

DBMS

(9)

४३

४४

18/08/2011

Contents:-

⇒ Introduction & Integrity Constraints.

Referential Integrity Constraints
ER model.

⇒ Schema Refinement

(Normalization)

[2+2] marks

⇒ Transactions & concurrency control.

[2] marks

⇒ Query lang

→ Relational Algebra

→ Tuple Relational Calculus

→ SQL

[2+2] marks

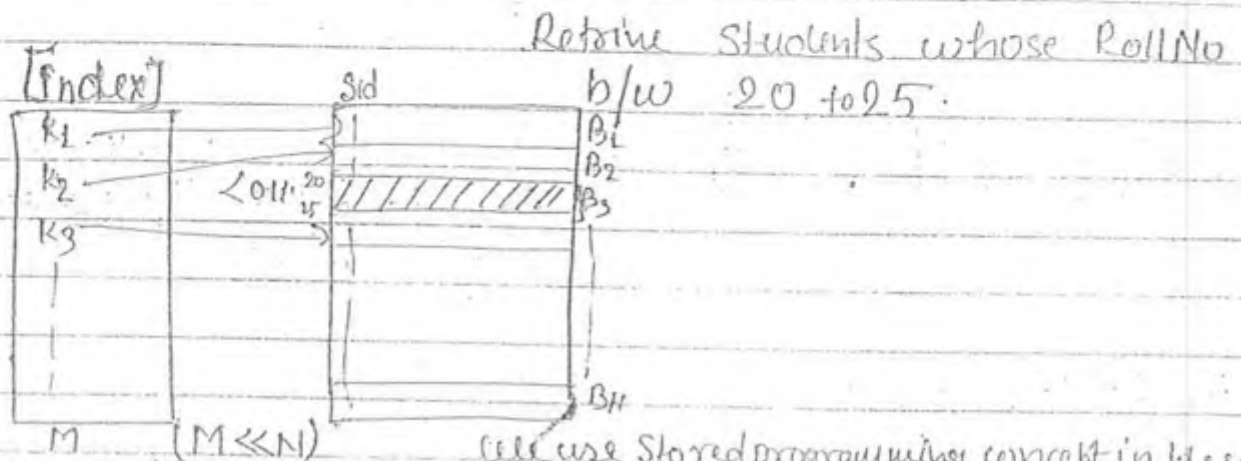
⇒ Indexing & File Organization [6-8]

[2] marks

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

Example:-

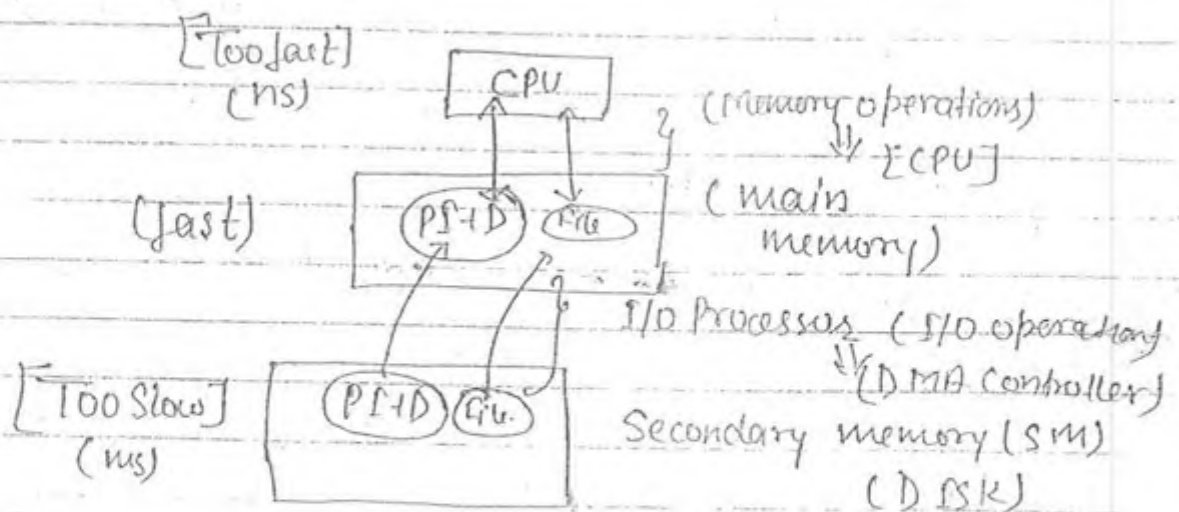
(10MB) ← {Student.txt} 500MB (Disk)



We use Stored programming concept in file system as well as in DB.

Stored Programming Concept:-

Program + Data Required to execute program should be stored in Highspeed main memory in-order to execute by CPU.



I/O Cost:-

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

In Worst Case

I/O Cost N blocks.

Use of Indexing:-

If we have a book of 500 pages

I/O cost

Without Index 500

500

30 index pages + 500

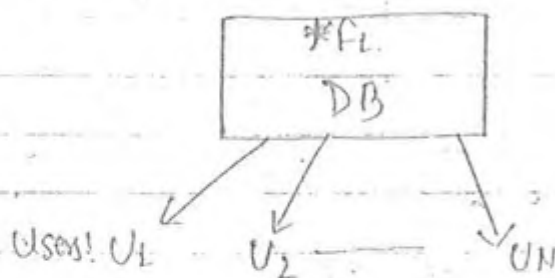
30 + 1

To access + current p

3) Concurrent Access:-

we can't
Inherent when Write $\leftarrow$ Read/Write
when Read $\leftarrow$ Write
we can't

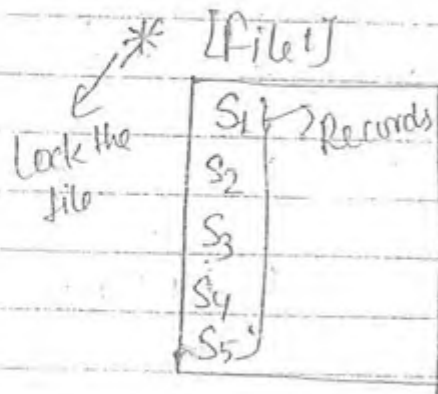
* less concurrency level becoz of file level locks.



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

Operating system lock the ^{entire} file at the time of updation.

Drawback of file level concurrency control:-



If user U₁! update S₁

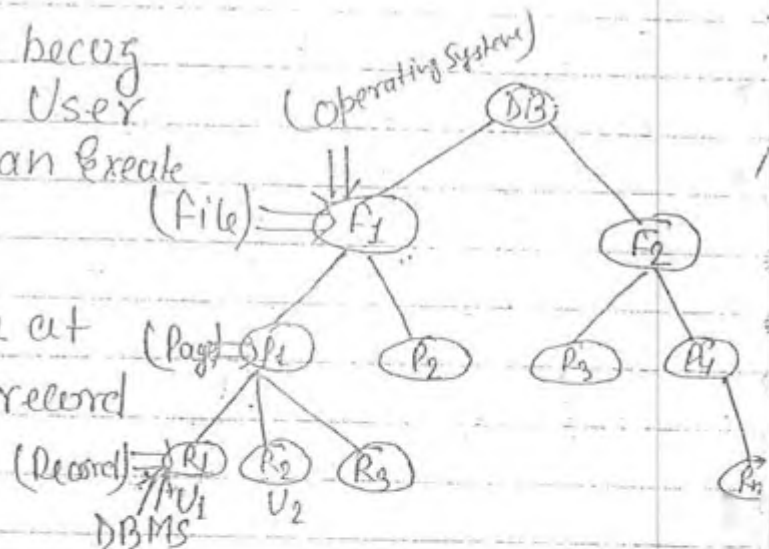
U₂! Read S₃

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

Benefit of DBMS in concurrency:-

* Record level locking becoz
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 ^{of this} at file level
concurrency decreases.

* Low level lock means more concurrency i.e. DBMS
uses low level locking because ^{of this} 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)

OP system needed ~~two~~ create two file for "Faculty" & "Admin" who contain data item as (Sid, Sname, Marks) & (Sid, Sname, Marks, Address). Using this method Redundancy causes. So, operating system is not applicable for security.

