \rightarrow 2$

$2 \rightarrow 3 \& 3 \rightarrow 2$ (non-conflict Serializable
Schedule)

Precedence Graphs-

Graph $G = (V, E)$

Vertices (V): Transactions of the schedule.

Edges (E):



- we can draw edge between T_i to T_j only if there exist conflict pair between T_i & T_j such that T_i precedes then T_j .

$\psi_{ij} : R(A) \rightarrow W(A)$

$$T_j : w(A) \rightarrow T_j : R(A)$$

$T_i : w(A) \cdot T_j : w(A)$

Testing Condition

If Precedence graph is Acyclic \Rightarrow Conflict

Serializability

⇒ Equivalent Serial Schedule: based on topological
(sequency) Order of acyclic Precedence graph.

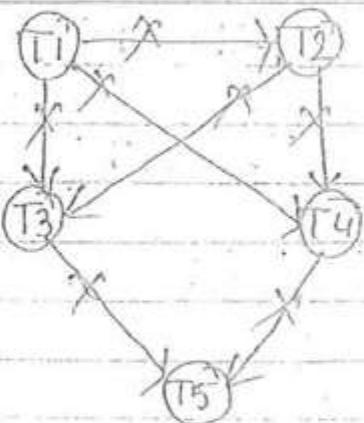
下

27/08/11

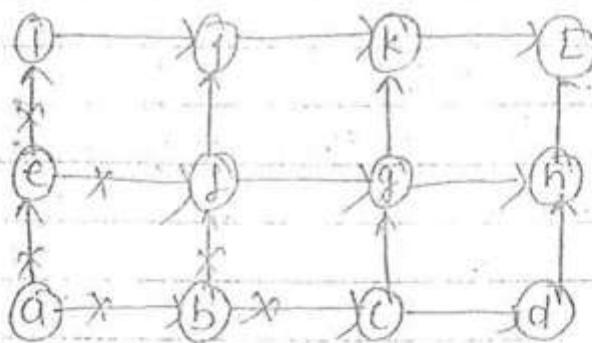
Topological Oscles

[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.



a: b : d : c : d : h :

a: b : c :

a: e : d :

a: e : i :

a: b : j : c : g : k :

a: b : d : 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 side change in old schedule. Then that operation is non conflict pair.

[a] T₁: R(A) T₂: R(A) : Non conflict Pair

$$\left(\begin{array}{c|c} T_1 & T_2 \\ \hline R(A) & \\ \end{array} \right) = \left(\begin{array}{c|c} T_1 & T_2 \\ \hline & R(A) \\ \end{array} \right)$$

S₁ S₂, becoz $\Rightarrow S_1 = S_2$

[b] T₁: R(A) T₂: W(A) : Conflict Pair

$$\left(\begin{array}{c|c} T_1 & T_2 \\ \hline R(A) & \\ \end{array} \right), \quad \left(\begin{array}{c|c} T_1 & T_2 \\ \hline & W(A) \\ \end{array} \right)$$

S₁ S₂ $\Rightarrow S_1 \neq S_2$

L

[C]	T1: W(A)	T2: R(A)	: Conflict Pairs
	T1 T2	T1 T2	
	W(A)	R(A)	

[d] T1: W(A) T2: W(A) : Conflict Pairs

T1	T2	T1	T2
W(A)			W(A)
	W(A)	W(A)	

Final update of A is done
by T2

Final update of A is
done by T1

$$\Rightarrow S1 \neq S2$$

[e] T1: R(A)/W(A) T2: R(B)/W(B) : Non Conflict

Conflict Pairs:-

Pairs of operations said to be
conflict only if

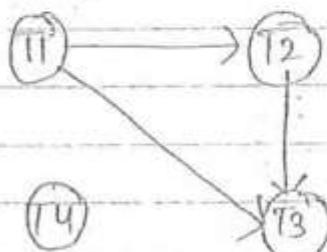
- 1) At least one write operation
- and 2) On same data item
- and 3) b/w different transaction

Conflict Equivalent Schedule:-

S1 & S2 said to be
conflict equivalent only if S2 resulted after
swapping consecutive non conflict pairs from
S1.

Ques) Identify Conflict Equivalent Serial Schedule ~

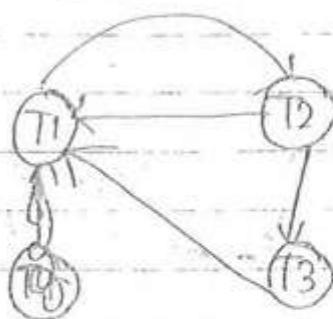
[1] $r_1(A)r_2(A)r_3(A)r_4(A)w_1(B)w_2(B)w_3(B)$



$T_4 : T_1 : T_2 : T_3$
 $T_1 : T_4 : T_2 : T_3$
 $T_1 : T_2 : T_4 : T_3$
 $T_1 : T_2 : T_3 : T_4$

conflict
Equivalent
Serial Schedule

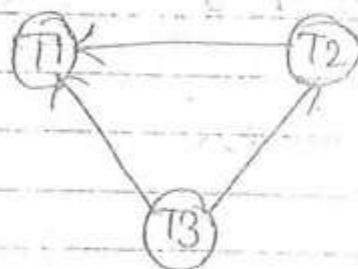
[2] $r_2(y) \xrightarrow{①} r_2(y)w_2(y)r_3(y) \xrightarrow{②} r_3(y)r_1(x)w_1(x)w_3(y)w_2(x)$
 $r_1(y) \xrightarrow{③} w_1(y)w_2(x)$



Non Conflict
Equivalent Schedule

[3] $\delta_1(A) \delta_2(A) \delta_3(B) w_1(A) \delta_3(C) \delta_2(B) w_2(B) w_1(C)$

$\delta_2(A) \rightarrow w_1(A)$
 $\delta_2(B) \rightarrow w_2(B)$
 $\delta_3(C) \rightarrow w_1(C)$

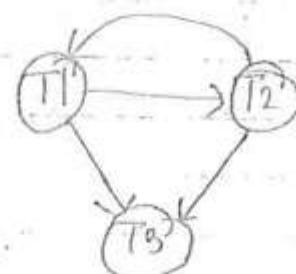


conflict Serializable
schedule

T3 : T2 : T1

* If conflict schedule is serializable then only we can say that it is conflict serializable schedule.

Ques	T1	T2	T3
S1	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)

$S1 = S2$ i.e.
 schedule are equal but
 not conflict equivalent.

S_2 is equivalent serial schedule of S_1 , so, it is equivalent serial schedule \Rightarrow (Serializable Schedule).

\Rightarrow if (Acyclic Precedence graph).

{ conflict serializable

{ Serializable

} If it is sufficient to say that schedule is serializable

else

{ Not conflict Serializable Schedule

may / may not be Serializable

} // Not Necessary Condition to say non serializable

* It is only sufficient but not necessary.



Schedule

View
Serializable
Schedule
Condition

Serializable

YES

* If it is sufficient &

also necessary

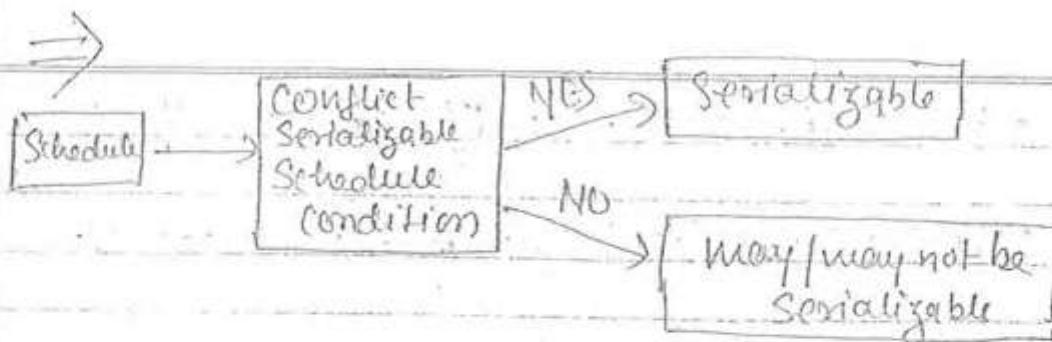
condition.

NO

Non-Serializable

Exponential Time complexity

(NP-Complete problem)



\Rightarrow Polynomial Time Complexity $O(n^2)$

ব্যাপ্তি

View Serializable -

S_1 & S_2 are view equivalent only if following condition are true:-

- [1] Initial Reads of S_1 & S_2 should be same.

(Initial DB) $\downarrow T_1$
 $\downarrow R(A)$

(S1)

(Initial DB) $\downarrow \frac{T_1'}{R_A}$

(S2)

<u>Example:-</u>	T1	T2	T3
	R(A)		
		<u>R(A)</u>	
			W(A)

(S1)

	T1	T2	T3
	R(A)		
	<u>R(A)</u>		
			W(A)

(S2)

S_1 & S_2 are not equivalent bcoz in S_1 $R_1(A)$ & $R_2(A)$ initially read data item from Data Base but in S_2 $R_2(A)$ read the data item updated by $T_{12}(A)$. So S_1 is not equivalent $S_1 \neq S_2$

→ Write, Read Sequence should be same in the S1 & S2 schedule

[2]	T_i	T_j	T_i	T_j	schedule
		$w(A)$		$w(A)$	
	$R(A)$		$R(A)$		

(S1)

Example:-	T_1	T_2	T_3	T_1	T_2	T_3
	$w(A)$				$w(A)$	
		$w(A)$		$w(A)$		
			$R(A)$			$R(A)$

(S1)

(S2)

$\Rightarrow S1 \neq S2$ becoz here $R3(A)$ in S1
 $R3(A)$ read the dataitem updated by $w2(A)$

But in S2 $R3(A)$ read the dataitem updated by $w1(A)$.

∴ $S1 \neq S2$ because write & Read sequence should not same.

Final Write should be same.

	T_i	T_i
	$w(A)$	$w(A)$
	↑ (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)

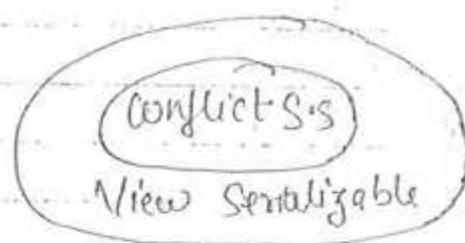
$S1 \neq S2$ because in S1 final update is done by $w_2(A)$ i.e. by transaction T2 but in S2 final update is done by $w_1(A)$ i.e. by transaction T1.

So, $S1 \neq S2$ because final write is not same.

View Serializable Schedule:-

View equivalent 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)	

T1	T2	T3
R(A)		
	W(A)	

(S1) Notes \Leftrightarrow T1:T2:T3 (S2)

S2 is equivalent serial schedule for S1. But S1 is not conflict serializable schedule even if view serializable schedule is possible.

\Rightarrow

T1	T2	T3	T4
W(A)			
	W(A)		
		W(A)	

T1:T2:T3:T4
(conflict SS)

There are 6 different serial schedules 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 do not change the serial of any update.

T1	T2	T3	T4
T1	T3	T2	T4
T2	T3	T1	T4
T2	T1	T3	T4
T3	T1	T2	T4
T3	T2	T1	T4

(Qn) $S: S_1(B) S_2(A) w_1(A) S_3(A) w_1(B) w_2(B) w_3(B)$
Identify all view equivalent serial Schedule.



i.e. Cycle is there so it is



Non conflict serializable schedule

For checking View Serializable Schedule:-

[1] Final Updation-



B: T1 T2



$(T_1 \ T_2) \rightarrow T_3$

\Rightarrow Either $T_1 \rightarrow T_2 \rightarrow T_3$
(due to initial
read condition)
 $T_2 \rightarrow T_1 \rightarrow T_3$

[2] Initial Read-

Initial Read

A: T2

Write

$T_1 \rightarrow T_2 \rightarrow T_3$

B: T2

$T_1, T_2, T_3 \Rightarrow T_2 \rightarrow T_1$

$T_2 \rightarrow T_2$

$T_2 \rightarrow T_3$

[3] WR Sequence-

$w_1(A) : r_3(A) \Rightarrow T_1 \rightarrow T_3$

Note:-
X! T1 T2

T2 T3 T4

T1 T2 → T3 T4 X
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 of the view equivalent serializable schedule.

(a) 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 :-

A! T1 T2

Initial Read Write

T3 T4

$\Rightarrow \{T1 \rightarrow T3, T2 \rightarrow T4\} (T1 T2) \times$

B! ~~000~~

T1 T2 T4

$\Rightarrow \{T1 \rightarrow T2, T4\}$

C! ~~000~~

T1

$\Rightarrow \{T1 \rightarrow T2, T4\}$

	Initial Read	Write
A	T1 T2	T3 T4
B	=	T1 T2 T4
C	=	T1

[B] WR Sequential

1) $w_1(C) \rightarrow s_3(C)$ $T_1 \rightarrow T_3$

2) $w_1(B) \rightarrow s_4(B)$ $T_1 \rightarrow T_4$

3) $w_1(I) \rightarrow s_4(I)$ $T_1 \rightarrow T_4$

4) $w_2(B) \rightarrow s_2(B)$ $T_2 \rightarrow T_2$

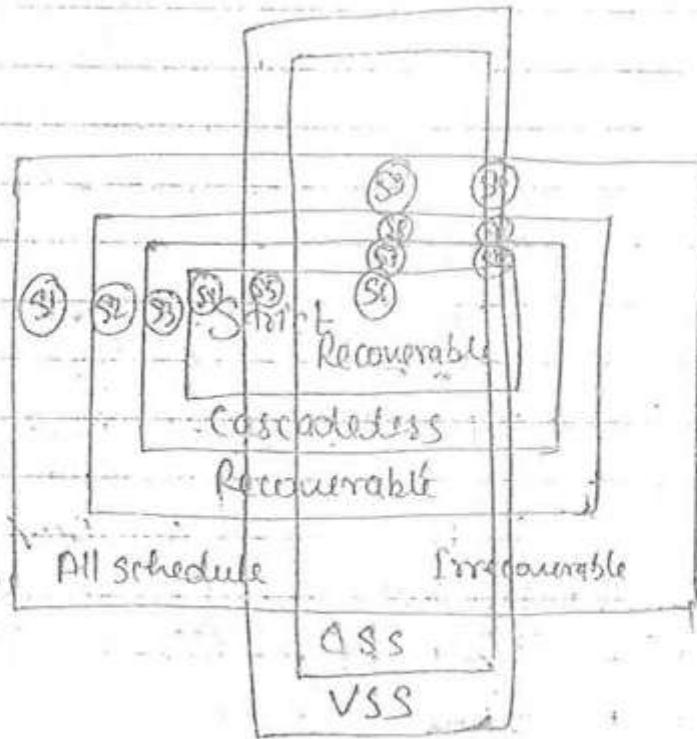
[D] $(T_1 T_2 T_3) \rightarrow T_4$ $(T_1 T_2) \rightarrow T_3 \rightarrow T_4$

[E] $(T_1 T_2) \rightarrow (B T_4)$

[F] $T_1 \rightarrow (T_3 T_4)$

$T_2 \rightarrow T_1 \rightarrow T_3 \rightarrow T_4$

Two view equivalent
schedule are possible.



Schedule is Irrecoverable & not Serializable!

	T1	T2
	R(A)	
	W(A)	R(B)
		R(B)
		W(B)
	C1	
		(S1)
		C2

Schedule is Recoverable but not Concurrent

	T1	T2
	R(A)	
	W(A)	R(A)
		R(B)
		R(B)
	W(B)	
	C1	
		C2

29/08/11

Concurrency Control Protocols:-

Ideal-

This protocol is allowed to execute only

- 1) Serializable Schedule { NO RW, Strict WR, WW Problem }
- 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.

T1	T2
w(A)	w(A)
w(B)	w(B)
*	

if the schedule is failed it rollback and update done by T2 is removed

(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.

\Rightarrow Every transaction in the schedule is illegal transaction

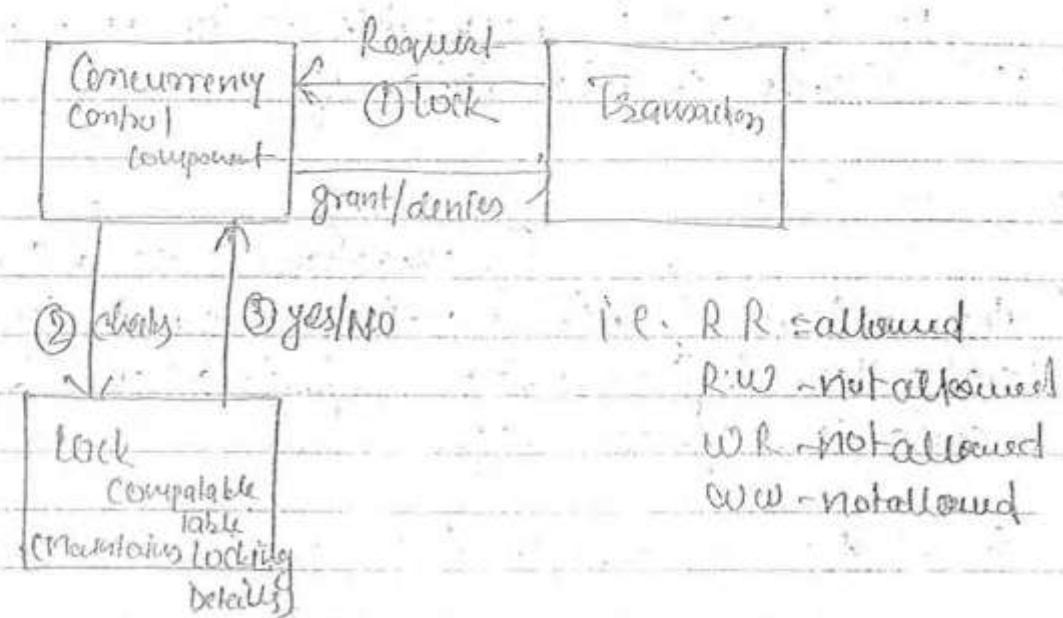
Ex:-
d(A)
R(A)
W(A)
V(A)

* '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 OS.
For this we use Semaphore: Mutex that also shares resource.

```
Process          : if(mutex=0)
if(mutex=0)      :
    mutex=1
    CS
    mutex=0
}
else
    { P2 leave to wait
        until mutex become 0( means
```

Components of legal transaction by locking Protocol



- Compatibility is in "Yes" condition only if -
- More than one transaction used only for "Reading" the data item
and
 - Data item is free from any transaction i.e. not used by any transaction.