Advantages of DBMS:-

VIEWS

[Virtual Table]

View Studfac (Sid, Sname, Marks)

Stud Info Admin (Sid, Sname, Marks, Address)

File:-

(Sid	Sname	Marks)	DOB	Address	PhoneNo
------	-------	--------	-----	---------	---------

One of the Data Model:-

Relational DBMS:-

Every data should be stored in proper Tabular format.

Table:-

Collection of rows and columns.

* DBMS take the data in proper format and every column there should be a unique name.

Relational Instance (Set of tuples in the table) ← Record (or) tuple

Sid	Snme	Branch
S ₁	A	CS
S ₂	A	IT
S ₃	B	CS

Key! Sid
Key! Sid Snme

Field (or) Attribute

Asity (or) Domain } No. of attributes in the Table.

It is a table with 3 fields or 3 attributes.

Attribute Range:-

possible values accepted by attribute.

Cardinality:-

No. of Records of table.

Abstraction of Table

(or)

Relational Schema

Student (Sid Snme branch)

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

Code Rule:-

Every tuple should be differentiate from other tuple.

(Or)

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

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

Key:-

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

* Keys ^{may} contain more than one attributes

Example:-

Sid	Cid	---
S ₁	C ₁	
S ₁	C ₂	
S ₂	C ₁	

(Sid Cid) ! Key

⇒ Composite key:-

Key forms with two or more attributes.

⇒ Simple key:-

Key forms with single attrib.

* A relation may contain more than one key.

Example:-

Sid	Sname	PassPho (PPno)	LicenceNo (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			

Assume Constraints:-

No two students with same DOB & Fname.

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

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

i.e.

if ABC is key then { A is not key, B is not key, C is not key, AB is not key, BC is not key, AC is not key }
 Proper subset of ABC is not allowed as key because ABC is a key.
 i.e. if (DOB Fname) is key then { DOB is not key, Fname is not key, DOB & Fname together uniquely differentiate the tuple then only they are keys. }

* All the keys collectively called as Candidate Key.

Candidate key :- { Sid, PPho, Lno, (DOB Fname) }

* Candidate key is same as key
 i.e. Sid is one of the C-key of PPho, Lno, (DOB Fname) & all 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 nat key

Candidate key

(Primary key)

(Alternative key)

Primary key

Alternative key

⇒ Primary key not allowed Null Value.

⇒ Allowed Null Value.

Null: Unknown (vs)
unexisted Value.

(integer)						(integer)	
Index	Sid	Sname	PPho	Lno	DOB	Pno	
	S ₁		Null	L ₄	10/10		✓
	S ₂		P ₄	L ₃	10/10		✗
	S ₃		Null	L ₁	11/12		✗
	S ₄		P ₃	L ₂	Null		

↓ Unexisted Value
↓ Unknown

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. Numerical values gain more priority to design Primary key.
- * If all the three constraints are satisfied by two candidate key then go for next constraints i.e. Candidate key with less no. of attributes.

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

Table Creation Command:-

```
CREATE TABLE Student (
```

```
    Sid Varchar(10) Primary key,
```

```
    Sname Varchar(30) NOT NULL,
```

```
    PPno Varchar(15) UNIQUE,
```

```
    Lno Varchar(10) UNIQUE NOT NULL,
```

```
    DOB date NOT NULL,
```

```
    Fname Varchar(30) NOT NULL,
```

```
    Unique (DOB, Fname));
```

Superkey:-

Set of attributes used to differentiate tuples of the relation.

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

Example:-

Sid

Sid Sname

PPno

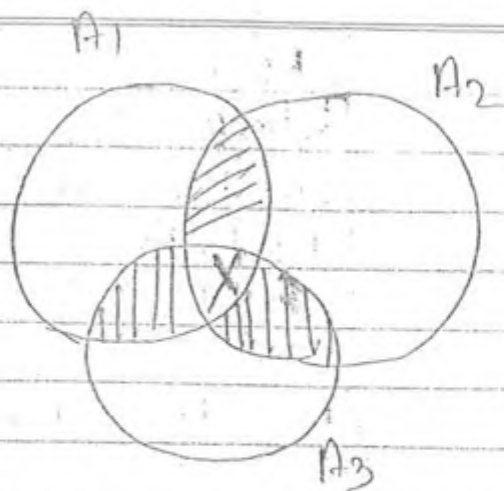
PPno Lno

} Superkeys

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

* Every candidate key is ~~only~~ Super key.
Reverse is not true.

e)



$$\begin{array}{ccccccc}
 2^{n-1} & + & 2^{n-1} & + & 2^{n-1} & - & 2^{n-2} & - & 2^{n-2} & - & 2^{n-2} & + & 2^{n-3} \\
 \times & & \times & & \times & & \times & & \times & & \times & & \times
 \end{array}$$

Ans

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

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

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

Que 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 1+ (Enrolled)

1) Insertion - (May causes 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.

Ques)
Delete

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

A! Primary key

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

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

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

Ques)

geq: greater than or equal to

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

CREATE geq

(lb integer not null,

Ub integer not null,

primary key lb,

Foreign key (Ub) References geq on delete cascade

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

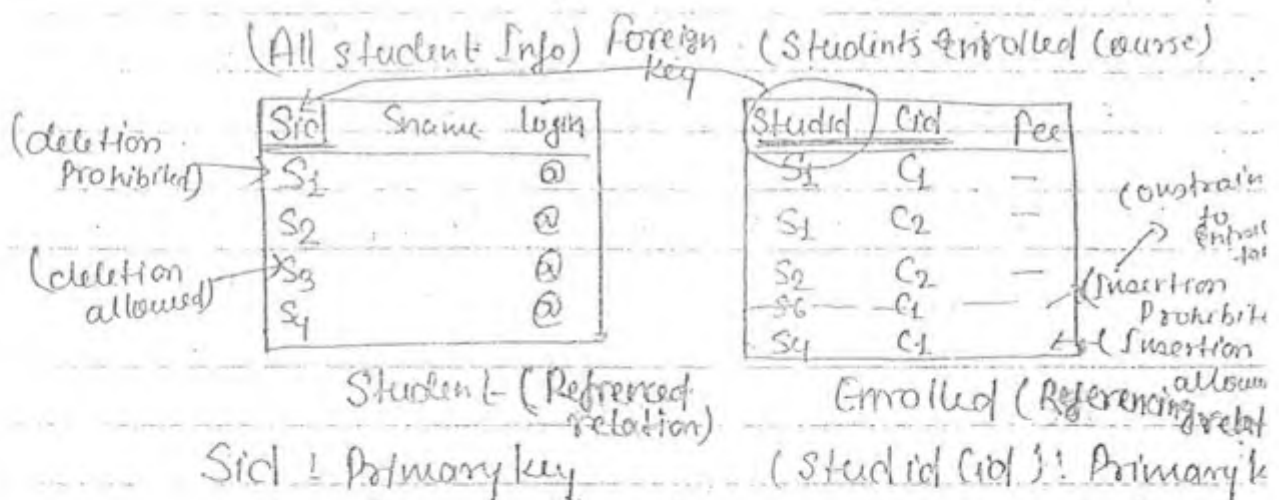
a) a tuple (z,w) with $z > y$ is deleted

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

Soln:- if $R(ABC)$

$\left. \begin{array}{l} A \\ B \\ C \\ AB \\ BC \\ AC \\ ABC \end{array} \right\}$ if every attribute is candidate key
 then no. of super keys
 $2^n - 1$ (we subtract 1 beca
 ϕ is not a super
 key)

FOREIGN KEY:-



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

Creation of Enrolled Table:-

```

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


Example for foreign key reference the same table:-

Foreign key

<u>Eid</u>	<u>(Sup-id)</u>
E ₁	Null
E ₂	E ₁
E ₃	E ₂
E ₄	E ₃

Here Sup-id reference the Eid means sup'id must belong to Eid i.e. no ~~other~~ ^{one} member not belongs to Eid is in Sup'id. i.e. every member of Sup-id must belongs to Eid.

* Referential key is also called foreign key.

Referential Integrity Constraints:-

* Referenced Relation :- (Student)

1) Insertion :- No violation.

2) Deletion :- (May cause ~~integrity~~ 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 prohibited deletion

If foreign key is on the same table then

<u>Eid</u>	<u>Sup-id</u>
E1	Null
E2	E1
E3	E2
E4	E3

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

c) ON DELETE SET NULL

first 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. If allowed then after delete the tuples from Referenced Relation otherwise not possible. Foreign key

Example:-

ON DELETE SET NULL

	<u>Eid</u>	Sup- id
If E1 is deleted	E1	NULL
	E2	E1
	E3	E2
	E5	E1
	E4	E3

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

3) Update:- (Referenced Attribute Set Update)
may cause Referential integrity violation.

- a) ON UPDATE NO ACTION (default)
 - b) ON UPDATE CASCADE
 - c) ON UPDATE SET NULL
- Mostly used update
less used update

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

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

Solⁿ:-



if (x, y)
 $(4, 4)$

(z, w)
 $(3, 4)$
 $(2, 3)$
 $(1, 2)$

then possibility is that $w < x$

- * In S/w design
 - Requirements
 - Design "UM2" 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 -

Example -

Sid

Sname

(Cname)

MVA

S₁

A

C/C++

S₂

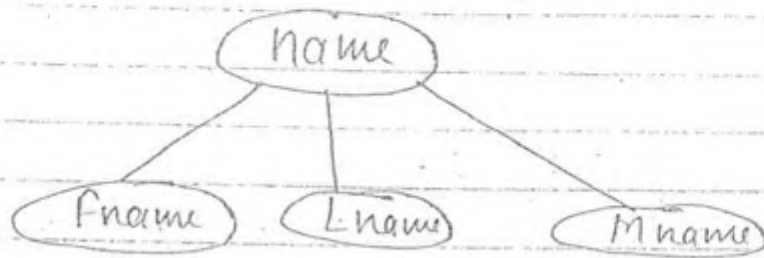
A

C++/Java

represented as

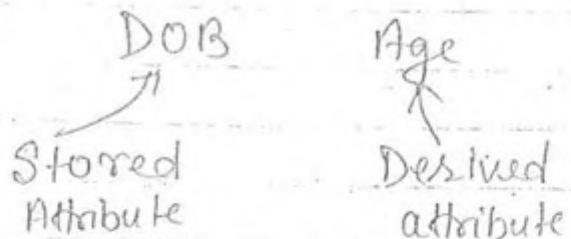


Compound attribute (Composite Attribute) - further subdivided into two or more attributes. Such kind of attribute is called compound attributes.



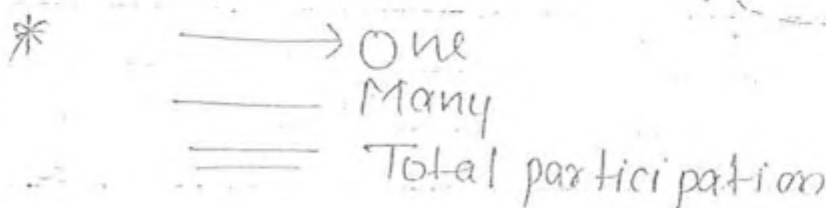
Derived Attribute -

Value of derived attribute derived from other stored attribute.



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

It is represented as:



19/08/2011

Relationship Set:-

Used to relate two or more entity sets.

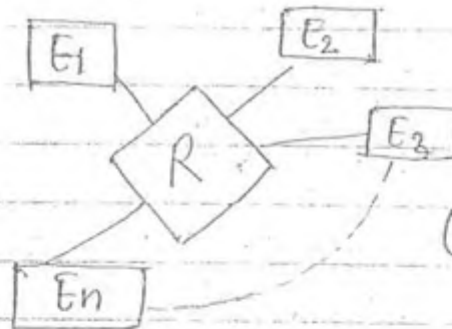
Symbol:-



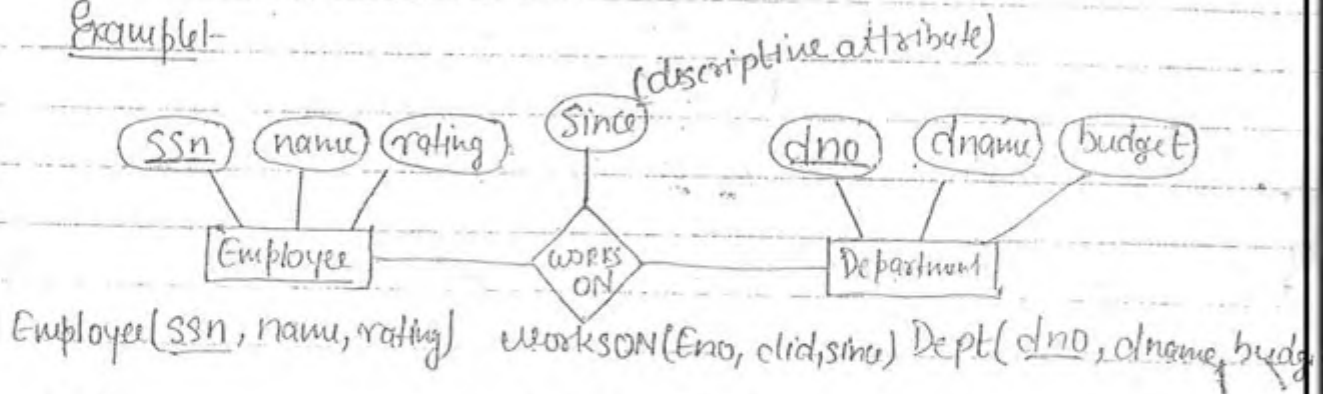
represents the Relationship

* 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 attribute of Relationship Set



Example:-

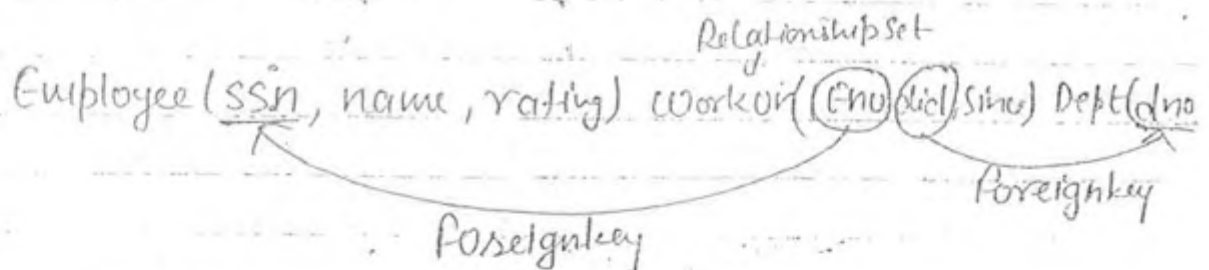


Constraints of Entity Set

- 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),

did varchar(15),

since date,

primary key (_____)

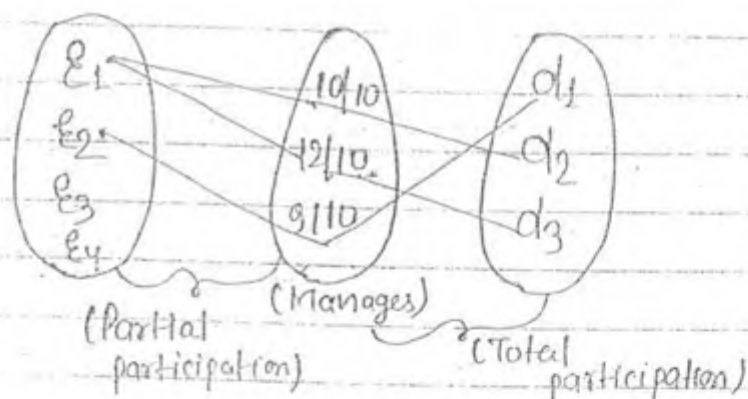
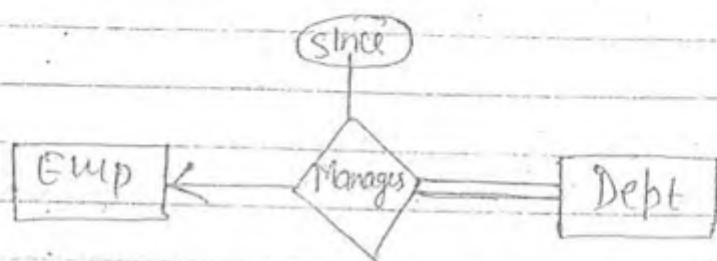
FOREIGN KEY (ENO) Reference Employee ON DELETE (CASCADE)

FOREIGN KEY (DID) Reference Dept ON DELETE (CASCADE).

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

* Relationship Set Constraints:-

1) Participation:-



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

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

* Foreign key 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 don't allow Eno & did as duplicate
i.e. both the entity should be key for One to One cardinality.

<u>Eno</u>	<u>did</u>
E ₁	d ₁
E ₂	d ₃



Candidate keys:
A, B

b) One to Many:-

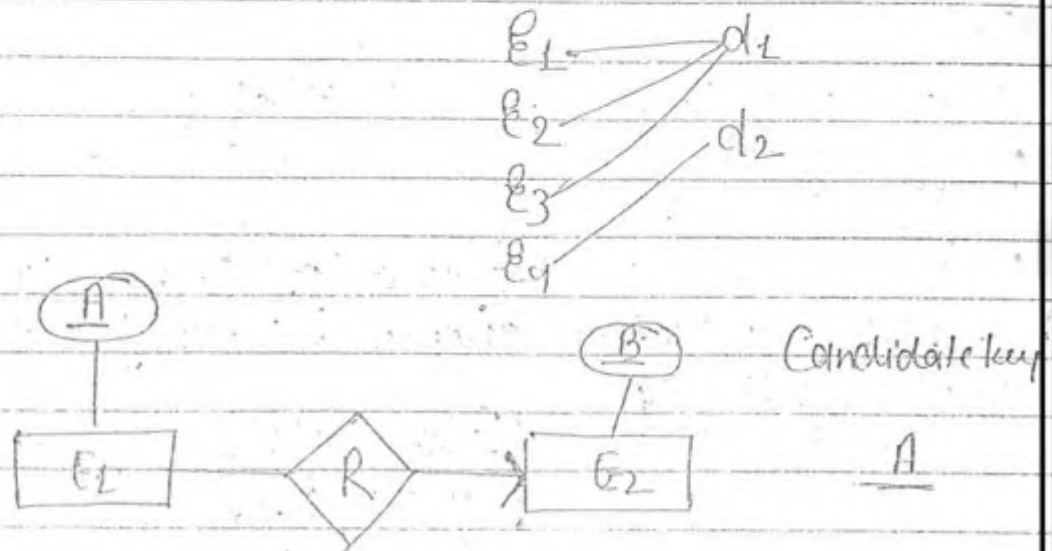