Example:-

⇒ Shared / Exclusive locking [→ {S/X}]

Shared :- {S}

Shared mode means that transaction can read data item A.

① $\frac{I}{S(A)}$
 $R(A)$
 $W(A)$

* 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) :-

Transaction 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 :-

		Requesting Tj	
		S	X
(Hold Ti)		YES	NO
	S	NO	NO
	X	NO	NO

Requested

Example:-	T1	T2
	X(A)	
	R(A)	
	W(A)	
	U(A)	
		S(B)
		R(B)
		U(B)
		S(C)
		R(C)
		U(C)
		X(B)
		R(B)
		W(B)
		U(B)

Transaction is legal and also it follow the compatable table.

→ Legal Trans & compatability satisfied b/w S/X locks.

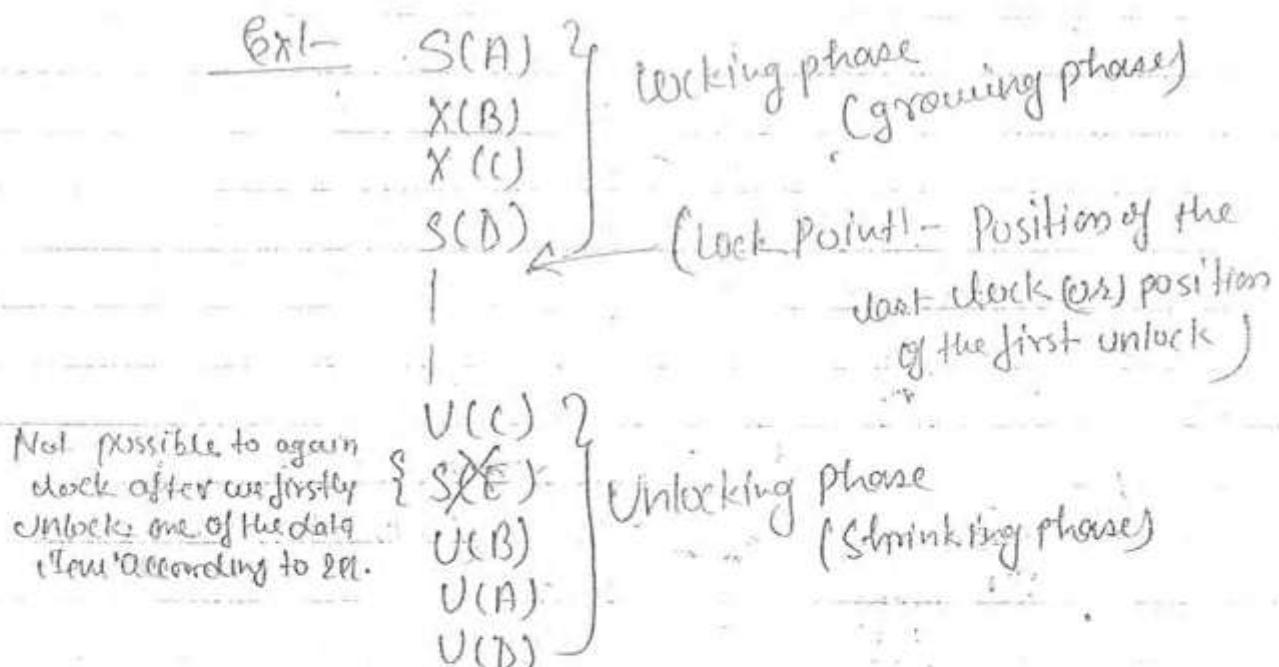
but non-serializable schedule

\Rightarrow (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

BT Two Phase Locking Protocol:-

Transition T not allowed
to request a lock if it is already performed
any unlock operation.



	T1	T2		T1	T2
	X(A)			X(B)	
	R(A)			R(A)	
	W(A)			X(B) / U(A)	S(A)
	V(A)	E(denied)		R(A)	
		S(A)		S(B)	
		R(A)		R(B)	
		U(A)		Q(B)	
		S(B)		W(B)	
		R(B)			
		V(B)			
		X(B)			
		RCB			
		W(B)			
		U(B)			

Not Possible to execute using 2PL.

⇒ 2PL! Always ensure conflict serializability. But it's not true i.e. every serializable is not 2PL.

- ⇒ If it is 2PL then it is serializable. But it's not always true i.e. every serializable is not 2PL.
- ⇒ Equivalent Serial Schedule based on the position of lock point.

Ques) Check that schedule is allowed to execute by 2PL or not.

T1	T2	T3
X(A) W(A)	S(A) R(A)	X(A) R(A)
<small>lock point</small> <small>unlock</small>		
W(B) V(B)	X(B) <small>lock point</small> W(B)	W(A)
V(B) V(B)	<small>unlock</small>	W(A)

Allowed To
execute by 2PL so it's
conflict serializable.

⇒ Not possible using 2PL

T1:T2

T1	T2	T3
X(B) R(B)	X(A) R(A)	
X(V(B))	X(Z(E)) R(A)	(X(M))
W(B)	X(U(A))	R(M)
		Not possible TC. Exclusive locking not allowed
		W(A)
		R(B)
		W(B)

T1 → T2 → T3

Conflict Serializable
Sequence 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 the same data item then we upgrade the lock to exclusive lock.

T1

S(A)

R(A)

|

|

X(A) ! reseekt Exclusive without
W(A) unlocking of the previous
shared lock.

⇒ Increase the concurrency level.

(Ques) Check this schedule to allowed or not allowed to execute 2PL using lock upgrading technique.

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

	T1	T2	T3
S(A)			
R(A)			
S(B)			
R(B)			
X(A)			
W(A)			
W(B)			
S(A)			
R(A)			
W(A)			

Not possible even by using 2PL with lock upgrading.

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.

(Ques) 2PL Allowed to execute?

T1	T2	T3	
S(A) R(A)			
S(B) R(B)	X(B) R(B)		→ Not allowed in 2PL
V(A) U(B)			
	W(B)		
		X(A) W(A)	
		V(X)	
	R(C)		
		W(C)	

lock lost upgrading technique using 2PL allowed or Not

T1	T2	T3	
S(A) R(A)		R(C)	
S(B) R(B)	S(B) R(B)		→ allowed to execute using lock upgrading technique.
V(A) U(B)	X(B) W(B)		
		X(B) W(A)	
	S(C) R(C)		
	V(C)	X(C)	Equivalent serial schedule
	U(B)	X(C) S(C) U(A)	
			T1 → T2 → T3

2PL with S/X Locking -

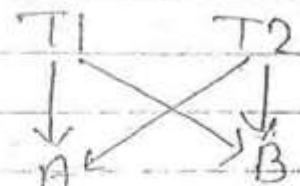
Limitation of 2PL -

Basic 2PL -	T1	T2
X(A)		
W(A)		S(A)
R(B)		R(A)
V(B)		
		⇒ Serializable schedule
		if allowed to execute
		by 2PL
		but UNPREDICTABLE
		Schedule.
		commit
	commit	

⇒ Serializable schedule
if allowed to execute
by 2PL

but UNPREDICTABLE
Schedule.

D	T1	T2
X(A)		
W(A)		X(B)
(Denied)		W(B)
	X(B)	X(B)
	W(B)	(Denied)
Waiting	{ W(B) }	Waiting.



e.g. T1 transaction needs T2 for execution & T2 need T1 for execution
then T1 & T2 gives deadlock condition.

* 2PL may results the deadlock also or may not free from Deadlock.

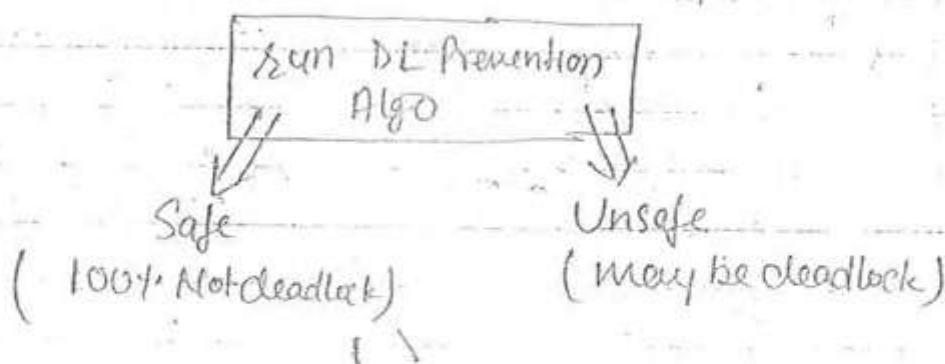
Starvation- If only one process can get to infinite block of data item then starvation occurs. One transaction permanently blocks due to unavailability of resource. Example- User has other transaction is executed this scenario is called starvation.

	T1	T2	T3	T4
denied (due to T2)	X(A)	S(A)		
	{			
denied (due to T3)	X(A)	S(A)		
	{			
denied (due to T4)	X(A)	V(A)	V(A)	S(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		T1	T2
	W(A)			W(A)	
		R(n)			R(A)
		C			Abort
	R/C			C	
	(Irrecoverable)				(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 exclusive locks should be held until commit or rollback.



we may unlock the shared lock before the commit or rollback
 But we always unlock the exclusive lock after the commit or rollback of transaction.

Example:-

	T1	T2
	X(A)	
	W(A)	
	commit	
	V(A)	S(A) / X(A)
		R(A)
		W(A)

Advantage of Strict 2PL:-

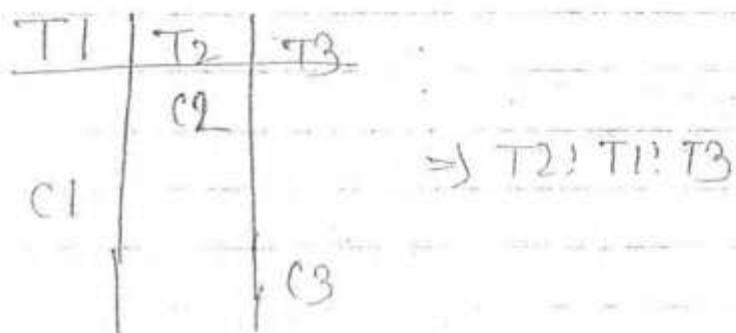
- * Always ensure strict Recoverable condition bcz of holding Exclusive locks until commit.
- * Bcz of 2PL ensuring the conflict serializability schedule. Equivalent Serial based on clock point.
- * This schedule is enough or sufficient to solve all the problem like strict Recoverable & conflict SS.

[3] Regarious 2PL Protocol-

Basic 2PL + All Exclusive locks / shared lock should be hold until commit.

T1
S(A)
X(B)
commit
V(B)
V(A)

- * Difference between Strict 2PL & Serializable 2PL is that in strict 2PL only all exclusive locks should be held until commit but in Serializable, all exclusive lock as well as all shared lock also should be held after commit.
- Ensuring strict Recoverability
- Ensuring serializability
- ⇒ Only difference is that equivalent serial schedule based on order of commit.



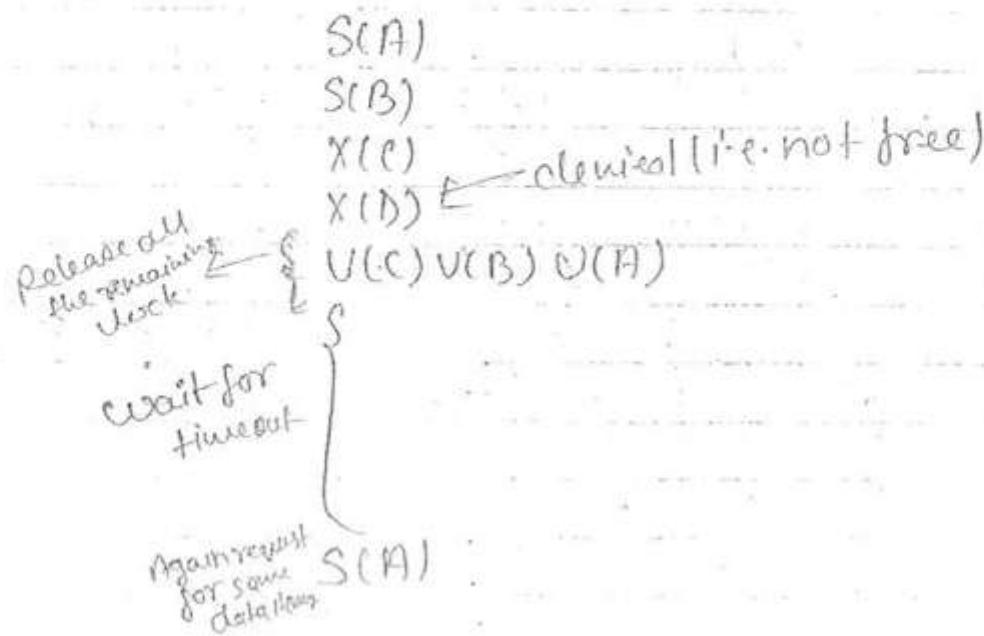
Conservative 2PL Protocol:-

- ⇒ This protocol is basically made for To Avoid Deadlock.
- ⇒ It says that lock the all data items 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)

Q) Some data item is not free -



Disadvantages of Conservative 2PL -

- Starvation Possible
- Irrecoverability is possible.

Example -

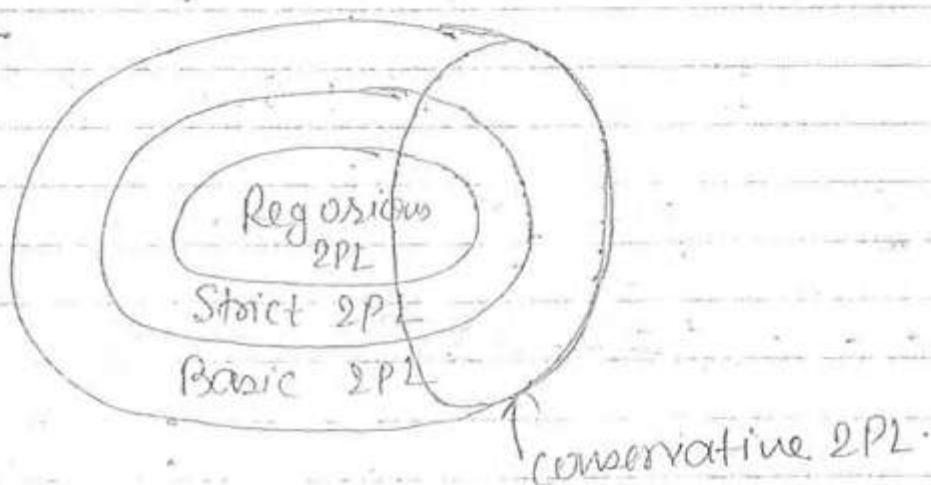
	T1	T2
X(A)		
X(B)		
W(A)		S(A)
U(A)		R(A)
		U(A)
		Commit
	W(B)	
	W(C)	
		Commit

Here it is irrecoverable
but also conservative
2PL protocol is possible.

* If we only put the restriction of unlock then only strict recoverability is possible.

[c] [The resources are held long before the requirement] because of this less concurrency is occur, so throughput is less.

[d] Prediction of locks required by the transaction is too difficult.



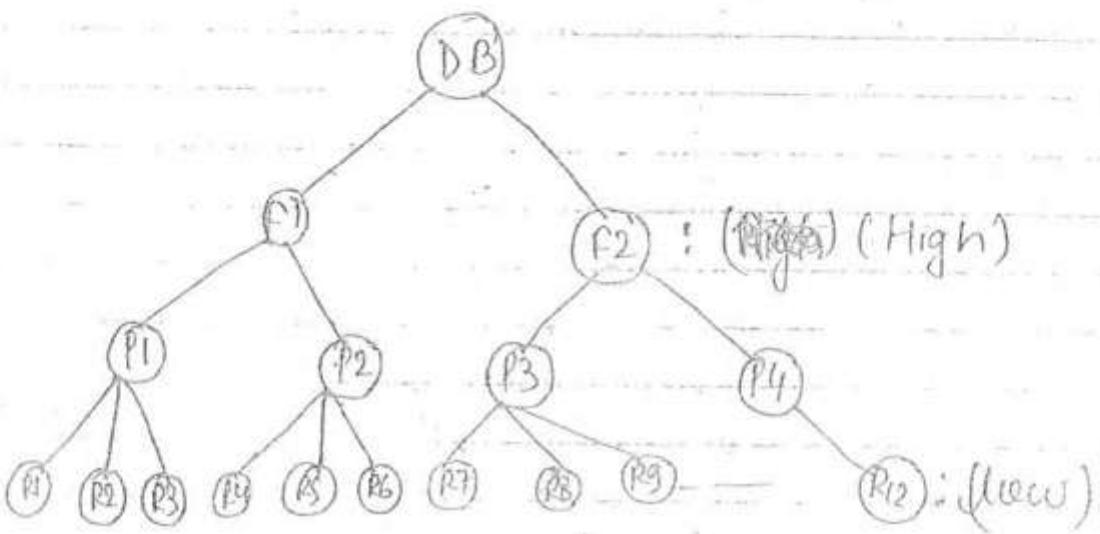
Ques)	T1	T2	T3
	S(A) R(A)		
	X(B) U(C)	S(C) R(C)	X(A) W(A)
	W(B) U(B)	R(A) U(A) U(C)	U(A) (viewed in strict 2PL)
	Commit	Commit	Commit

- a) Not in 2 PL
- b) 2PL but not strict 2PL
- c) Strict 2PL but not Regressions
- d) Regressions 2PL

Multi-level Granularity Protocol! - (Tree Protocol)

Granularity-

Granularity means lock in the DB.