In one to many cardinality did should be primary key.

<u>Eno</u>	<u>did</u>
E ₁	d ₂
E ₂	d ₃
E ₂	d ₁



Candidate key of R:
B

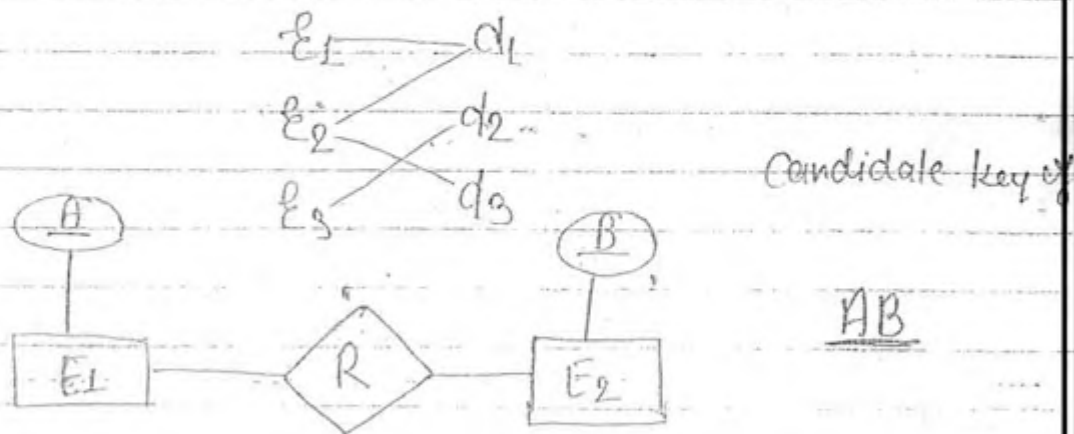
c) Many to One:- In Many to One $\underline{eno}$ should be primary key.



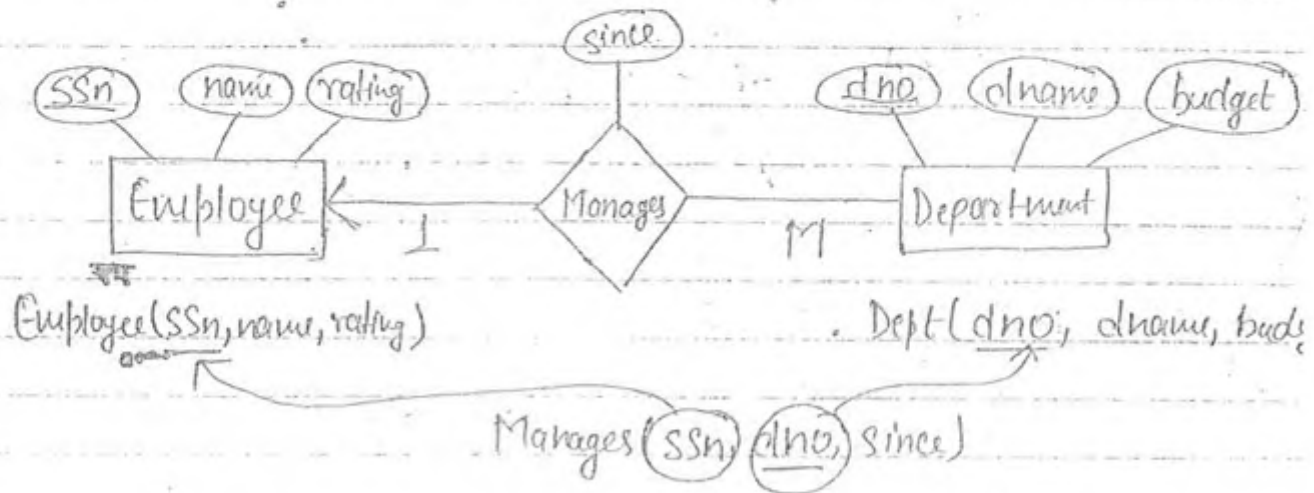
d) Many to Many:- (1:M & N:1)

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

Here in both $\underline{eno}$ & $\underline{d}$ duplication is possible.



Example:-



Primary key : dno

foreign key : SSN
dno

After merging the manages table & Department Table:

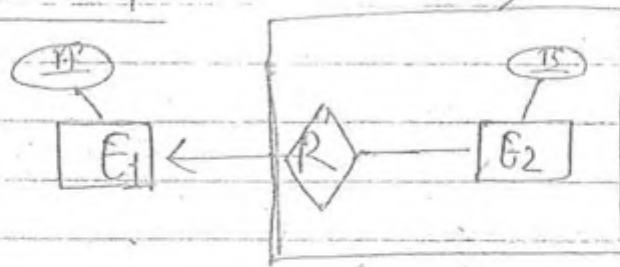
Dept.Manages(dno, dname, budget, SSN, since

foreign key (SSN) Reference Emp.

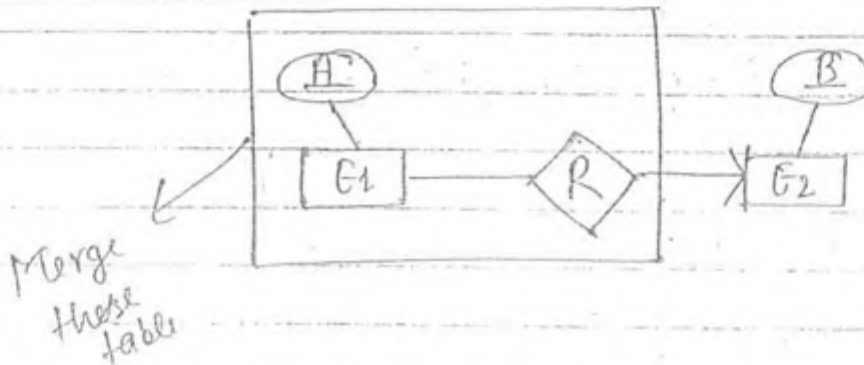
Minimization of ER Model:-

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

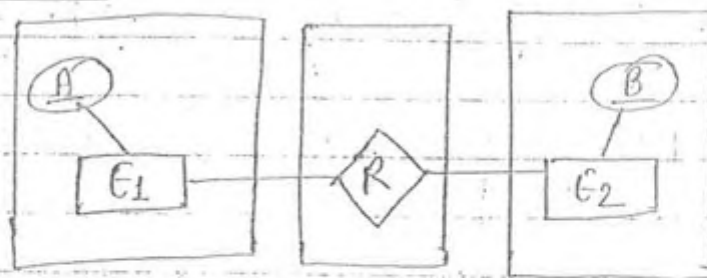
In One to Many:-



In Many to One:-



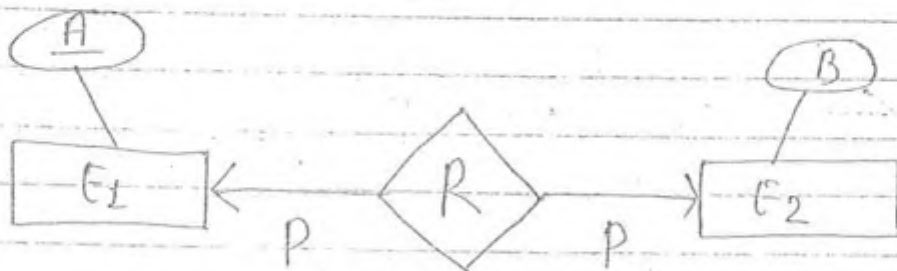
Many to Many:-



* In many to many we always need 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:-



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

<u>A</u>	A ₁	A ₂
1	a ₁	a ₂
2	a ₁	a ₂
3	a ₃	a ₄
4	a ₄	a ₄

Partial

<u>A</u>	<u>B</u>
1	13
2	12

Partial

<u>B</u>	B ₁	B ₂
11	b ₁	b ₁
12	b ₁	b ₂
13	b ₁	b ₂

$(E_1 R)$

<u>A</u>	A ₁	A ₂	<u>B</u>
1	-	-	13
2	-	-	12
3	-	-	Null
4	-	-	Null

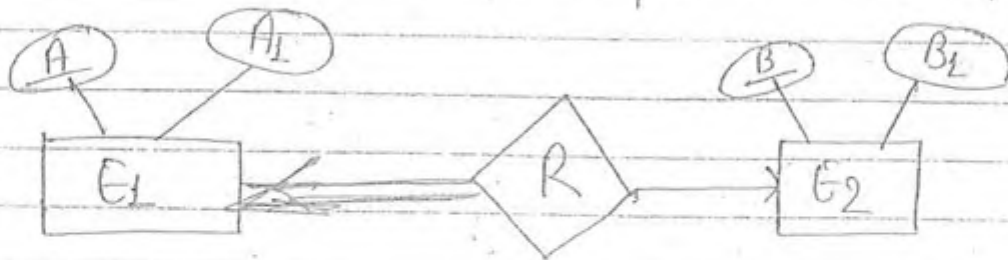
$R E_2$

<u>B</u>	<u>A</u>	B ₁	B ₂
11	Null	-	-
12	2	-	-
13	1	-	-

A: Primary key B: alternative key

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

Assume - Total participation exists b/w E_1 & R .



E_1	
A	A_1
1	a_1
2	a
3	a_2

R	
A	B
3	13
2	11
1	12

E_2	
B	B_1
11	b_1
12	b_1
13	b_2
14	b_2

A	A_1	B	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

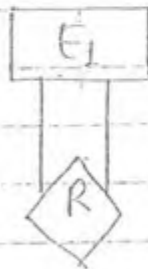
B : Primary key
 A : Alternative key

* 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.

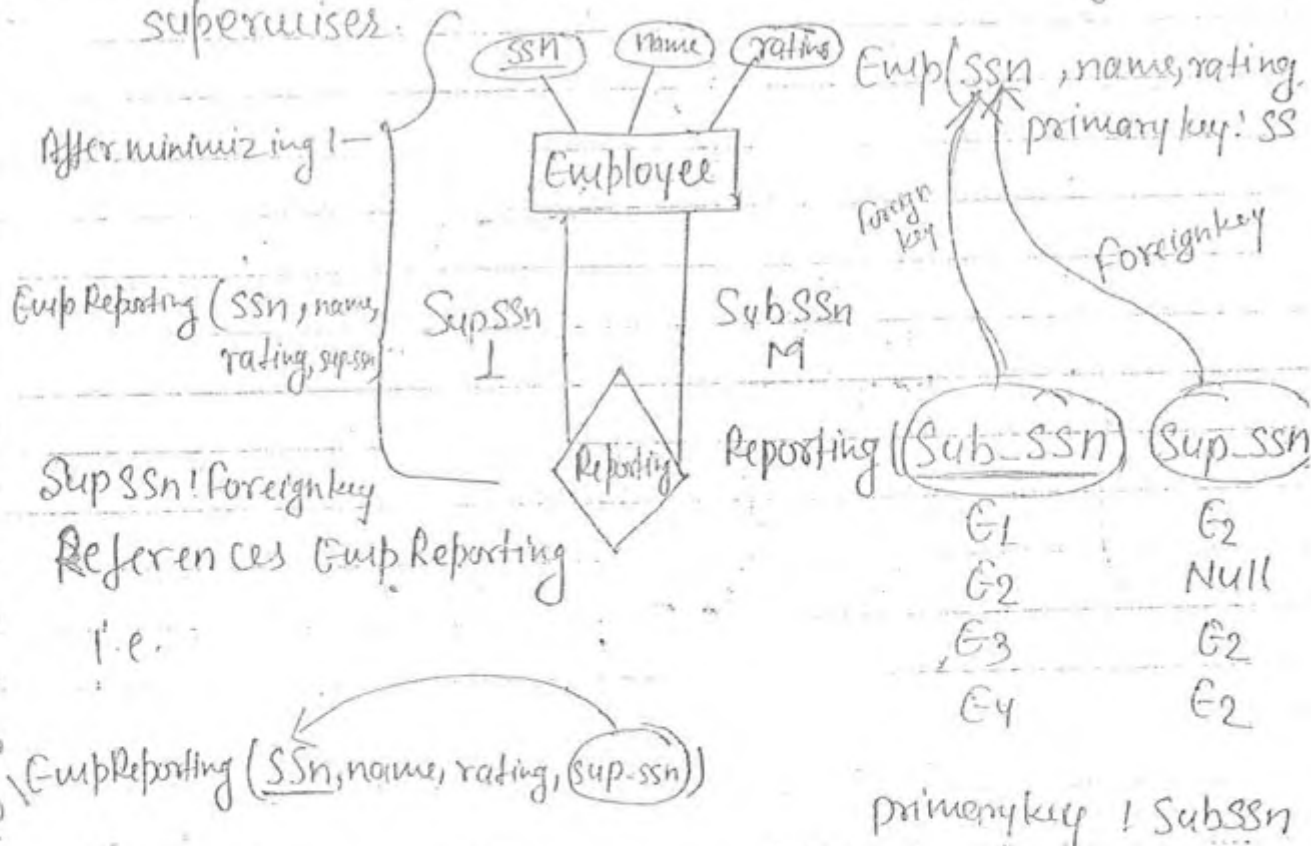
*

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.



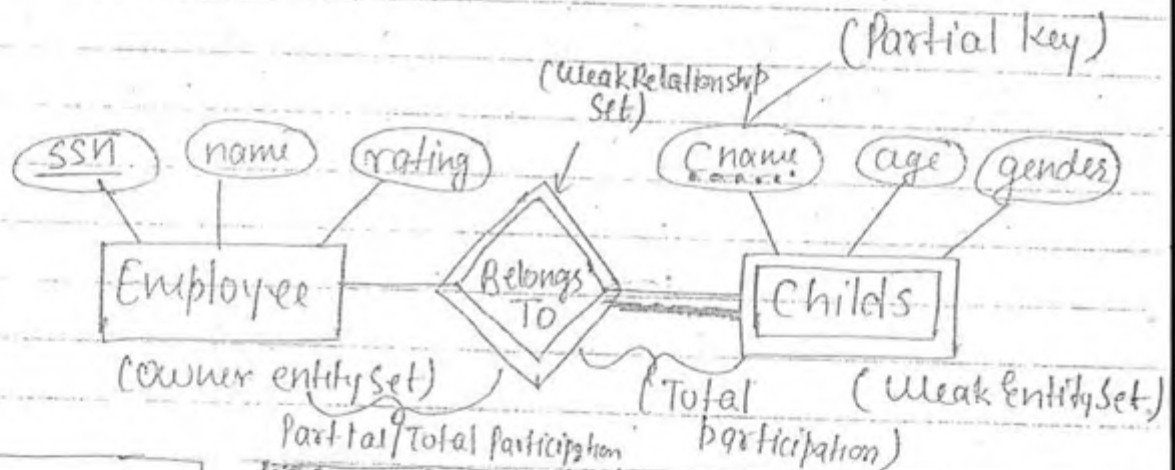
Weak Entity Set:-

- 1) Entity Set with no-proper candidate key.

It is represented by:-



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



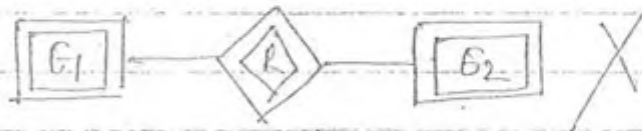
SSN
E1
E2
E3
E4

SSN	Cname
E1	A
E2	A
E3	B

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

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

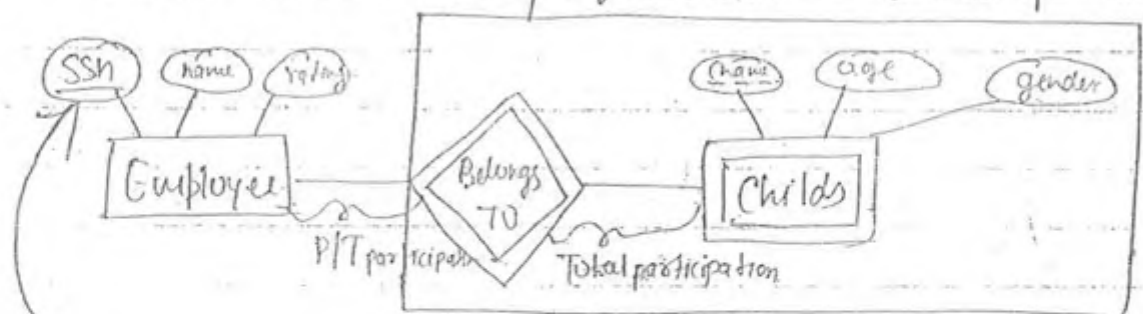
4) Two weak entity set not possible to relate with Relationship set.



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 combines with weak entity & weak relationship set.



Child belongs

SSN	Cname	age	gender
E1	A	20	M
E2	A	25	M
E3	B	20	F

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

* max no. of Tables or min. no. of tables $\Rightarrow 2$

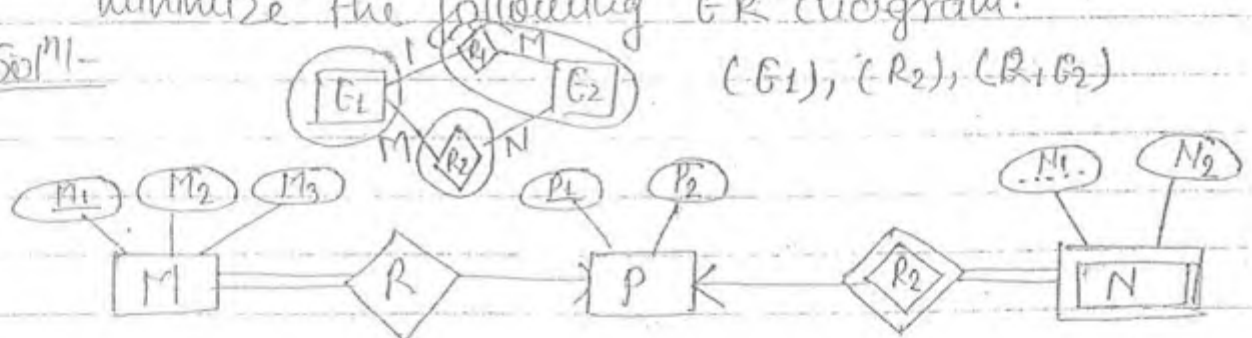
Ques

E_1 & E_2 are Entity Sets

R_1 & R_2 Relationship sets

Relating b/w E_1 & E_2 with 1:M & M:N cardinality
how many minimum no. of tables required to minimize the following ER diagram.

Soln



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

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

a) $\{M_1, M_2, M_3, P_1\}$

b) $\{M_1, P_1, N_1, N_2\}$

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

d) $\{M_1, P_1\}$

3) Deletion Anomaly:-

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

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

2) $\Rightarrow$ 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.

$Cid \rightarrow Cname$