There are few types of lock on the basis 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- * T3 : (L1, f1)

Advantage- No. of locks are very less
- Lock computable Table
- Easy to maintain

* T2 : L2(f1) L2(f2)

Disadvantage- results less concurrency
because Transaction may hold unnecessary resource

[2] Low-level locking :-

Disadvantages

- * T1: $\cup_{j=1}^3 R_j \cup_{i=1}^2 R_i \cup_{k=1}^3 R_k$ } Complex to maintain
lock compatibility
 - * T2: $\cup_{j=1}^2 R_j \cup_{i=1}^2 R_{i,j}$ } (More concurrency level)

[3] Multi-level Granularity - (Best possible duck deployment) ^{Transit}
^{reqd} occupancy

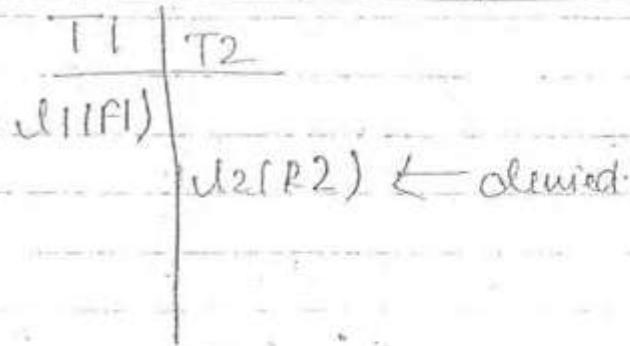
Depends on transaction fee

should allow to look at any level.

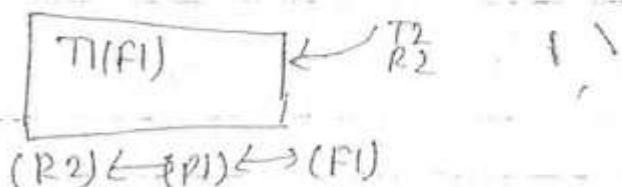
- * TI : $\cup_1(f)$

- * T2: $\mathcal{J}_2(f_2) \cup \mathcal{J}_2(\bar{R}_{12})$

* Firstly file is checked thereafter request for clocking record?
Example: T1 T2



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
 \leq constant. (To grant or to deny the permission)

\Rightarrow If first ~~record~~ is locked & we request for locking the file in which lock record is belongs! -

T1	T2
	U ₂ (R ₂)
U ₁ (F ₁) ↓ denied	

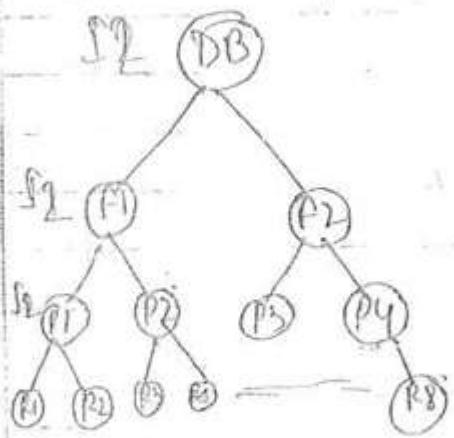
Difficulty comes for concurrency control to allow or grant the permission or deny the permission!

U₂(R₂)

Checked for locking of F₁, P₁, P₂, R₁, ..., R₆)

$O(2^n)$ i.e. exponential time.

To solve the above difficulty:-
we have to maintain Intention lock at every level i.e.



I₂(DB) I₂(F₁) I₂(P₁) U₂(R₂)

Sol 08/11

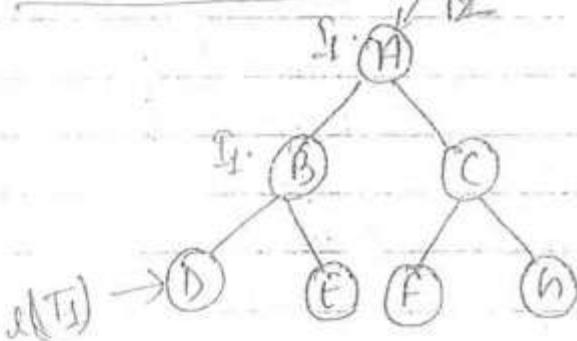
Tree Protocol:-

a) Multilevel Granularity Protocol:-

Intension Lock:-

A Node N can be locked by Trans T in Intension mode means that from T can allowed to request lock on Any descendent of N.

To reduce the below problem we use Intension lock.



T1	T2
$d_1(D)$	

$d_2(A) \in$ (Exponential time)
becoz lock status require to check Any descendants of A.

Intension lock

T1

$d_1(B)$
 $d_1(D)$
 $R(D)$
 $d_1(E)$
 $w(E)$

Direct lock

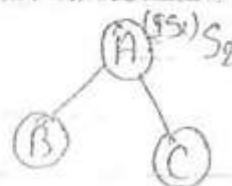
T1

$d_1(B)$
 $R(D)$
 $w(E)$

There are following Intension locks:-

[i] Intension Shared lock [S] :- A node N locked by transaction T in intension shared (S) mode means that any descendant of N can be request shared lock (S) by transaction T.

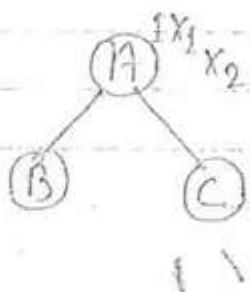
→ If parent is intension shared lock it only allow the read the data item.



	T1	T2
S(A)	S(A)	S(A) (only shared lock)
S(B)	R(B)	
R(B)		R(C)
S(C)		
R(C)		W(C)
X(C)		

[ii] Intension Exclusive lock [IX] :-

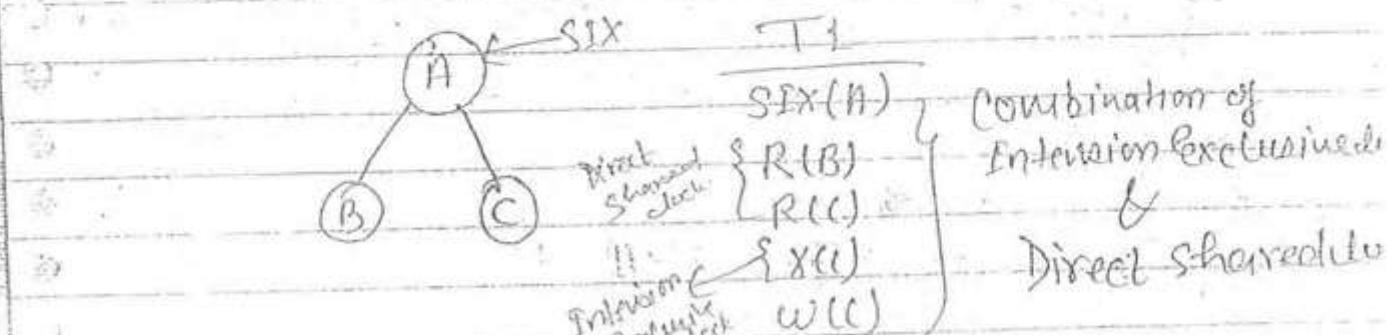
A node N locked by transaction T in intension exclusive mode means that any descendant of N can be request shared / exclusive lock by transaction T.



	T1	T2
IX(A)	X(A) (only Exclusive lock)	
S(B)		R(B)
R(B)		
X(C)		W(C)
W(C)		

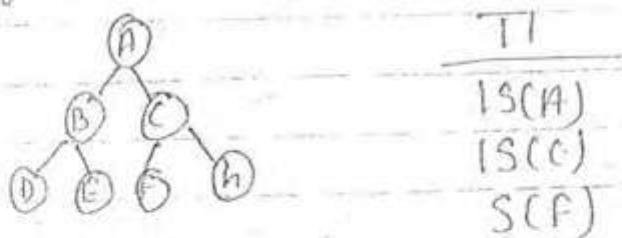
BJ Shared Intentions Exclusive Lock [SIX]

It is Combination of Intentions Exclusive lock & Shared lock.

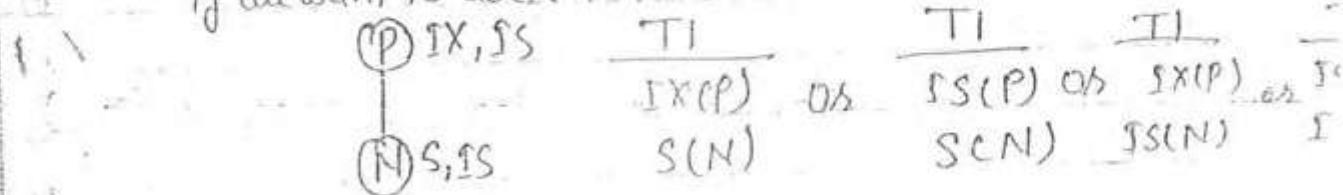


Multilevel Granularity Protocol Conditions:-

- [1] A node N can be locked by Transaction T only if parent of N is already locked by Transaction T.
If we want to lock data item "F". Then



- [2] A node N can be locked by Trans T in S, IS only if parent of N is already locked by Trans T in either 'IS' or 'IX' mode.
If we want to lock N in shared mode:-



(Ques) What is worst-time complexity for Quick Sort
a) $\log n$ b) n c) $n \log n$ d) n^2

(Ques) Which of the rays are not emitted by any particle.
a) α b) β c) γ d) None of these

(Ques) Which of the layer is responsible for routing
a) Physical layer b) Data link layer c) Network layer d) Transport layer

(Ques) There are 16 elements in the array. Then what will be the values from after execution of this program
 $a = [4\ 4\ 5\ 5\ 6\ 6\ 6\ 7\ 7\ 7\ 8\ 8\ 8]$
count = 0

```
for(i=1 to 16)
    if(a[i] == 7) then
        count = count + 1
    else if(a[i] == 8)
        i++;
```

a) 3 b) 5 c) 6 d) 7

(Ques) float a = 63.75 b = 35.75

float a = 35.75 b = 71.50

b = a

if(a = b)

b = 29

b = b/2

Then what will be the value of b.

(Ques) Which join is used to select only distinct column
a) Natural join b) Outer join c) equijoin d) none

(Ques) Which one of them is not important in Digital signature
a) Hash function b) Encryption c) data hiding d) Decryption

- (Ques) Int *P, *q then which of them is not current operation
 a) P+q b) P-q c) P*q d) P / q
- (Ques) In the given sorted array if there are two elements of same value then which type of sorting keep the element in same order after sorting them.
 a) Stable b) Strict
- (Ques) What is the full form of URL
- (Ques) In complete binary tree what the difference between two sub tree of any node are possible
 a) 1 b) 2 c) 0 d) 3
- (Ques) If x=1111 y=1111 and z=1111 are elements arranged in order then in preorder which element is taken first
 a) x b) y c) z d) None of these
- (Ques) Calculate the value of $-ad + -cd/b$ if $a=2, b=4, c=3, d=5$
- (Ques) What is the full form of SSL in security purpose in SSL.
- (Ques) 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 c) 4NF d) None of these
- (Ques) In a ER model which of them is not so important to taken
 a) Entities b) Attribute c) Primary key d) Relationship
- (Ques) No. of minimum interchange required to keep the maximum no. of elements at the root in tree
 89, 17, 12, 15, 213, 4, 79,
 if $n \geq 8 > 200: 300$ if $n = 4$ $y = 4$ $z = 6$
 a) 200 b) 300 c) 100 d) None of these
- (Ques) $S = S \cup S^*$ which of them is not belongs to this language
 a) aabb b) abab c) aaabb d) aabab

T1	T2	T1	T2
<u>U(A)</u>		<u>Y(X(A))</u>	
<u>X(B)</u>	<u>Z(A)</u>	<u>X(C)</u>	<u>Y(X(C))</u>
<u>U(B)</u>	<u>Y(B)</u>	<u>W(B)</u>	<u>X(B)</u>

14841010-23221

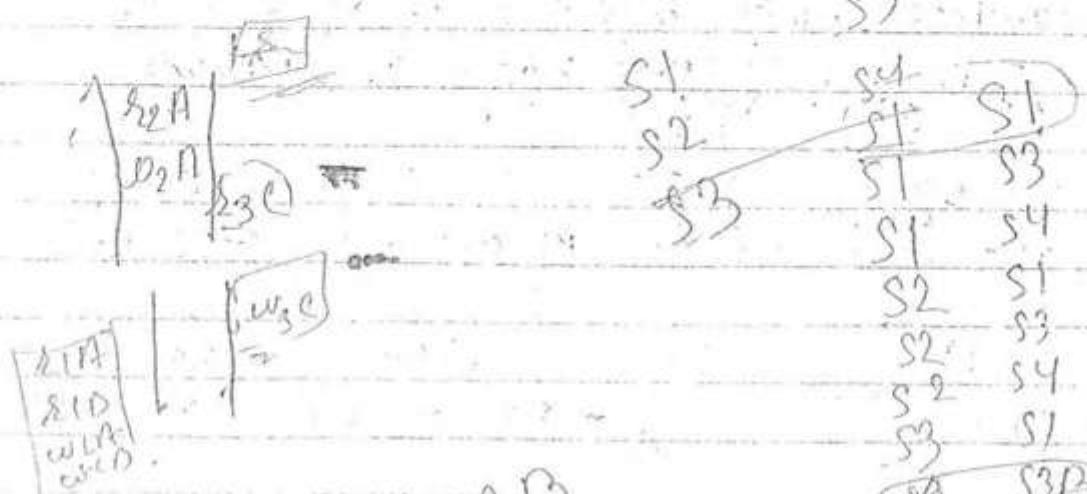
Krishna Kumar Pa.
Cahra Bank
Jitsingh Marg New

green 110067

seed

53

110067



$A \oplus B$
 $A \oplus B^{\perp}$
 $A^{\perp} \oplus B^{\perp}$
 $A^{\perp} \oplus B$
 $A^{\perp} \oplus B^{\perp}$
 $A^{\perp} \oplus B^{\perp}$

~~33~~ S3P
~~34~~ S4

2

620 600 15071 301 694 37

7450

A B C D

11 D

$$q = b$$

- [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.

(P) IX, SIX

(N) X, IX, SIX

- [4] ^{Transaction}
An ancestor can be request for any lock on Node only if Transaction T not performed any unlock operation (2PL)
(This condition is required to ensure serializability)

- [5] A Node N can be unlock by Transaction T only if None of the descendants of N is already locked by T. (i.e. unlock is ^{down} from bottom Top.)

[6] Compatable Table:-

Compatibility:-

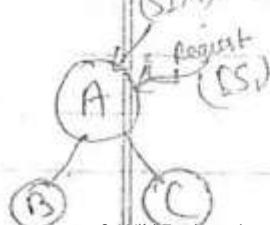
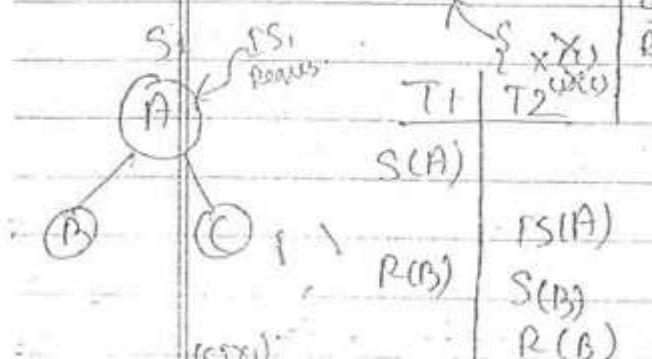
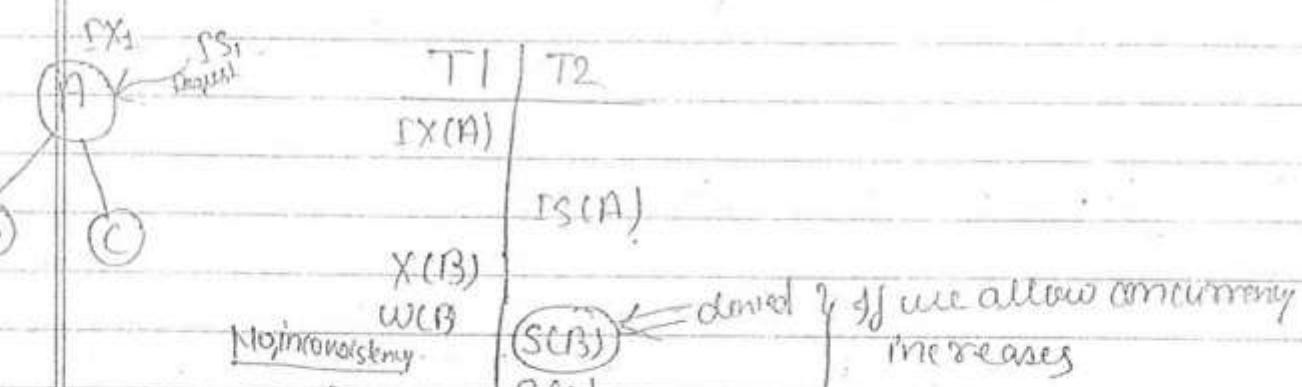
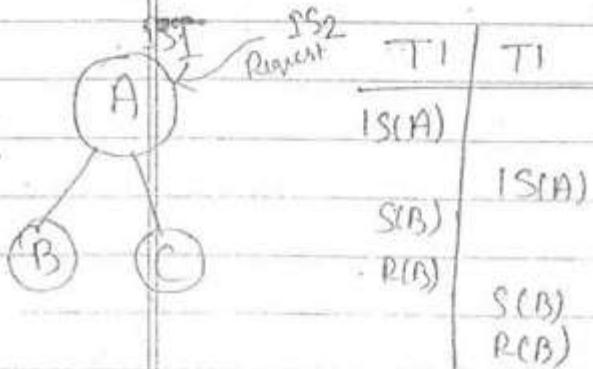
If Node N is locked by Transaction (Ti) & another transaction (Tj) request for locking the same node then compatibility that whether to grant the permission or deny the request of transaction (Tj).



where $i \neq j$

ifj	IS	IX	S	SIX	X	→ Hold Tr
IS	YES	YES	YES	YES	No	
IX	YES	YES	No	No	No	
S	YES	No	YES	NO	NO	
SIX	YES	No	NO	NO	NO	
Tj	X	No	No	NO	NO	

↓ Request

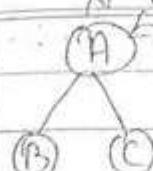


STX → IS

S → IS ⇒ Yes

IX → IS ⇒ Yes

X_1 Request (IS₁)



T1 | T2
X(A)

IS(A) ← if we grant

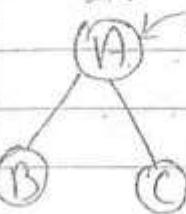
SC(B)

R(B)

(W(B))

if we grant then R/W problem may possible

X_1 Request (SX₂)



T1 | T2
S(A)

IX(A) ← grant

X(B)

W(B)

R(B)

SIX — IX \leftrightarrow S — IX(N)
SIX — IX \leftrightarrow IX — IX(Y)

SIX — IS \leftrightarrow S — IS(Y)
SIX — IS \leftrightarrow IX — ISC(Y)

*

Intension — Intension lock

IX — IX ?

IX — IS

IS — IX

IS — IS

} Compatable

Simultaneous Read Compatable! -

S — IS ?

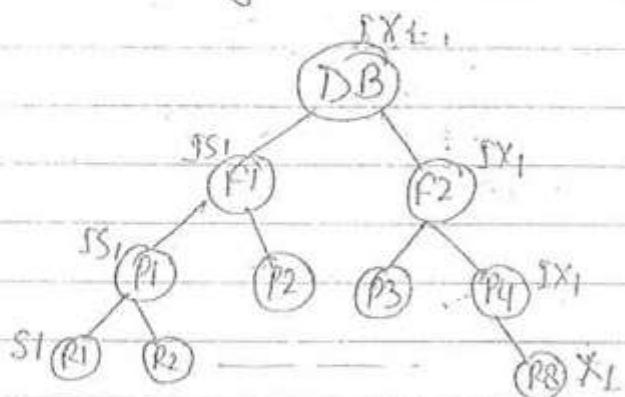
S — S } Compatable

IS — S

* If the transaction satisfy all the above six condition then only it is called multilevel granular

(Qn)

T1! Read R1 & update R8 then what will be the locking pattern.

T1IX₁(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)
(becoz it is in 2PL)

⇒ Precoverability Possible becz there is no any restriction for commit of any transaction

Example:- check multilevel granularity is possible or not.

	T1	T2
A	IX(A)	
B	X(B)	
C	W(B)	
	V(B)	
	U(A)	
...		
		IS(A)
		SCB}
		R(B)
		V(B)
		V(A)
		Commit
	Commit	

i.e. multilevel granular
is possible but it
Precoverable schedu

⇒ Deadlock & Starvation possible.

* To remove the Precoverability 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/rollb

⇒ Ensure Serializability & Strict Recoverability.

⇒ Even deadlock & starvation is possible.

Time stamp Ordering Protocol:-

One difference between locking & Timestamp protocol is that Time stamp 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 orders.

Advantages of Timestamp values:-

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.

(Transaction with less TS value gets high priority).

Example:-

	10	11	12	13
T1				
older				
T2				
T3				
T4				
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)		

$$RTS(n) = 30$$

2) W_S(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 step ordering

(T1:T3:T2)

* Concurrenty of the Timestep 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 school 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)) →

It violates the serial i.e. $T1 \rightarrow T3 \rightarrow T2$
so T3 is illegal

If given serial is $T_1 \rightarrow T_3 \rightarrow T_2$

$$\Rightarrow \begin{array}{l} T_1 \rightarrow T_3 \\ T_1 \rightarrow T_2 \\ T_2 \rightarrow T_1 \end{array}$$

- * $T_2 \rightarrow T_1$ is possible but $T_1 \rightarrow T_2$ is not allowed i.e. cyclic dependency is not possible so it is deadlock free protocol.

Date _____
Page _____

Possibility of Rollback I:-

1)

10 <u>T_1</u>	20 <u>T_2</u>	i.e. $T_1 \rightarrow T_2$
$R(A)$ (Rollback T_1)	$W(A)$	<p style="text-align: center;">" If younger transaction updated data item. Then older not allowed to Read" bcz conflict pair from $T_2 \rightarrow T_1$.</p>

Transaction T_1 issues $R(A)$ operation -

$$WTS > TS(T_1)$$

Rollback T_1

2)

10 <u>T_1</u>	20 <u>T_2</u>	i.e. $T_1 \rightarrow T_2$
$W(A)$ (Rollback T_1)	$R(A)$	<p style="text-align: center;">" If younger transaction need data item A . Then older transaction not allowed update A" bcz conflict pair order is from $T_2 \rightarrow T_1$.</p>

Transaction T_1 issues $W(A)$ operation -

$$RTS > TS(T_1)$$

Rollback T_1

3)

10 <u>T_1</u>	20 <u>T_2</u>	i.e. $T_1 \rightarrow T_2$
$W(A)$ (Rollback T_1)	$W(A)$	<p style="text-align: center;">" If younger transaction updated data item . Then older transaction not allowed to update " bcz conflict pair order is from $T_2 \rightarrow T_1$.</p>

* When conflict pair is based on timestamp orders then there is no need to rollback. But when conflict pair is not based on timestamp order then we have to rollback the transaction.

Date _____
Page _____

Transaction T1 issues W(A) operation -

$$WTS(A) > TS(T)$$

Rollback T1

Basic 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

Then Set $RTS(A) = \max\{RTS(A), TS(T)\}$

Ex:-	10 T1	20 T2	30 T3
R(A)			
	R(A)	R(A)	

$$RTS(A) = 10, 30$$

R(A) Here is no any conflict pair then we allow this transaction.

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

$$WTS(A) = TS(T)$$

Ex:-	10 T1	20 T2	30 T3	$WTS(A) = 10, 30$
	$w(A)$		$w(A)$	$w(A)$

1 By default we take transaction no. as timestamp value if the timestamp value is not given.

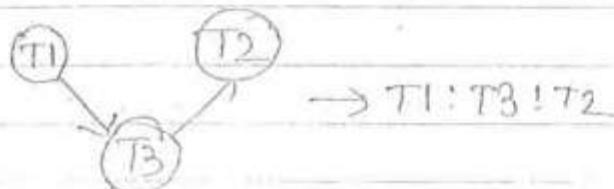
(Ques) which of the time stamp values allow to execute the schedules

S: $s_1(A)s_2(B)w_1(C)s_3(B)s_3(C)w_2(B)w_3(A)$

- a) $\{10, 20, 30\}$
- b) $\{30, 10, 20\}$
- c) $\{10, 30, 20\}$
- d) $\{30, 20, 10\}$

$s_1(A): w_1(A) \xrightarrow{1} 3$

$s_3(B): w_3(B) \xrightarrow{3} 32$



(Ques) If timestamp $\{10, 20, 30\}$ are given then which transition is roll back.

$\{10, 20, 30\}$

$T1 \rightarrow T2 \rightarrow T3$

$T1 \rightarrow T2$

$T2 \rightarrow T3$

$T1 \rightarrow T3$

then check in the transition one by one by taking operation from above schedule and check for taking previous operation, that where conflict pair is and is the conflict pair satisfy above precedence of the given question.

S: $s_1(A)s_2(B)w_1(C)s_3(B)s_3(C)w_2(B)w_3(A)$

$T3 \rightarrow T2 \neq T2 \rightarrow T3$

So, T2 transition roll back T2

Once you discard the one transaction then discard all the operation related to that transaction for further checking that whose other transaction will be rollbacked.

Ex- $s_1(A) s_2(B) w_1(C) s_3(B) s_3(C) w_2(B) w_3(A) w_2$

b) T_1, T_2, T_3
 $(30, 10, 20)$

$T_2 \rightarrow T_3 \rightarrow T_1$

i.e. $T_2 \rightarrow T_3$
 $T_3 \rightarrow T_1$
 $T_2 \rightarrow T_1$

$s_1(A) s_2(B) w_1(C) s_3(B) s_3(C) w_2(B) w_3(A)$

$w_1(C)$ is conflict with $s_3(C) \Rightarrow 1 \rightarrow 3$ not allow
 so, we discard till operation of T_3
 \Rightarrow Rollback the Transition T_3 .

b) T_1, T_2, T_3
 $(30, 20, 10)$

$T_3 \rightarrow T_2 \rightarrow T_1$

$T_3 \rightarrow T_2$

$T_2 \rightarrow T_1$

$T_3 \rightarrow T_1$

$s_1(A) s_2(B) w_1(C) s_3(B) s_3(C) w_2(B) w_3(A)$

$s_1(A) \rightarrow w_3(A) \Rightarrow 1 \rightarrow 3$ (not possible)
 So, T_3 is rollbacked.

THOMAS WRITE TIMESTAMP ORDERING PROTOCOL

Possibilities of Rollback -

[1]	10	20
	T1	T2
	R(A)	
	<u>R(A)</u>	

(Rollback T1)

[2]	10	20
	T1	T2
	<u>R(A)</u>	
	W(A)	

(Rollback T1)

[3]	10	20	el
	T1	T2	
		W(A)	↓ younger transaction updated data item then older transaction updates are ignored"
	<u>W(A)</u>		

Ignore Write(A),
but continue the
Execution of T1

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 $WTS(A) = TS(T)$

(a)	10 T1	20 T2	30 T3
$R(A)$			
$w(A)$			

Withdrew

Basic TO protocol:- (close to conflict Serializable)

Rollback T_1 bcoz of $w_1(A)$

TWR TO protocol:- (close to view serializable)

10 T1	20 T2	30 T3
$R(A)$		
$w(A)$		

Withdrew

No Rollback

($w_1(A)$ ignored)

$\Rightarrow T_1 \rightarrow T_2 \rightarrow T_3$ is equal to view

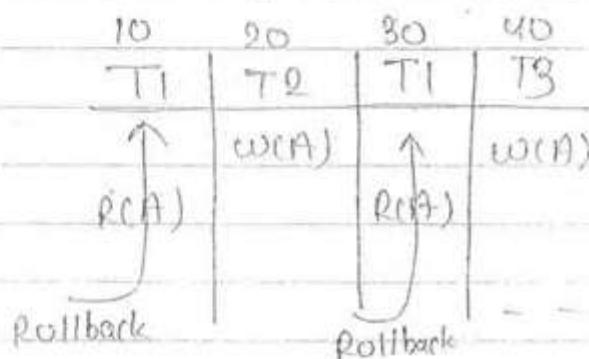
serializable
www.raghul.org

RULE of Thomas Write Time Stamp Ordering protocol
4k Basic TS ordering:-

[1] Serializability
 (Equivalent Serial based on order
 of TS values)

[2] free from Deadlocks

[3] Not free from starvation:



[4] May not free from irrecoverability -

10	20
T1	T2
W(A)	
R(A)	

BTO + TWRTU allowed to
 execute the schedule. But
 commit this schedule is irrecoverable.

(commit)

Strict Recoverability :-

10	T1	T2	20

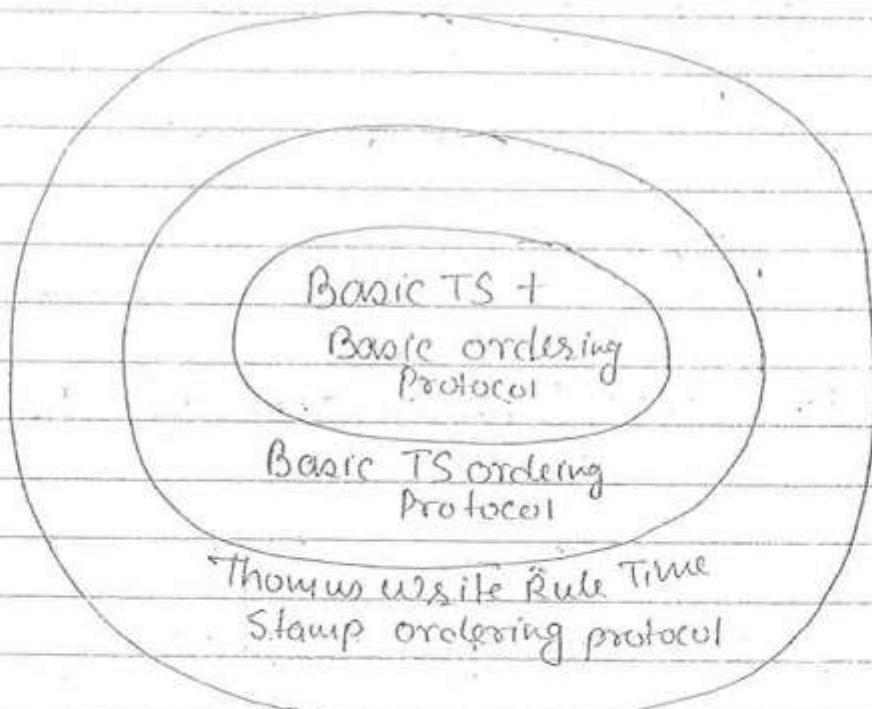
10 20

T1	T2
wcn)	
C/R	
	R(A)/W(A)

i.e. $WTS(A) < TS(T_2)$ Strict Timestamp Ordering Protocol:-

Transaction T_2 issues $R(A)/W(A)$ operations if $WTS(A) < TS(T_2)$. Then transaction (T_2) which issuing $R(A)/W(A)$ operation should be delayed until commit/ Rollback of T_1 that is already performed write operation.

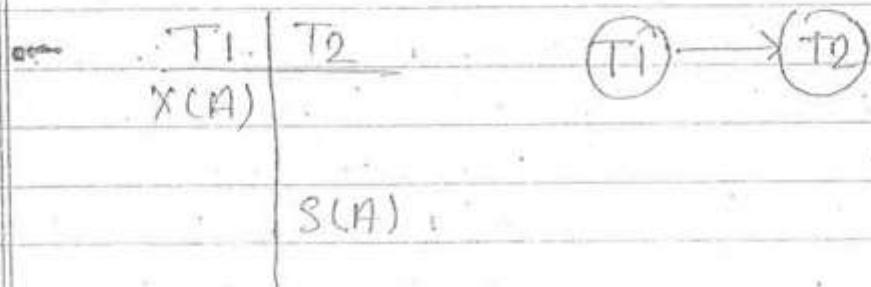
*



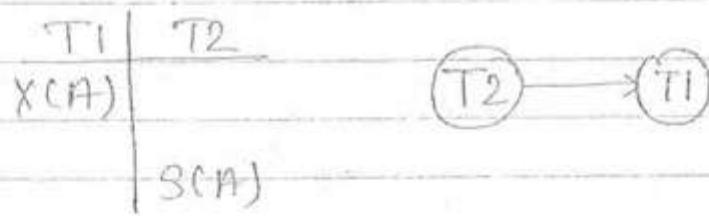
Deadlock Prevention Algorithms:-

The basic purpose of deadlock prevention algorithm is to prevent the deadlocks in lock based protocol using Time Stamp Values.

Precedence Graph:-



Dependency Graph:-



\Rightarrow i.e. dependency graph is opposite of Precedence graph.

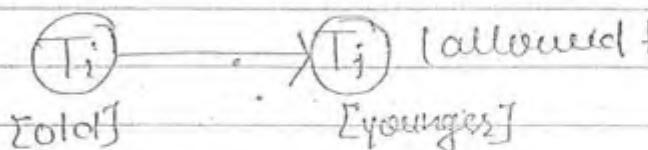
These are two types of protocol:-

WAIT-DIE Protocol:-

a) Assume $TS(T_i) < TS(T_j)$

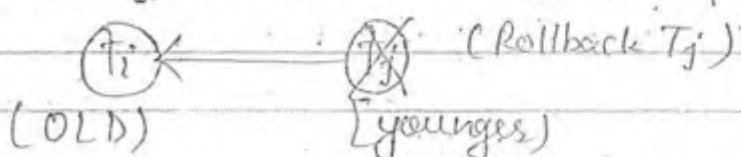
b) If Transaction T_i required resources

that is held by T_j , Then T_i allowed to wait until T_j unlocks.

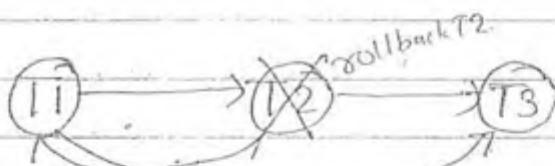


$\times(T_j)$ (allowed to wait)
[YOUNG]

- (ii) If Transaction T_j required resource that is held by T_i , then Rollback T_j (& to remove starvation) Restart with same Timestamp value.



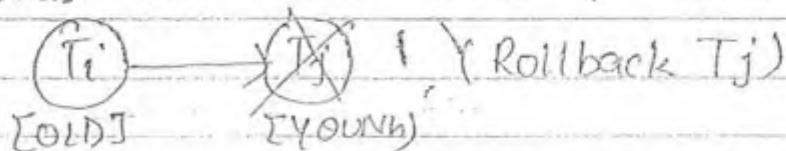
⇒ Deadlocks are not possible here bcoz we are not allowed to form cyclic dependency.



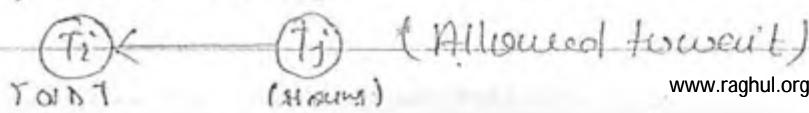
[2] CROUND-WAIT Protocol:-

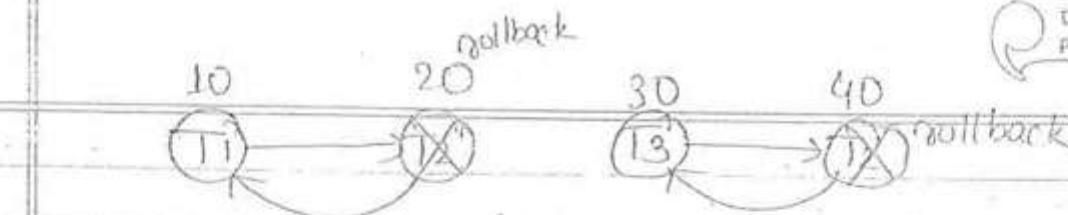
a) Assume $TS(T_i) < TS(T_j)$

b) If Transaction T_i required resource that is held by T_j , then Rollback T_j (& to remove starvation) Restart with same Timestamp value.

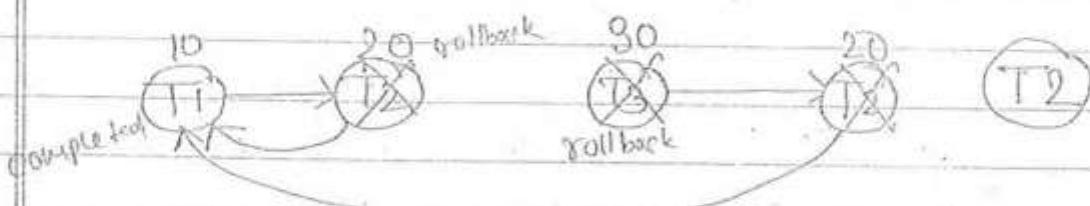


c) If Transaction T_j required resource that is held by T_i , then T_j allowed to wait until T_i unlocks.