$Sid \rightarrow Sname$

$Cid \rightarrow Fee$

Also

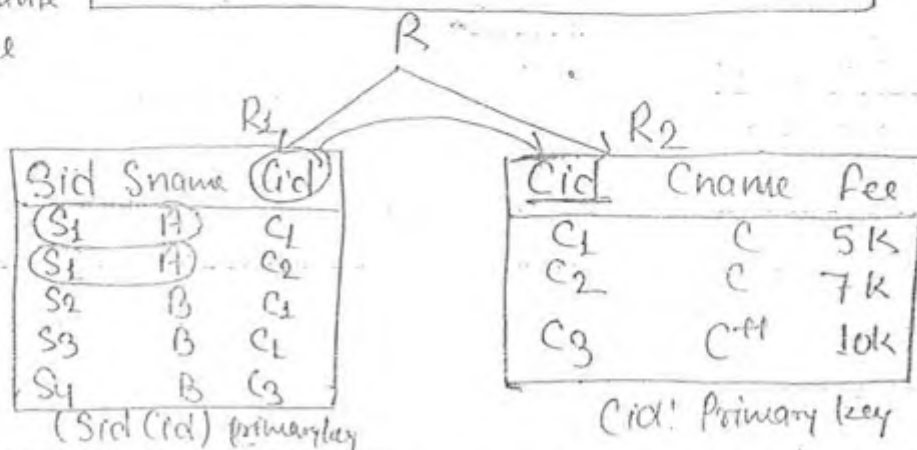
$SidCid \rightarrow Sname$

$\rightarrow Cname$

$\rightarrow Fee$

R

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



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

this then put common key or attribute in both the table. To put relationship between two decomposed 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.

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

$$R_1 \bowtie R_2 \supset R \text{ (Lossy Join)}$$

if R

S ₁	C ₁
S ₁	C ₂

$R_1 \bowtie R_2$

S ₁	C ₁
S ₁	C ₂
S ₁	C ₃

extra tuple

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

2) Dependency Preserving Decomposition:-

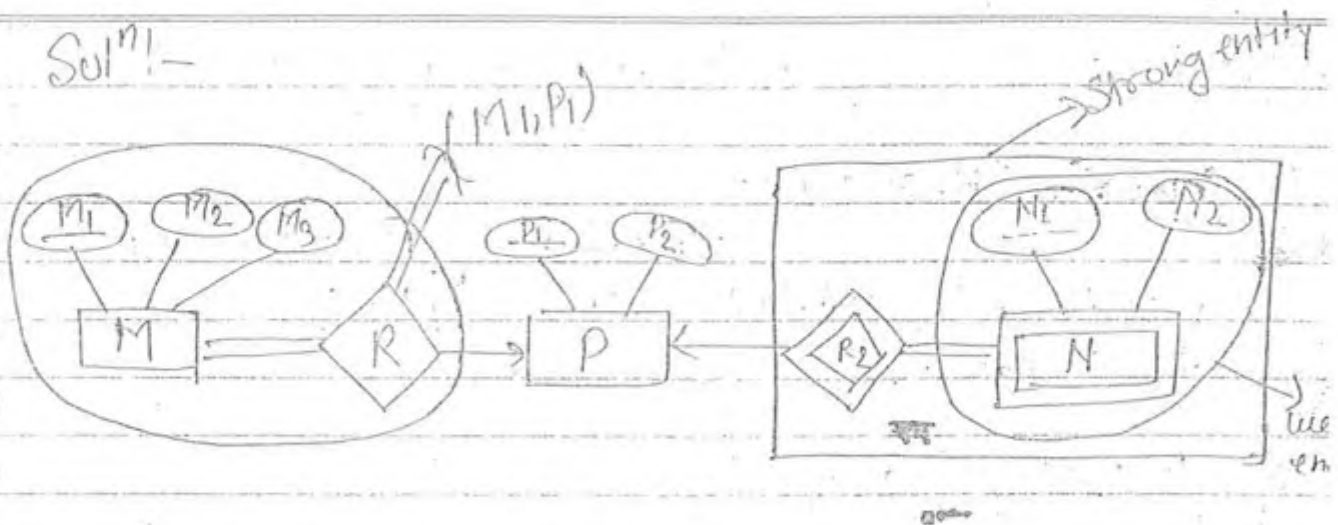
Because of decomposition, ^{relation} ~~you~~ should not lose any dependency.

Example:-

Cid → Cname
 Sid → Sname
 Cid → fee

All ~~parts~~ of the dependencies of relation also exist after decomposition into two tables.

Solⁿ:-



$MR(M_1, M_2, M_3, P_1)$ $P(P_1, P_2)$ $R_2N(P_1, N_1, N_2)$

NORMALIZATION:- (Schema Refinement)

Used to eliminate / Reduce the redundant

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

Sid	Sname	Cid	Cname	Fee
S ₁	A	C ₁	C	5K
S ₁	A	C ₂	C	7K
S ₂	B	C ₁	C	5K
S ₃	B	C ₁	C	5K
S ₄	B	C ₃	C++	10K

↳ primary key: (Sid Cid)

⇒ Problems becoz of Redundancy! - (Anomalies)

1) Update Anomaly! -

Update of single value results all duplicate copies updation which 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	C3	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 $t_1 \cdot X = t_2 \cdot X$ Then $t_1 \cdot Y = t_2 \cdot Y$

Example 1 -

X	Y
Cid	Cname
C ₁	C
C ₂	C

	X	Y
t_1	X_1	Y_1
t_2	X_2	Y_2

$$X \rightarrow Y$$

	<u>X</u>	<u>Y</u>
t_1	X_1	—
t_2	X_2	—

$$X \rightarrow Y$$

(X! is can-key)

	X	Y
G ₁	X ₁	Y ₁
G ₂	X ₁	Y ₂

$$X \rightarrow Y \text{ not}$$

Exists in $\mathbb{R}$

Dependency is of Two types:-

1) Trivial dependency:-

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

Example 1 -

Trivial Functional dependencies

- $Sid \rightarrow Sid$
- $SidName \rightarrow SidName$
- $SidName \rightarrow SidName$
- $SidName \rightarrow Sid$

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

2) Non Trivial functional dependency - ($X \rightarrow Y$ attribute Set)

$X \rightarrow Y$ is non trivial only if

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

Example -

$Cid \rightarrow Cname$
 $Sid \rightarrow Sname$
 $SidCid \rightarrow Sname$

} Non trivial functional dependency

3) Semi-non trivial functional dependency -
Combining of trivial & non trivial functional dependency is called Semi-non trivial functional dependency.

Example -

$Sid \rightarrow Sid Sname$

$\swarrow$ $Sid \rightarrow Sid$ (Trivial)
 $\searrow$ $Sid \rightarrow Sname$ (Non trivial)

Properties of functional dependency -

$\Rightarrow$ Reflexivity - If $X \supseteq Y$ Then $X \rightarrow Y$ is reflexive FD

* Trivial functional dependency are reflexive functional dependency.

⇒ Transitivity:-

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

⇒ Argumentation:-

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

Example:- if $Sid \rightarrow Sname$
then $Sid Cid \rightarrow Sname Cid$

⇒ Union of Functional Dependency:-

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

Then $X \rightarrow YZ$

but converse is not true i.e. if $\begin{matrix} A \rightarrow C \\ B \rightarrow C \end{matrix}$ then $AB \rightarrow C$ } not allowed