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 no 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 infinite time. So, starvation is removed.

→ Concurrency problems

→ PW

→ WR

→ UW

→ Classification of schedule

→ Recoverability

→ Serializability

→ 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:-

(i) Spanned Organization-

(ii) Unspanned Organization.

(i) Spanned Organization-

Record can be spanned b/w
the blocks.

100-B1	R1	40
	R2	40
	R3	40
B2		

Block Size = 100 by.
Record size = 40 by.

Block Factor-

No. of
records per block.

$$= \frac{\text{Block Size}}{\text{Record Size}}$$

$$= \frac{100}{40} = 2.5$$

Advantage of Spanned organization-

→ No. memory leaks (or) No internal fragmentation.

Disadvantages-

→ More I/O cost.

[2] Unspanned Organization:-

Entire record should be accommodated in single block.

Bl	R1	40
	R2	40
		20
B2	R3	
	R4	

Block Size = 100 bytes.

Record size = 40 bytes

$$\text{Block factor} = \left\lceil \frac{\text{Block Size}}{\text{Record Size}} \right\rceil$$

$$= \frac{100}{40} = 2$$

Disadvantage:-

⇒ Internal fragmentation possible.

Advantage:-

⇒ Less I/O cost.

becuz

Secondary memory \rightarrow 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

search key

Here we use Empno. to select the data as search key.

Select *
From Emp
where Ppno = Y
Search key

Search key is a key use to compare the data.

* Seary key may not be primary key.

Emp DB file

ORDERED FILE -

DB records: physically ordered based on Search key attribute.

Select from Group

where $E_{\text{NO}} = x$
 Search k_n

Search key

11

(ordered file)

Select *

From Emp

where phon- y

Search key

11

(UN-ORDERED FILE)

Worst Case I/O cost:-

if the file is ordered file : I/O cost :- $\lceil \log_2 N \rceil$

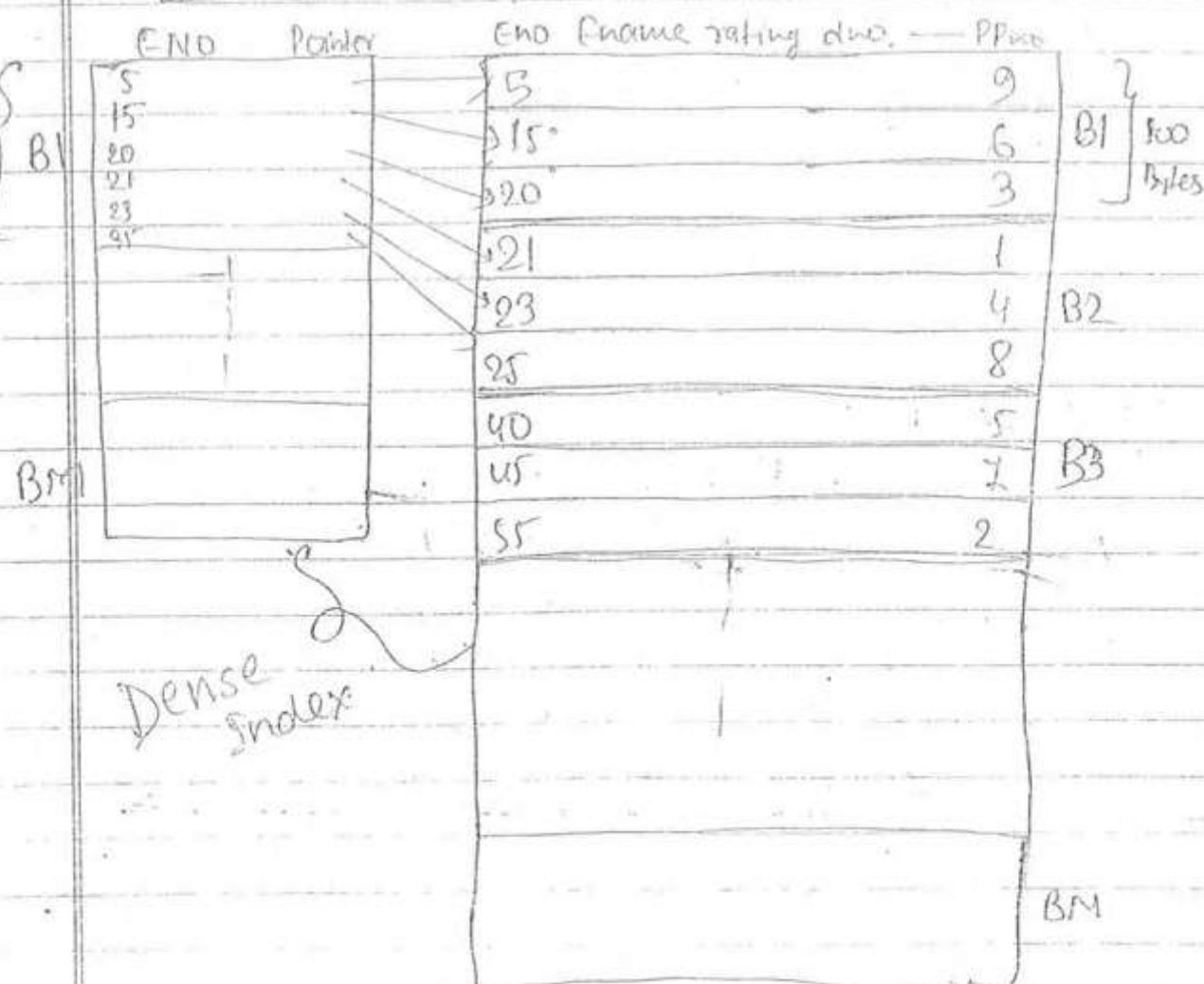
if the file is un-ordered : N^2 blocks

To reduce the I/O cost from $\lceil \log_2 N \rceil$ up to N
we go for indexing.

INDEX INFILE

Idea 1 - [To reduce I/O cost]

Size of DB Block = Size of Index block
Indexfile (E.g. DB file)



⇒ Entry of the Index file :- Two fields

< Search Key , Pointer >

Ex:- < Eno , Pointer >

⇒ Entry size of Index file << Entry size of DB Rec.

⇒ No. of entries per block = Block Factor

as per above diagram block factor for

Index file = 6

DB file = 3

⇒ "Block Factor" of Index file \gg Block Factor of DB file

⇒ Total No. of Index File Block << Total No. of DB File Blocks.

$$(M \ll N)$$

⇒ I/O cost with Index = $\lceil \log_2 M \rceil + 1 < \lceil \log_2 N \rceil$

Categories of Index -

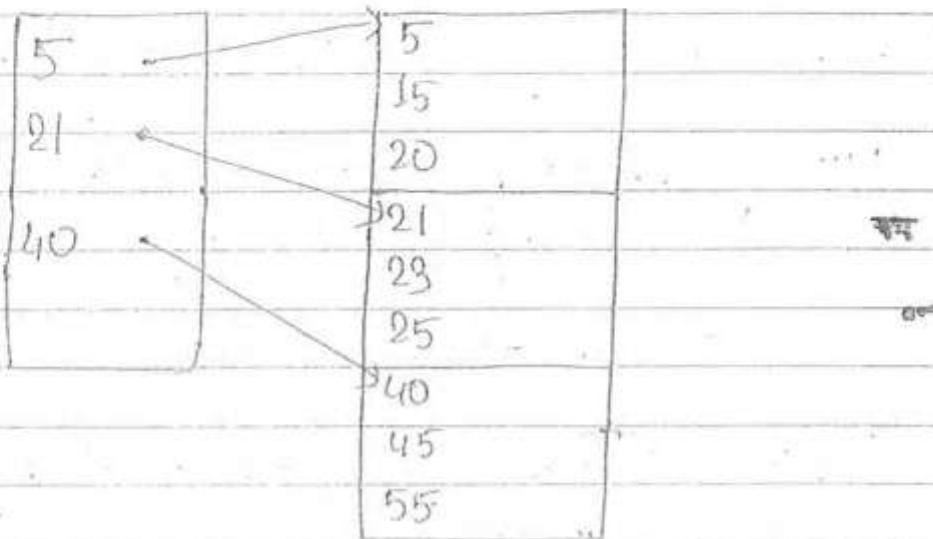
[ii] Dense Index -

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} Records
Per Record

Special Case:-

$$\left\{ \begin{array}{l} \text{No. of Index} \\ \text{file entries} \end{array} \right\} = \left\{ \begin{array}{l} \text{No. of DB} \\ \text{Blocks} \end{array} \right\}$$

(Ques) Blocks are 1000 bytes.

Records are 100 bytes of which 10 bytes takes keys & pointer take 8 bytes.

10000 records exist in the database.

How many no. of Dense & sparse Index blocks?

$$\text{Block Factor of DB file} = \frac{1000}{100} = 10 \text{ Record/Block}$$

$$\text{No. of DB Blocks} = \frac{\text{No. of records}}{\text{Block Factor}} = \frac{10000}{10} = 1000 \text{ Blocks}$$

I/O cost without index $\Rightarrow \lceil \log_2 1000 \rceil = 10$ blocks

$$\text{Or} \quad = \frac{1000}{N} \text{ blocks}$$

By using Indexing:-

Entry size of Index = $10 + 8 = 20$ Bytes

Block factor for index file = $1000 \cdot \frac{50}{20} = 50$

Dense Index

No. of Index File Entries -
No. of DB record
 $= 10,000$

$$\text{No. of Block} = \frac{10000}{50} \\ = 200 \text{ blocks}$$

$$\text{I/O cost} = \lceil \log_2 200 \rceil + 1 \\ = 8 + 1 = 9 \text{ blocks}$$

Sparse Index

No. of Index File Entries:
No. of DB record =
 1000

$$\text{No. of Block} = \frac{1000}{50} \\ = 20 \text{ blocks}$$

$$\text{I/O cost} = \lceil \log_2 20 \rceil + 1 \\ = 5 + 1 = 6$$

Types of Indexes:-

(i) Primary Index:-

Index is said to be primary index if it satisfies following condition:-

(i) Conditions:-

(1) Search key used in the index file should be

used to physically order 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.

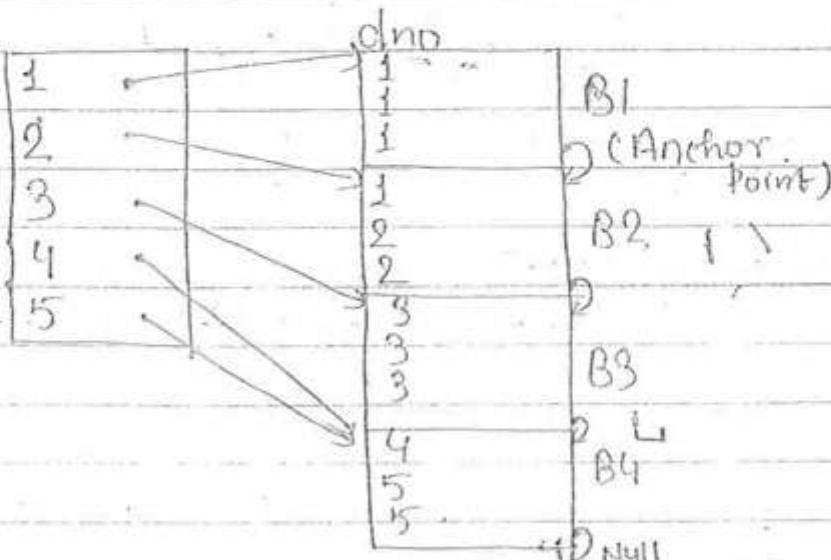
(3) At most one primary index is possible.

[2] Clustering Index:-

(a) Condition:-

[1] Search key used in the index file should be used to physically order the DB file.
[Ordered file].

and [2] Search key should be Non-key
[Non - KEY].



- * Ordering of file is done on only one key or 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 $I/O \text{ cost} = 1 + N = N$

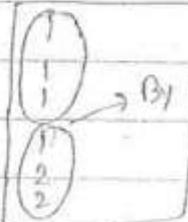
Date _____

Page _____

Select *

from Emp

where dno = 1



BN

मात्र

(b)

Always sparse?

(c)

At most one clustering index is possible

if $\lceil \log_2 N \rceil + 1$

Select *

from Emp

where Eno = x

$\left\{ \begin{array}{l} \text{Select *} \\ \text{from Emp} \\ \text{where Eno = x} \end{array} \right\}$ 50+

Q2.

N

Select *

from Emp

where ppno = x

$\left\{ \begin{array}{l} \text{Select *} \\ \text{from Emp} \\ \text{where ppno = x} \end{array} \right\}$ 50+

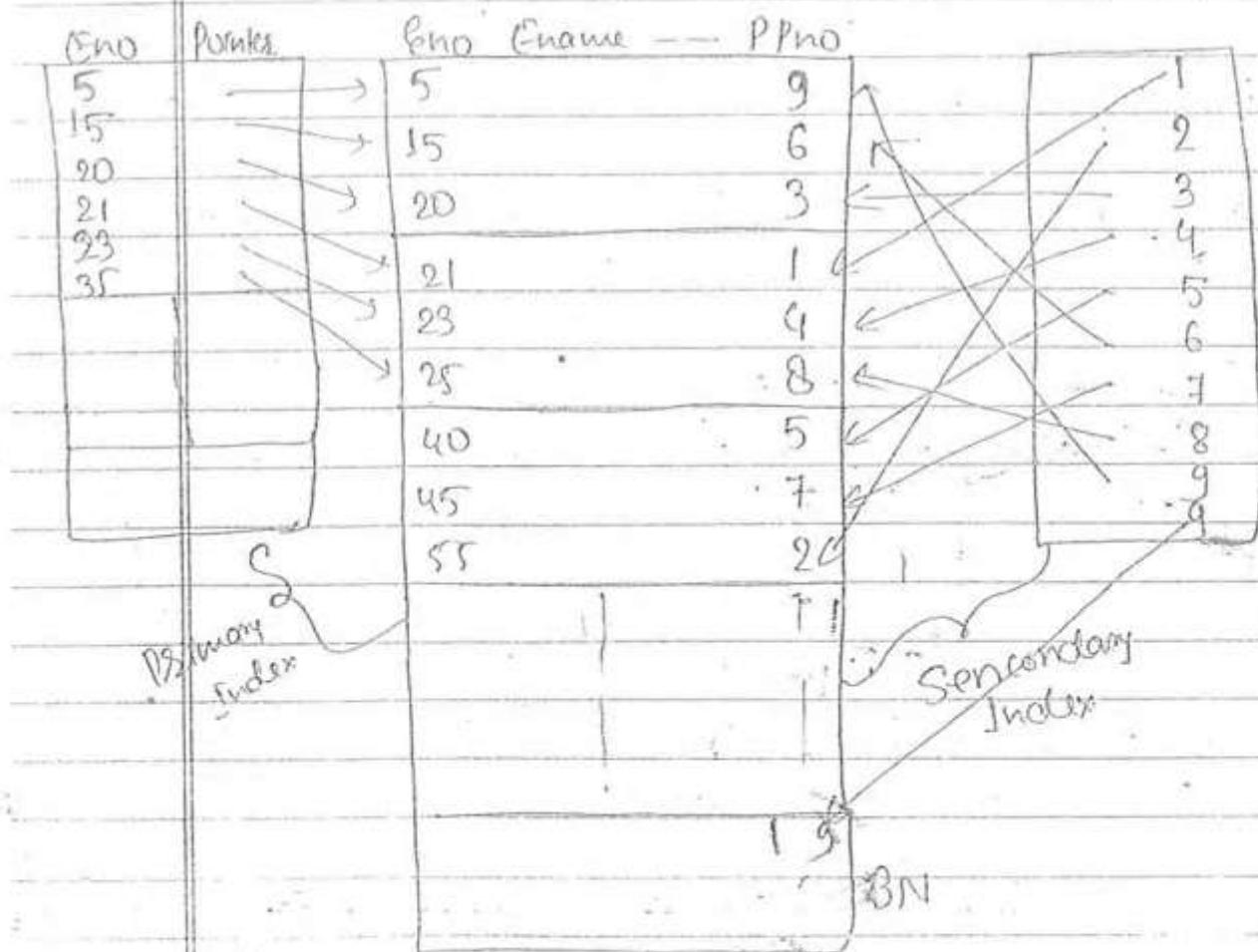
[8] Secondary Index -

Secondary way of accessing the data if primary index or clustering index already exist.

(a) Conditions:-

[1] Databases not physically ordered based on search key used in the index file.
[UN- ORDERED FILE]

[2] Search key can be key or Nonkey.

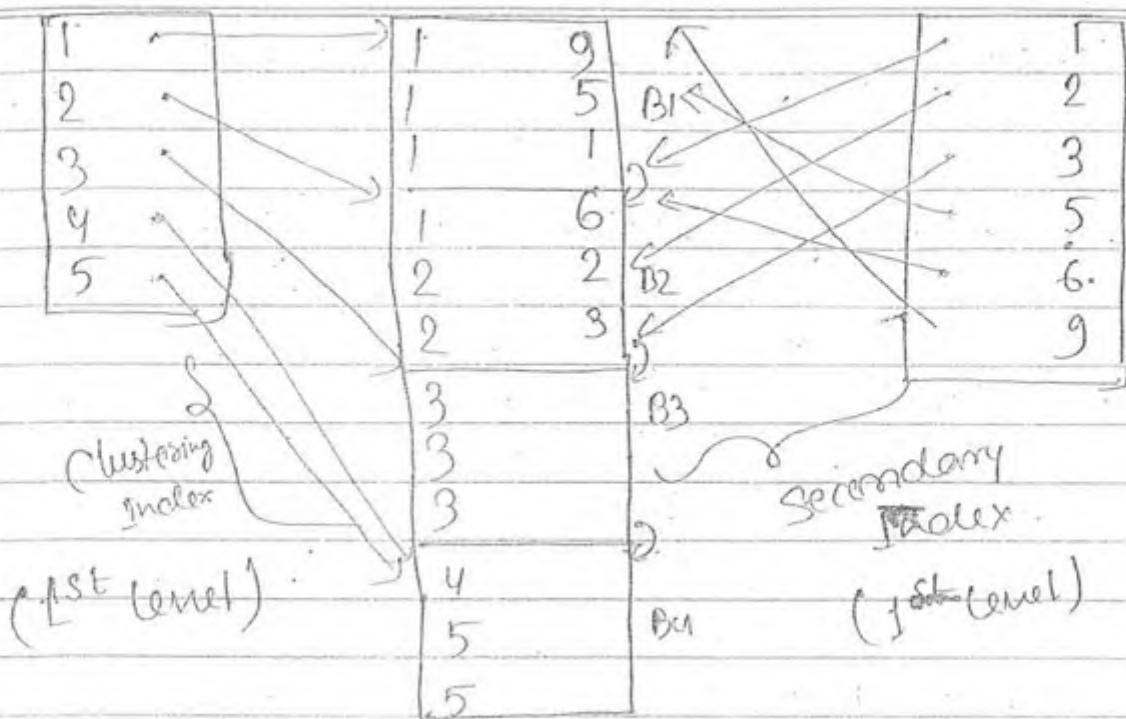


(Index i.e. direct connect with database is called 1st level Index.)

classmate

Date _____

Page _____



(b)

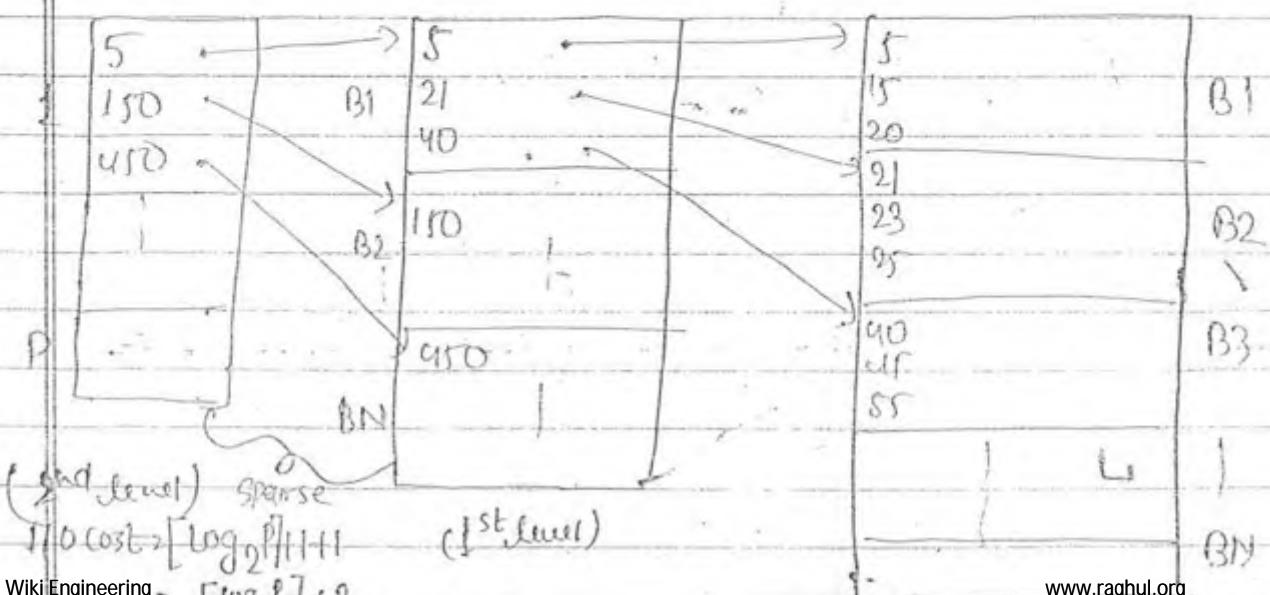
Always Dense Index.

(c)

More than one secondary index possible.

MULTILEVEL INDEX:-

Index to the Index and so on until last level become single block.



- \Rightarrow Entry size of 2nd level Index = Entry size of 1st level Index
- \Rightarrow Block factor of 1st level Index = Block factor of 2nd level Index
- \Rightarrow 2nd level or 3rd level os --- Always Sparse.
- \Rightarrow IO Cost = No. of levels + 1

log 55
in 11)

16384 Records

Record = 32 byte

file size = 6 byte

Enclosed DIA Nonkey

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 DB records
 $= 16384$

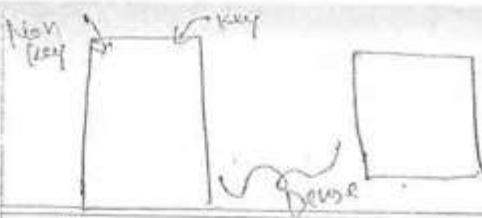
Block Size = 1024 bytes

Entry size = $16384 \times 6 + 10$
 $= 16$

Block factor = $\frac{1024}{16}$

$\frac{1024}{16} = 64$ Entries per block

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



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 =

$$\text{No. of Blocks} = \frac{256}{64} = 4 \text{ Blocks}$$

No. of entries in 3rd level = 4

No. of block = 1

$$\text{No. Blocks} = \{256 + (4+1)\} : \text{Blocks}$$

- Ques) If Ordered on key &
SPARSE Index build on key in the above Ques
then 1) No. of levels = 82
2) No. of Index blocks = 89

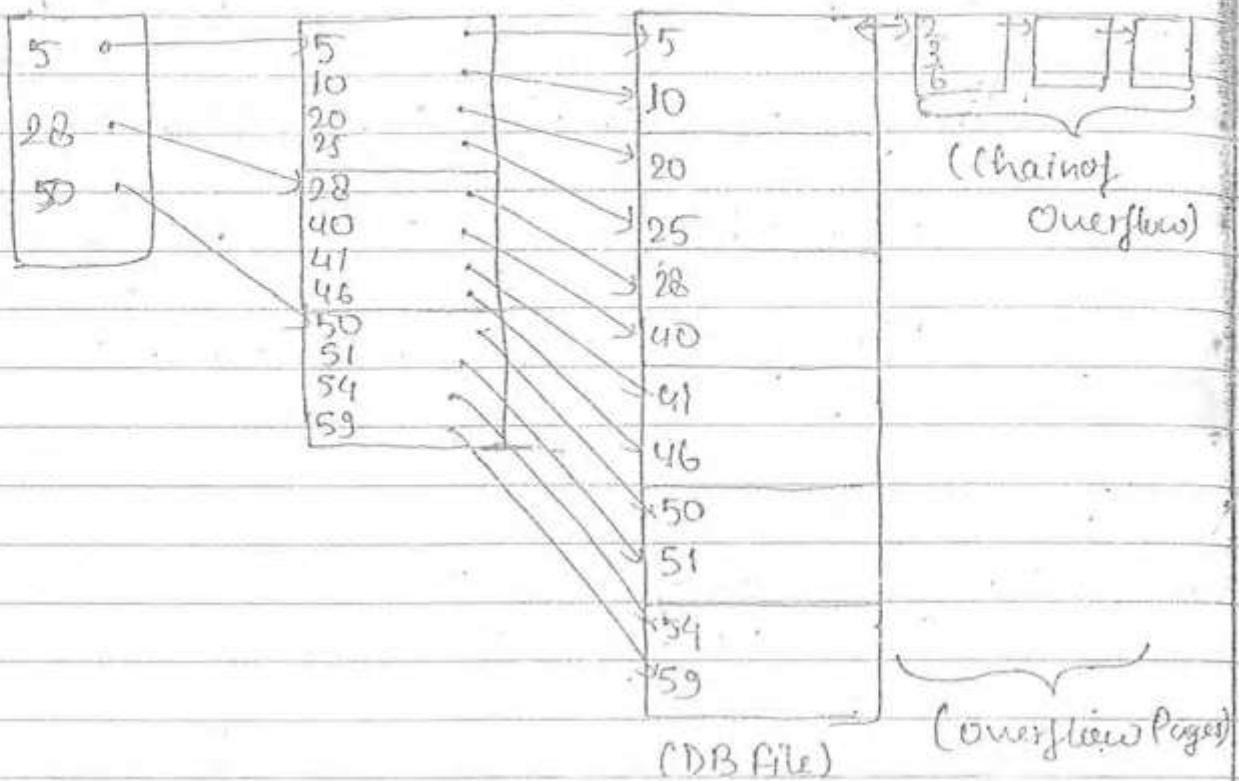
1/09/2011

classmate

Date _____

Page _____

STATIC MULTI LEVEL INDEX



Limitation of Static Multi-level Index

→ Insertion:-

[1] Shifting DBI-

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 DBI-

Which results Entire Index file change.

[2] Leave Space-

Some index block may result 0% occupied (means 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:-

(i) Insertion/ Deletion:-

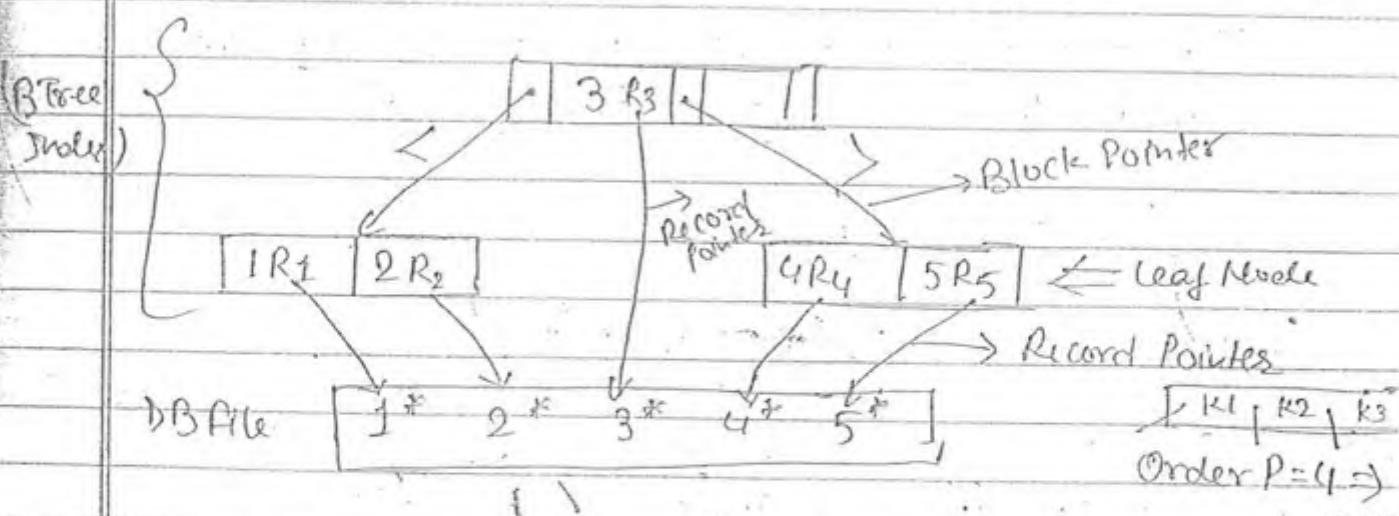
i.e. Insertion/deletion should not result entire Index file changes.

[2] No overflow pages allowed.

[3] Before deletion: No Index Node should be less than 50% occupied.

* B Tree & B+Tree comes under Dynamic Multilevel Index.

B Tree:-



Record Pointers

(R₁)

Data Pointers

(D₁)

Value Pointers

Block Pointers

(B₁)

Node Pointers

(N₁)

Child Pointers

(C₁)

Tree Pointers

4 B P

3 Key

3 R P

P O

I/O Cost To access any Record =

$$\Theta(\log_p N)$$

N: No. of keys. = 5

P: maximum no. of child pointers in
B-Tree Node ≥ 3

To access all records:-

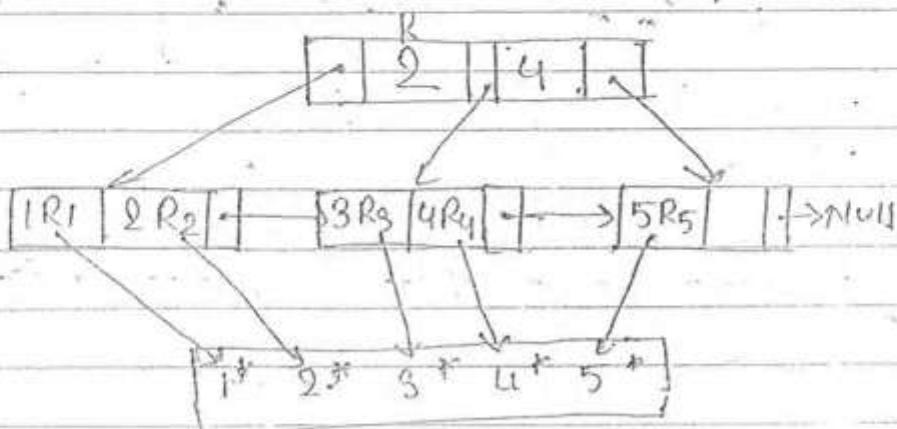
$$\text{I/O cost} = \Theta(2^m)$$

m = No. of Index Nodes.

\Rightarrow Not best Suitable for to access all records of the DB.

To avoid the problem in accessing all records in B tree we goes through B^+ Tree.

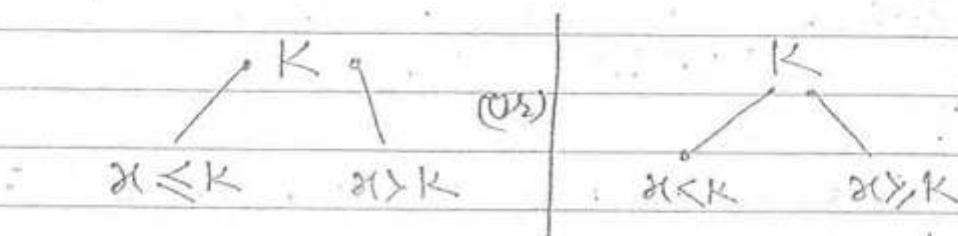
B^+ Tree -



\Rightarrow Every key is stored at Leaf Node

\Rightarrow Every leaf Node points to next leaf node.

if K is a key then Block pointers point the value whose keys value are follow below pattern i.e.



\Rightarrow S/O cost to access any record = $\Theta(\log_p N)$

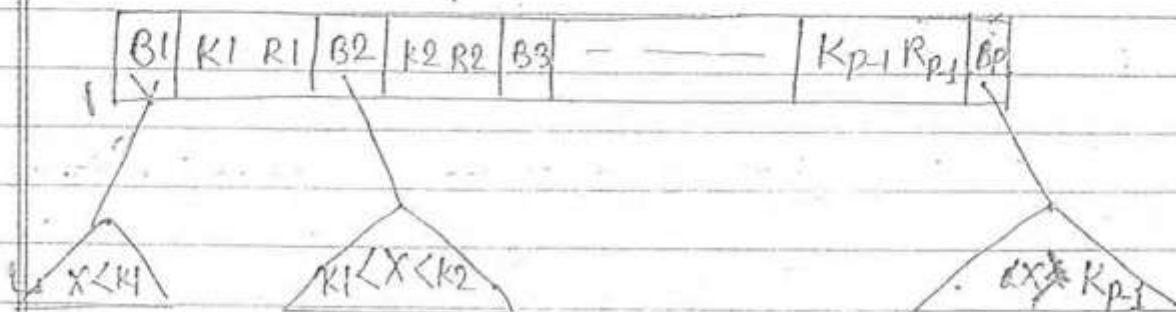
\Rightarrow All records only required leaf Nodes.

B-Tree Definition :-

ORDER P = maximum no. of block pointers in B-Tree nodes

① Structure of Internal Node :-

$P B_p + (P-1) \text{ keys} + (P+1) \text{ Record Pointers} \leq \text{Index Block}$



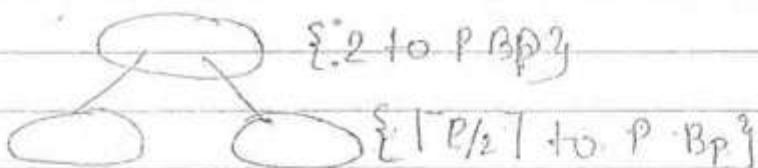
② Structure of Leaf Node:-

B_1	$K_1 R_1$	B_2	$K_2 R_2$	B_3	—	$K_{p-1} R_{p-1}$	B_p
Null	Null	Null	Null	Null	—	Null	Null

③ Keys within the node should be in Ascending Order $\{ K_1 < K_2 < \dots < K_{p-1} \}$

④ Every Node Except Root should be at least $\lceil P/2 \rceil$ Block Pointers at most p block pointers.
(To ensure the 50% occupancy condition).

⑤ Root Node can be at least 2 BP at most P Block pointers.



⑥ Every leaf node should be same level.

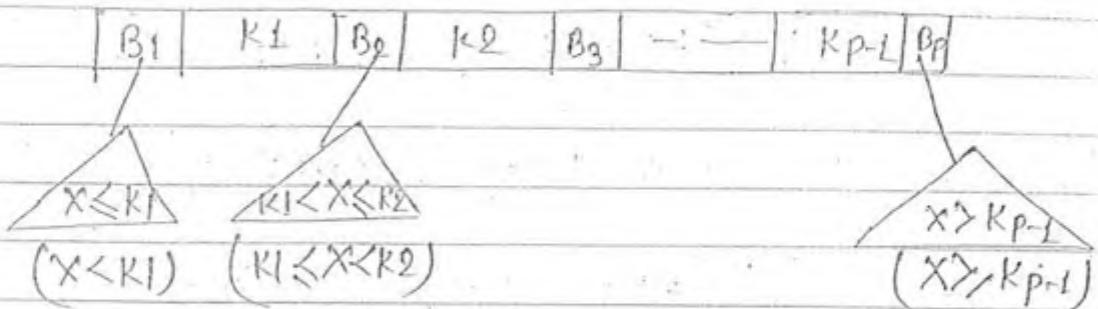
B⁺ Tree Definition:-

① Structure of Internal Node:-

Assume

Order: P = maximum no. of Block Pointers in B⁺ tree internal node

P Block Pointers + $(P-1)$ key \leq Block



② Structure of Leaf Node -

Order P : maximum no. of keys in B^+ Tree leaf Node

P keys + P Record Pointers + 1 BP \leq Block Size



③ Keys within the node should be in Ascending Order: $K_1 < K_2 < \dots < K_{p-1}$

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 0 BP at most P Block pointers.

⑥ Every leaf node should be same level.