⇒ 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 Fun dependency:-

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

* $X \rightarrow Y$ non trivial dependency with X 's not candidate key, ^{then we can say that} attribute set (X, Y) suffering from Redundancy.

Example:- If $R(ABC)$ & if A is Candidate key & " $B \rightarrow C$ "

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

Que)

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

Identify non trivial dependency.

Solⁿ:-

Using attributes A, B, C

no. of non trivial dependency

$A \rightarrow B$

$\cancel{A \rightarrow C}$

$\cancel{A \rightarrow BC}$

$\cancel{AB \rightarrow C}$

$\cancel{AC \rightarrow B}$

$\cancel{ABC \rightarrow A}$

$B \rightarrow A$

$\cancel{B \rightarrow C}$

$\cancel{B \rightarrow AC}$

$C \rightarrow A$

$\checkmark C \rightarrow B$

$\checkmark C \rightarrow AB$

$\checkmark AC \rightarrow B$

L

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 $A \rightarrow ABCD$

$A \rightarrow A$

$A \rightarrow B$

$A \rightarrow C$

$A \rightarrow D$

$A \rightarrow A$

$A \rightarrow B$

$A \rightarrow C$

$A \rightarrow D$

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

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

$D^+ = \{D\}$

Ques

$\{AB \rightarrow CD, AF \rightarrow D, DE \rightarrow F, C \rightarrow G, E \rightarrow E, H \rightarrow A\}$

Option is false:-

a)

b)

c)

d)

$(CF)^+ = \{ACDEFH\}$ $CF^+ = \{CFHEDAD\}$

$(BH)^+ = \{ABCDH\}$ $BH^+ = \{BHACD\}$

$(AF)^+ = \{ACDEFH\}$ $AF^+ = \{AFED\}$

$(AB)^+ = \{ACDFH\}$ $AB^+ = \{ABCDH\}$

Super key:-

Let R be the Relational schema and X be the set of attributes over R .

X is said to be Super key of R only if X^+ should determine all the attributes of the Relation R .

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

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

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

AB ! Super key.

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

Not Superkey

Then AB ! minimal Superkey (Candidate key)

Ques) $R(ABCD)$
 $\{A \rightarrow B, B \rightarrow C, C \rightarrow D\}$
 Identify Candidate key

Solⁿ1- $(A)^+ = \{A B C D\}$

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

* If super key is also ^{of min} one attribute then it is also candidate key or minimal super key.

Ques) $R(ABCDE)$
 $\{AB \rightarrow C, C \rightarrow D, B \rightarrow E\}$

Solⁿ1- $(AB)^+ = \{A B C D E\}$

$\Rightarrow AB$! Super key

Check for minimal super key:-

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

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

$\Rightarrow AB$! Candidate key

Ques) $R(ABCDE)$
 $\{AB \rightarrow C, C \rightarrow D, B \rightarrow EA\}$

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

Solⁿ:- $(AB)^+ = ABCD$

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

$(ABE)^+ = ABCDE$

$(ABE)^+$ Candidate Key

Ques) $R(AB CDEF)$
 $\{A \rightarrow BCDEF, BC \rightarrow ADEF, B \rightarrow F, D \rightarrow E\}$

Solⁿ:-

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

$B^+ = BF$ 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) $R(ABCD)$
 $\{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A\}$

Solⁿ1-
 $A^+ = ABCD$
 $B^+ = ABCD$
 $C^+ = ABCD$
 $D^+ = ABCD$

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

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

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