(Ques)

Internal node B⁺ Tree order P = maximum no. of keys

$$\therefore (P+1)BP + P \text{ keys} \leq \text{Block Size}$$

Order P! min P keys, max 2P keys in B⁺-Tree internal node.

$$\therefore (2P+1)BP + 2P \text{ keys} \leq \text{Block Size}$$

Order P! min P keys max 2P keys in B-Tree node

$$\therefore (2P+1)BP + P \text{ keys} + 2P RP \leq \text{Block Size}$$

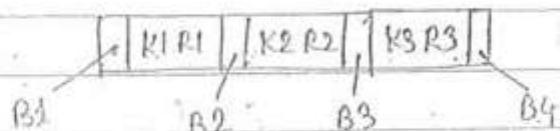
Order P! min P BP max 2P Block pointers in B Tree Node

$$(2P)BP + (2P-1) \text{ keys} + (2P-1) \text{ record pointer} \leq \text{Block size}$$

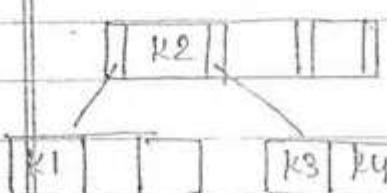
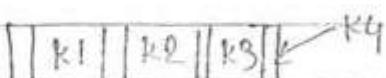
(Ques)

Construct B-Tree with Order P=4 and following sequence of keys.

20, 5, 10, 3, 12, 18, 15, 45, 38, 69, 33



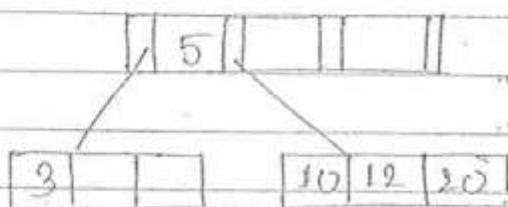
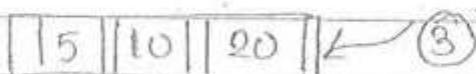
Node Splitting



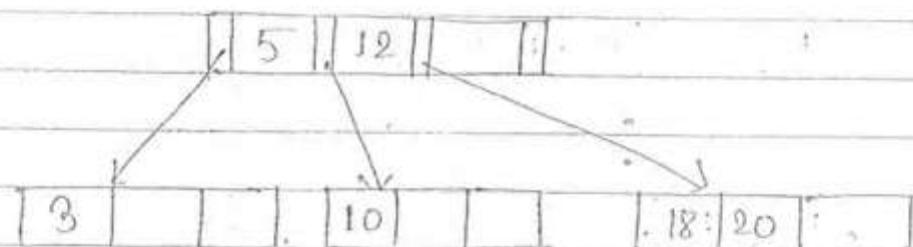


Balanced = Height Restricted =
 $\Theta(\log_p N)$

i.e. height must not go higher than $\log_p N$ for balancing the tree.

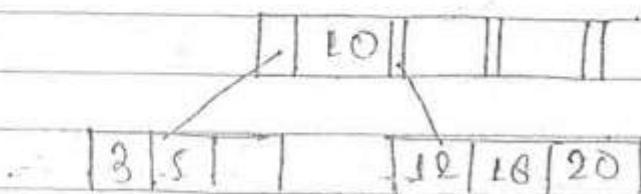


a) Node Splitting:-



b) Redistribution of the keys:-

if any leaf Node consist some free position we can distribute key to accommodate New key in only Existing Nodes.



try redistribution if not possible then go for Node splitting

By using redistribution

5	10	20	4	3
---	----	----	---	---

5					
3		10	12	20	18

10						
3	5	1	12	18	20	15

12						
3	5	10	15	18	20	15

12	10						
3	5	10	15	18	20	45	38

12	10							
3	5	10	18		20	38	45	69

12	20							
3	5	10	15	18	20	38	45	69

19	33							
3	5	10	15	18	20	38	45	69

B-Trees

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	$2 \lceil p_2 \rceil^2$	$2 \lceil p_2 \rceil^3$	$2 \lceil p_2 \rceil^2 (\lceil p_2 \rceil - 1)$
-----	-------------------------	-------------------------	---

$n-1/d$	$2 \lceil p_2 \rceil^{d-1}$	$2 \lceil p_2 \rceil^h$	$2 \lceil p_2 \rceil^{h-1} (\lceil p_2 \rceil - 1)$
---------	-----------------------------	-------------------------	---

(minimum null pointers)

Total min. no. of keys in B-Trees with order P & height h

$$1 + 2(\lceil p_2 \rceil - 1) + 2 \lceil p_2 \rceil (\lceil p_2 \rceil - 1)^{d-1} + \dots + 2 \lceil p_2 \rceil^{h-1} (\lceil p_2 \rceil - 1)$$

$$1 + 2(\lceil p_2 \rceil - 1) \left\{ 1 + 2 \lceil p_2 \rceil + \lceil p_2 \rceil^2 + \dots + \lceil p_2 \rceil^{h-1} \right\}$$

$$1 + 2(\lceil p_2 \rceil - 1) \left\{ \frac{\lceil p_2 \rceil^h - 1}{\lceil p_2 \rceil - 1} \right\} = 1 + 2 \lceil p_2 \rceil^h - 2$$

Total min. no. of keys in B-Tree with order P & level l.

$$= 1 + 2 \left\{ \left[\frac{P}{2} \right]^{l-1} - 1 \right\}$$

(Ques) Total min. no. of keys in B-Trees with order P & level l.

a) $1 + 2 \left\{ \left[\frac{P}{2} \right]^{l-1} \right\}$ { Here height starts from 0 }

b) $P^{l+1} - 1$

c) $P^{2l+1} - 1$

d) $\left\{ \left[\frac{P}{2} \right]^{2l+1} - 1 \right\}$

Height/level	Max. no. of nodes	Maximum no. of Block pointer	Minimum 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/m	P^h	P^{h+1}	$P^h(P-1)$
			(maximum Null pointer)

Total maximum no. of keys in B Tree with Order P & height h.

$$(P-1) + P(P-1) + P^2(P-1) + \dots + P^h(P-1)$$

$$(P-1) \{ 1 + P + P^2 + \dots + P^h \}$$

$$(P-1) \left\{ \frac{P^{h+1} - 1}{P-1} \right\} = \boxed{P^{h+1} - 1}$$

\Rightarrow Total maximum no. of keys in B Tree with order P & level h

$$\boxed{= P^h - 1}$$

Height of the B-Tree with N keys:-

$$N = 1 + 2 \{ \lceil \frac{P}{2} \rceil ^{h-1} \}$$

$$\Rightarrow \boxed{h = \log_p \left\{ \left(\frac{N-1}{2} \right) + 1 \right\}} \rightarrow \text{maximum possible height with } N \text{ keys}$$

$$N = P^{h+1} - 1$$

$$\Rightarrow \boxed{h = \log_p (N+1) - 1} \rightarrow \text{minimum possible height with } N \text{ keys}$$

* In B Tree:- Minimum Null pointers = $\lceil \frac{P}{2} \rceil^h$

Maximum Null pointers = P^{h+1}

* Height of B Tree:-

$$\Theta(\log_p N)$$

where:-

N = No. of keys

P = Orders

Ques)

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 $h=4$

Orders $P=3$ $h=4$

Key:

One block can accommodate :-

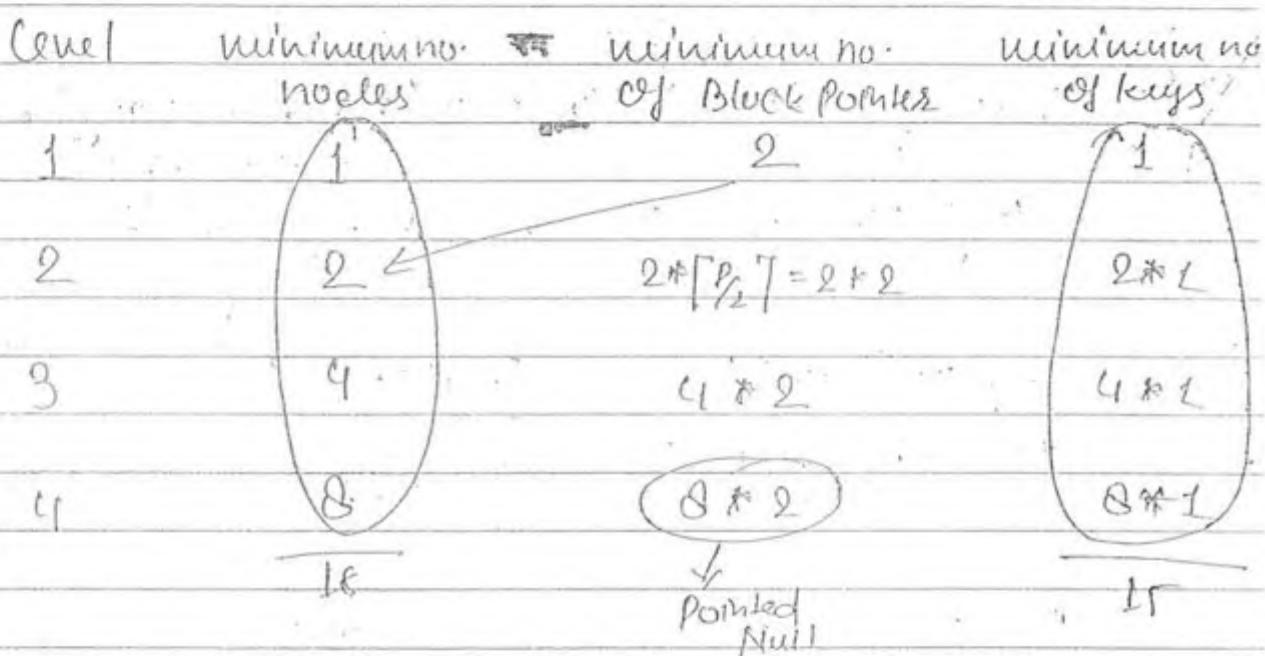
P keys + $(P+1)$ Block pointers + P recordpointer \leq Block

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	$64 = 64$	$64*4$	$64*3$
	<u>85</u>	B.P pointed Null	255

Ques 1) Order P: max no. of Bp in BTree node.
How many minimum no. of keys & Nodes in B-Tree with order P=4, Level=4

One block can accommodate:-

$$(P)BP + (P-1).Keys + (P-1)R.P \leq \text{Block}$$



page-55

Ques 12

P: maximum no. of Tree pointers in B-Tree Node
keys = 10 bytes

Block = 512 bytes

Data Pointers = 8 bytes

Block Pointers = 5 bytes

maximum value of P = ?

$$P + BP + (P-1)Key + (P-1)RP \leq \text{Block size}$$

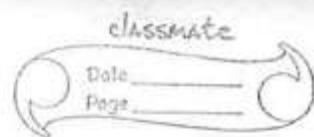
$$P * 5 + (P-1) * 10 + (P-1) * 8 \leq 512$$

$$23P \leq 512 + 18$$

$$23P \leq 530$$

$$P = \left\lfloor \frac{530}{23} \right\rfloor = 23$$

20911



B⁺ Tree :-

Internal Node :-

B_1	K_1	B_2	—	K_{p-1}	B_p
-------	-------	-------	---	-----------	-------

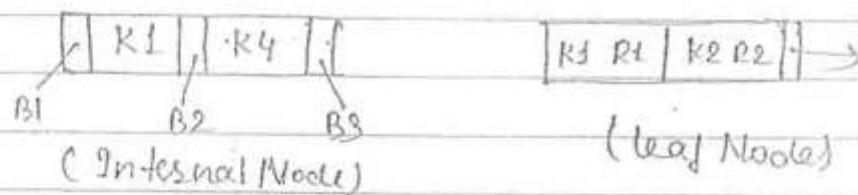
P Block Pointers + $(P-1)$ keys

Page 56

Ques 16)

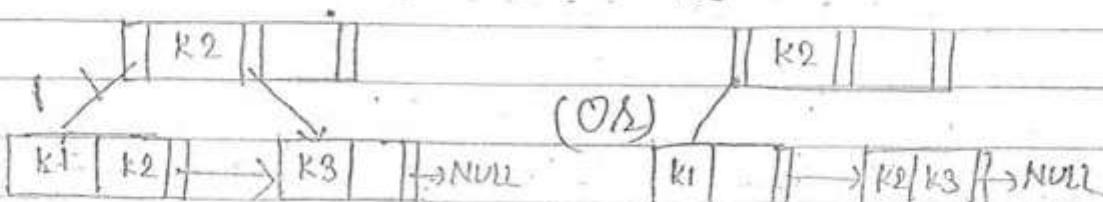
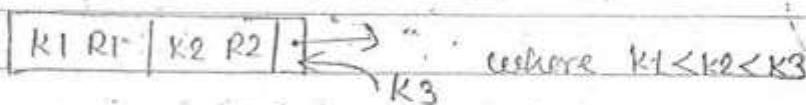
Order of Internal Node = 3 (maximum no. of Tree pointers)

Leaf Node = 9 (maximum no. of data item)



Keys Inserted = 10, 3, 6, 8, 4, 9, 1

NODE SPLITTING IN B⁺ Tree :-



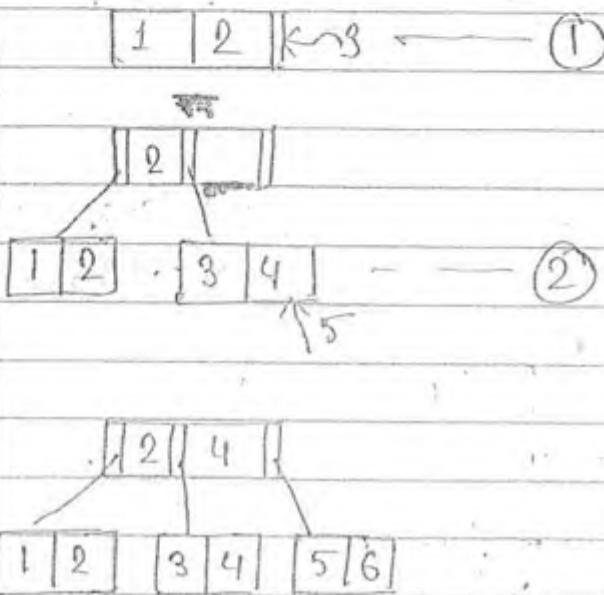
(left biasing)

(Right biasing)

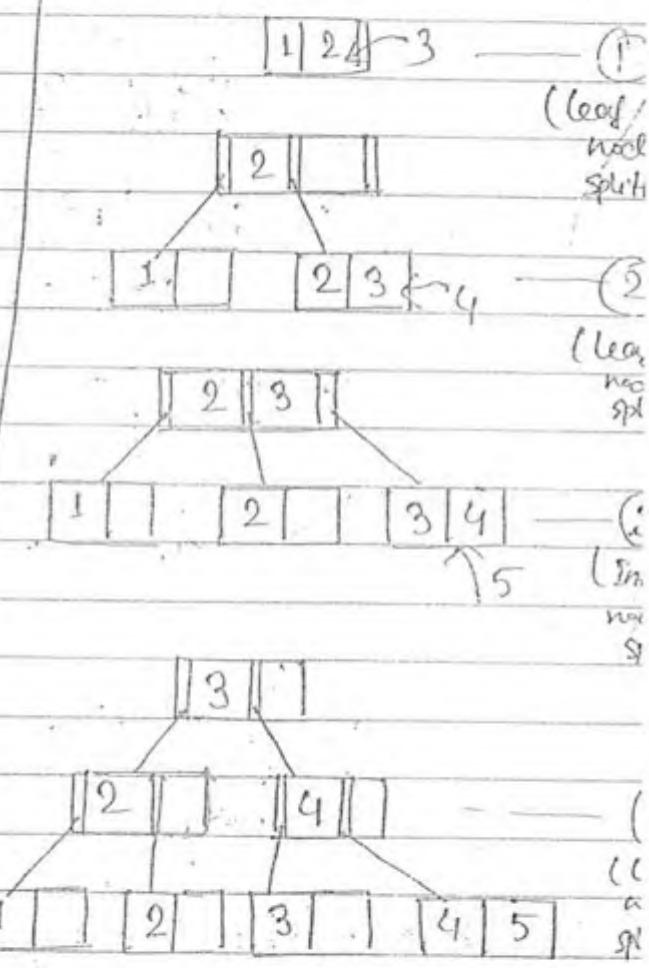
If keys are in Ascending Order

1, 2, 3, 4, 5, 6, _____, N

By left Biasing How many node splitting



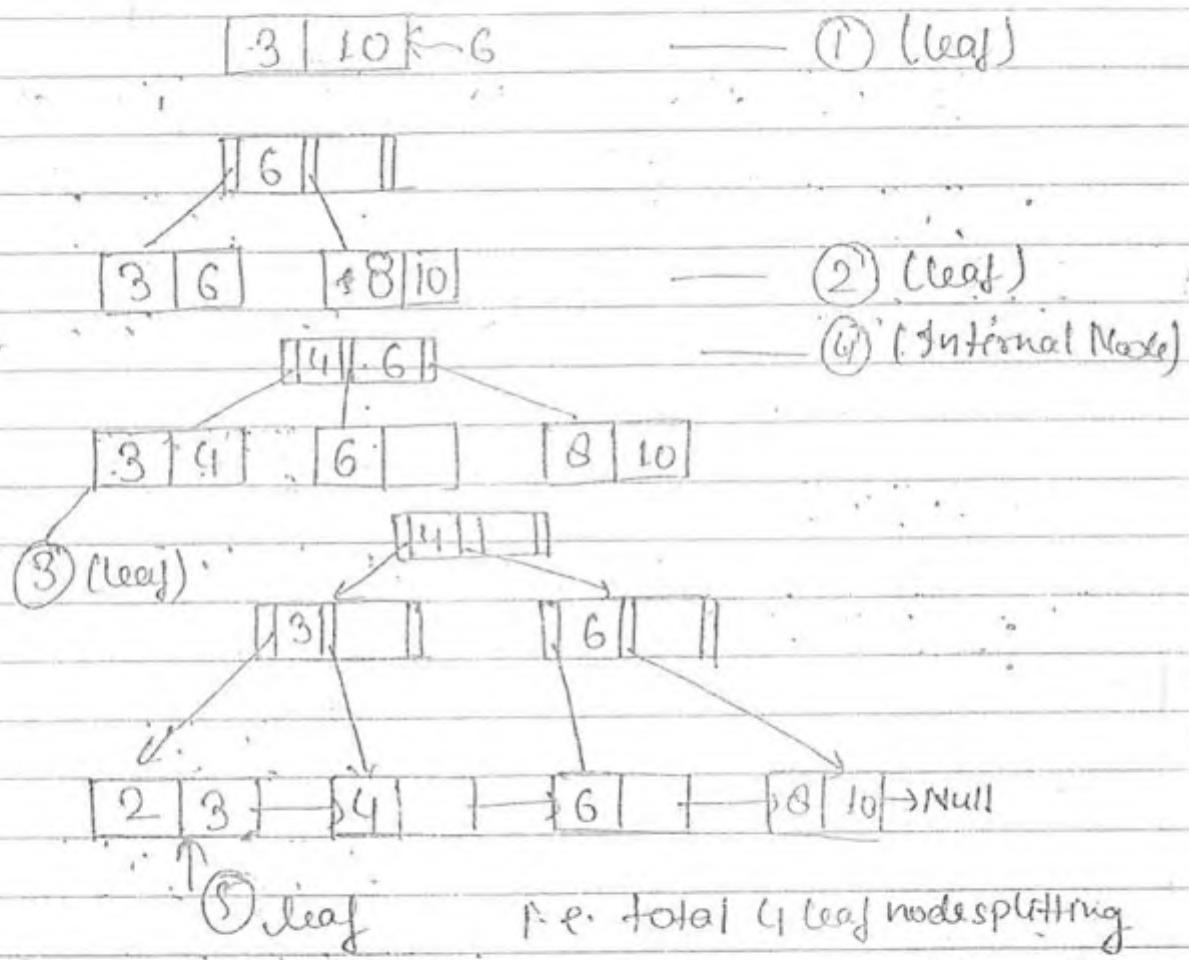
By Right biasing No. of node splitting.



⇒ If keys are in Ascending Order then Right bias gives the more Node splitting then the Left bias i.e. In the above example we see that upto insert 5 element we get 2 node splitting by using left bias and we get 4 node splitting using right biasing.

⇒ If keys are in Descending order then left Biasing gives more Node splitting than the right Biasing.

In the above question 10, 3, 6, 8, 4, 2, 1 more than half elements are in descending order. So, we go through left biasing for obtaining minimum no. of node splitting.

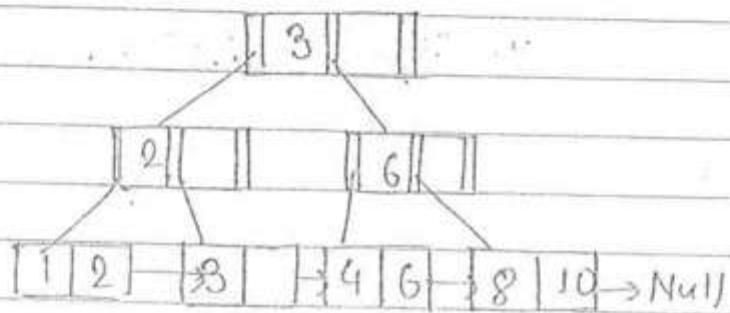
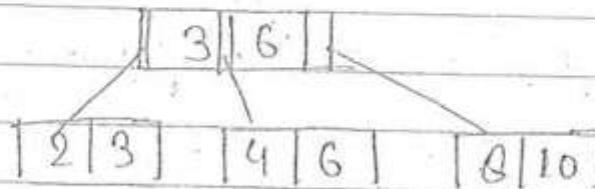
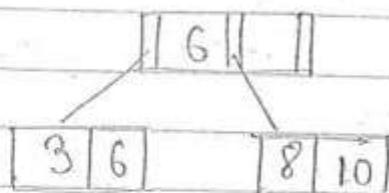


⇒ Redistribution always gives less node splitting. So, whenever question asked like minimum no. of node splitting we always go through Redistribution method.

By applying redistribution method:-

We get 3 leaf node splitting & 1 internal node splitting.

3 10 K (6)



Ques 5)

Keys = 8 bytes

Block Size = 512 bytes

Index Pointers = 4 bytes

$$PBp + (P-1)k \leq \text{Block size}$$

$$P \times 4 + (P-1)8 \leq 512$$

$$4P + 8P - 8 \leq 512$$

$$12P \leq 512 + 8$$

$$12P \leq 520$$

$$\therefore P \leq \frac{520}{12}$$

$$P = \left\lfloor \frac{520}{12} \right\rfloor$$

(Ques)

Record = 5,00,000

Record Size = 200 bytes

Search key Size = 15 bytes

 $B_p = R_p = 5 \text{ bytes}$

Block = 1024 bytes

ORDER of β^+ Tree: → No. of pointers

Internal Node

Leaf Node

$$P B_p + (P-1) \text{keys}$$

$$(P-1) \text{keys} + (P-1) R_p + 1 B_p$$

$$P(5) + (P-1)15 \leq 1024$$

$$5P + 15P - 15 \leq 1024$$

$$20P \leq 1024 + 15$$

$$20P \leq 1039$$

$$\frac{P}{20} \leq 1039$$

(Ques)

 $P B_p + (P-1) \text{keys} \leq \text{Block Size}$

(Ques)

Leaf Node Order P:

Maximum no. of (value, data R_p) pairs it can hold

$$P * \text{keys} + P * R_p + 1 B_p \leq \text{Block Size}$$

(Q13) Largest possible oses of Nonleaf Node (Internal Node)

$$PBp + (P-1) \text{ keys} \leq \text{Block size}$$

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 1000 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.

$$5 \text{ record/block} = 10 \text{ keys & 11 pointers}$$

Ifgo of Data Block

If Index Block

($\lceil \frac{P}{2} \rceil Bp$)

Index Block [10 keys + 11 pts]

min occupancy
(except root)

[$6Bp + 5 \text{ keys}$]

max occupancy
($P Bp$)

[$10 \text{ keys} + 11 \text{ pts}$]

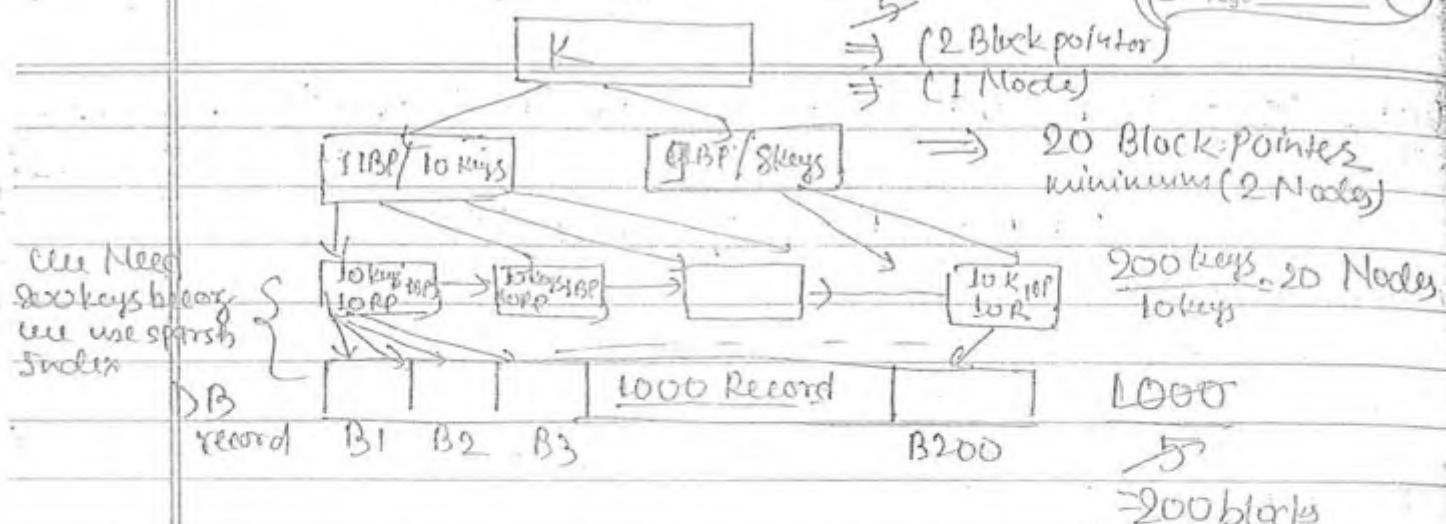
\Rightarrow Smallest No. of index blocks are possible with only Maximum occupancy per Node

\Rightarrow Largest No. of index blocks are possible with only Minimum occupancy per Node

1000 record
Total no. of block = 1000
200 blocks

classmate

Date _____
Page _____



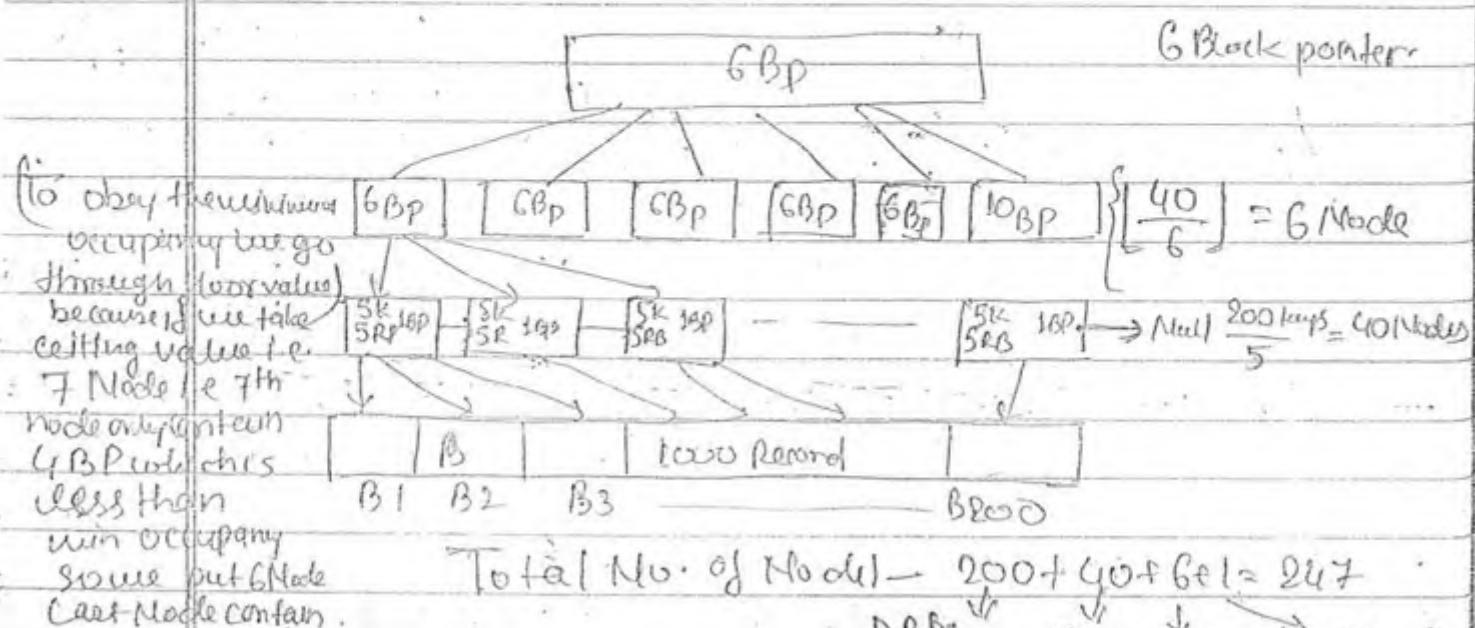
Total No. of Nodes -

$$200 + 20 + 2 + 1 = 223$$

↓ ↓ ↓ ↓
 DataBase 1st Index 2nd Index root level

Largest No. of Blocks + Sparse Index -

Note - Minimum B⁺ Tree Node capacity.



Identify Smallest No. of Blocks for Dense Index:-

10 Block Point
= 1 Node.

$\frac{100}{10} = 10$

100 keys = 100
blocks = 100

100 = 100 Block

Identify Largest No. of Blocks for Dense Index:-

TUPLE RELATIONAL CALCULUS:-

→ It is one of the Non Procedural Query language.

NonProcedural Query language:-

These what to retrieve from DataBase, we least bothered as how to retrieve from DB.

⇒ First Order Logic formula.

⇒ Format of the query $\Rightarrow \{ T / p(T) \}$

(It results tuples T such that $p(T)$ is true)

Example Retrieve the suppliers whose rating greater than 10.

$\{ s / \exists S \in \text{Suppliers} (s.\text{rating} > 10) \}$

Query:- Retrieve the Sid's of suppliers whose rating greater than 10.

$R(A) :- \pi_{\text{Sid}} (\sigma_{\text{rating} > 10} (\text{Suppliers}))$

TR $\{ T / \exists S \in \text{Suppliers} (s.\text{rating} > 10 \wedge T.\text{sid} = S.\text{sid}) \}$

* we can't project bounded tuple variable only free tuple variable can projected.

classmate

Date _____

Page _____

\Rightarrow Bounded Tuple Variable -

Tuple variable set to be bounded only if \exists, \forall used before Tuple Variable

$S \in \text{Suppliers}$
 \uparrow
(Free Tuple Variable)

$\exists S \in \text{Suppliers}$
 \uparrow
(Bounded Variable)

Query Retrieve Sid's of suppliers who supply some red part.

$\pi_{\text{Sid}}(\text{catalog} \bowtie \sigma_{\text{Color}=\text{Red}}(\text{Parts}))$

$\{ T / \exists C \in \text{catalog} (\exists P \in \text{Parts} ((P.\text{Color} = \text{RED} \wedge P.\text{Sid} = C.\text{Sid}) \wedge T.\text{Sid} = C.\text{Sid})) \}$

Query Retrieve Name of the Suppliers who supplied Red or green part.

$\{ T / \exists S \in \text{Suppliers} (\exists C \in \text{catalog} (C.\text{Sid} = S.\text{Sid}) \exists P \in \text{Parts} ((P.\text{Color} = \text{RED} \vee P.\text{Color} = \text{Green}) \wedge P.\text{Pid} = C.\text{Pid})) \}$
1 T.Sname
2 3 S.Sname

Query Retrieve Sid's of suppliers who supplied Red & green part.

$\{ T / \exists C \in \text{catalog} (\exists P \in \text{Parts} ((P.\text{Color} = \text{Red} \wedge P.\text{Color} = \text{Green}) \wedge P.\text{Pid} = C.\text{Sid})) \}$
1 T.Sid = C.Sid

Above query is wrong because we can't take intersection

$\{ \exists T / \exists C1 \text{ Catalog} \left(\exists P1 \text{ Parts} (P1.\text{color} = \text{RED} \wedge P1.\text{Pid} = C1.\text{Pid}) \right) \wedge$

$\exists C2 \text{ Catalog} \left(\exists P2 \text{ Parts} (P2.\text{color} = \text{Green} \wedge P2.\text{Pid} = C2.\text{Pid}) \right) \wedge (C1.\text{Sid} = C2.\text{Sid}) \wedge$
 $T.\text{Sid} = C1.\text{Sid} \}$

Query: Retrieve Sid's of the suppliers who supplied some Red part but not green part.

$U : \vee$

$\cap : \wedge$

$\setminus : \neg$

$\{ \exists T / \exists C1 \text{ Catalog} \left(\exists P1 \text{ Parts} (P1.\text{color} = \text{RED} \wedge P1.\text{Pid} = C1.\text{Pid}) \right) \wedge$

$\exists C2 \text{ Catalog} \left(\exists P2 \text{ Parts} (P2.\text{color} = \text{GREEN} \wedge P2.\text{Pid} = C2.\text{Pid}) \right) \wedge$
 $C1.\text{Sid} \neq C2.\text{Sid} \}$

$T.\text{Sid} = C1.\text{Sid}$

Query1: Retrieve Sid's of the suppliers who supplied every Red part.

$\pi_{\text{Sid} | \text{Pid}} (\text{Catalog}) / \pi_{\text{Pid}} (\pi_{\text{color} = \text{Red}} (\text{Parts}))$

$\Rightarrow \forall t (P(t)) \Rightarrow$ Tuples that are for all tables
if t such that $P(t)$ is true.

$\{ T / \exists C_2 \in \text{Catalog} \left(\forall P \in \text{points} \left(\exists C_1 \in \text{Catalog} \left(P.\text{col} = RGDAC1, P.id = P.P1 \right) \right) \right)$

$\begin{matrix} 1 \\ 1 \\ 1 \end{matrix} \quad \begin{matrix} 2 \\ 2 \\ 2 \end{matrix} \quad \begin{matrix} 3 \\ 3 \\ 3 \end{matrix}$

$1 (C_1.Sid = C_2.Sid)) \quad 1 T.Sid = C_2.Sid \quad \begin{matrix} 4 \\ 4 \\ 4 \end{matrix}$

$\{$ For all which is equal
to division $\}$

Query $\{ S / \exists T (S \in \text{suppliers}) \}$ Free variable

\Rightarrow O/P! Infinite ! retrieve tuples those are not
belongs to suppliers table.

\Rightarrow Unsafe Query. (means results of Query infinite)

$\{ T / \exists S \in \text{suppliers} \ 1 T.Sid = S.Sid \}$

\Rightarrow O/P! empty

\Rightarrow Safe Query

* Basic Relation Algebra Expressive Powers =

Safe TRC Expressive Power

SQl

\Rightarrow All $\sqsubset \sqsubseteq$ 3

All \geq ANY

ALL

EXISTS ()

} it only checks inner condition
is True or False.

\Rightarrow INNER JOIN : NATURAL JOIN

CROSS JOIN : Cross Product

UNION JOIN : RUS

A	B	Δ	A . B	=	A . B
1	2	Δ	2 2	=	1 2
2	1	Δ	2 2		2 1

Δ = empty

\Rightarrow ODBC - (Open Data Base Connectivity) (Microsoft)
JDBC - Java (Also, RDBMS) (Sun Microsystems)

C Program / Java

ODBC/JDBC

Data Base

Relational Query

(1) Take less time than

to execute the query

$$10^9 \Rightarrow ?$$

$$(10)^9 \times \log_{10} 10^9 C = 10^9 \times \log_{10} 10^9 \times \frac{1}{3000} \text{ ms.}$$

$$= 10^9 \times 9^3 \times \frac{1}{300} \Rightarrow 3 \times 10^7$$

~~= 3000,000,0~~

~~Data Structures~~

Data STRUCTURE

170