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

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

$(AC)^+ = \{ACDEFB\}$ not super key

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

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

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

$(AE)^+ = \{AEFBCD\}$

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

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

$D^+ = \{DEFBA\}$

$(AC)^+ = \{ABCDEF\}$ $C^+ = \{CDEFB\}$ x

Ques) $R(ABCDEFGHIJ)$
 $\{ AB \rightarrow C$
 $A \rightarrow DE$
 $B \rightarrow F$
 $F \rightarrow GH$
 $D \rightarrow IJ \}$

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

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

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

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

Ques) $R(ABDLPT)$
 $\{ B \rightarrow PT, T \rightarrow L, A \rightarrow D \}$
 $B^+ = (BPTL)$
 $(AB)^+ = (BPTLAD)$

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

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

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

Ques) $R(ABCDEFGHIH)$

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

$$(ABFH)^+ = \{ABCD E F H\}$$

$$(BCFH)^+ = \{ABCD E F H\}$$

$$(ACFH)^+ = \{ABCD E F H\}$$

$$\text{Candidate Keys} = \{ABFH, BCFH, ACFH\}$$

Ques) $R(ABCDE)$

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

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

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

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

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

$$(AC)^+ = \underline{\hspace{2cm}}$$

$$\text{Candidate Keys} = \{A, E, CD, BC\}$$

20/08/11

Ques 1) $R(ABCDEF)$
 $\{AB \rightarrow C, C \rightarrow DE, E \rightarrow F, F \rightarrow A\}$

Ques 2) $R(ABCDEF)$
 $\{AB \rightarrow C, C \rightarrow D, D \rightarrow E, E \rightarrow A, D \rightarrow B\}$

Soln 1)

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

Candidate Key 1- $\{AB, FB, EB, CB\}$

Soln 2)

$ABF^+ = \{ABCDEF\}$
 $\cancel{X} ADF^+ = \{ABCDEF\}$
 $\cancel{X} EDF^+ = \{ABCDEF\}$
 $CF^+ = \{ABCDEF\}$
 $(ABD)^+ = ABCDEF$
 $(EBF)^+ = ABCDEF$
 $E^+ = EA$
 $B^+ = B$
 $\cancel{(DFA)^+ = ABCDEF}$
 $(BF)^+ = BFE$
 $(DF)^+ = DFEABC$
 $(CF)^+ = ABCDEF$

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

Ques 3)

$R(ABCDE)$

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

$AB^+ = \{ABCDE\}$

$AB^+ = \{ABCDE\}$

$A^+ = \{A\}$

$\cancel{A}D^+ = \{ABCDE\}$

$B^+ = \{B\}$

$D^+ = \{DEBAC\}$

$AB^+ = \{ABCDE\}$

$E^+ = \{EA\}$

$C^+ = \{CDEAB\}$

$\cancel{D}B^+ = \{ABCDE\}$

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

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

* If $R(ABCD)$
 $\{AB \rightarrow C, DE \rightarrow F, C \rightarrow D, E \rightarrow F, DE \rightarrow C\}$
 $\Downarrow$
 $C \rightarrow D, C \rightarrow E$

then remaining Relation

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

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

Properties of Functional dependency Set:-

1) Membership test:-

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

Q

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 membership set of the set.

$\Rightarrow$ Let F be the functional dependency set and $X \rightarrow Y$ and $P.D.$
 $X \rightarrow Y$ logically ($\models$) in P only, if closure set of X should consist 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 1 - $F = \{A \rightarrow B, B \rightarrow C\}$

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

F^+ contains

$$\textcircled{1} \leftarrow \begin{cases} \phi^+ = \phi \\ A^+ = ABC \end{cases}$$

$$\textcircled{1} \leftarrow \begin{cases} B^+ = BC \\ C^+ = C \end{cases}$$

$$\textcircled{2} \leftarrow \begin{cases} (AB)^+ = ABC \\ (BC)^+ = BC \\ (AC)^+ = ABC \end{cases}$$

$$\textcircled{3} \leftarrow (ABC)^+ = ABC$$

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

43

Here F^+ is 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)

$$A^+ = AB$$

$$A \rightarrow \Phi \quad A \rightarrow A \quad A \rightarrow B \quad \text{— (4)}$$

$$A \rightarrow AB$$

$$B^+ = AB$$

$$B \rightarrow \Phi \quad B \rightarrow A \quad B \rightarrow B \quad \text{— (4)}$$

$$B \rightarrow AB$$

$$AB^+ = AB$$

$$AB \rightarrow \Phi \quad AB \rightarrow AB \quad \text{— (4)}$$

$$AB \rightarrow A \quad AB \rightarrow B$$

13

3) Equality b/w FD Sets -

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

b) F & h said to be Equal only if

[1] F covers h -

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

[2] h covers F -

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



F covers G

G covers F

T

T

$\Rightarrow F = G$

T

F

$\Rightarrow F \supset G$

F

T

$\Rightarrow G \supset F$

F

F

$\Rightarrow F \neq G$ (No

comparison value)

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

$E \rightarrow A$
 $F \rightarrow H$
 $A \rightarrow C$
 $A \rightarrow D$

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

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

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

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

F COVER G is true

G COVER F is false

$\Rightarrow F \supset G$

Properties of Decomposition:-

1) Lossless Join Decomposition:-

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

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 $R_1 \bowtie R_2 \bowtie \dots \bowtie R_n \supset R$

Then Lossy join decomposition

Example:-

$R(ABC)$ decomposed into $R_1(AB)$ $R_2(BC)$

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

A	B
1	1
2	1
3	2

B	C
1	2
1	1
2	1

$$R_1 = \pi_{AB}(R)$$

$$R_2 = \pi_{BC}(R)$$

π : Projection

To Select attribute of Relation

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

σ : Selection operator

Select the tuples those are satisfied predicate condition

SECOND NORMAL FORM:-

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

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

Partial Dependencies:-

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



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

OR

X : any candidate
 key of R

Y : Proper Subset
 of any candidate
 key

A : Non-Prime
 attribute

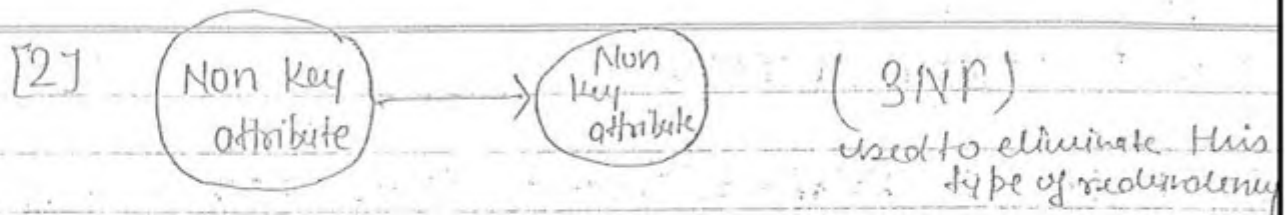
~~if~~ $Y \rightarrow A$ Non-trivial FD is partial dependency: only if

(1) Y is proper Subset of Candidate key.
 and

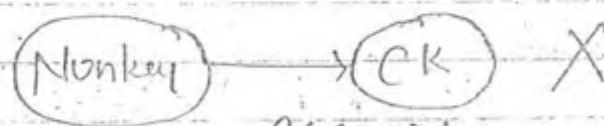
(2) A is Non-Key or Non Prime attribute.

~~if~~ $Y \rightarrow A$ Non-trivial FD is partial dependency: only if
 need to check the relation is in 2NF or not. if yes, use the
 above condition for checking that above relation is in
 2NF or not.

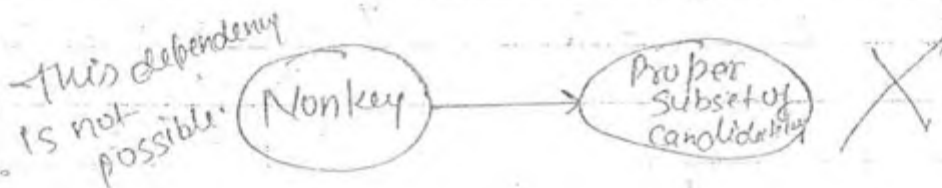
* If right side of dependency candidate key is not possible.
 i.e. if candidate key is present in right side of dependency then reduction is not possible.



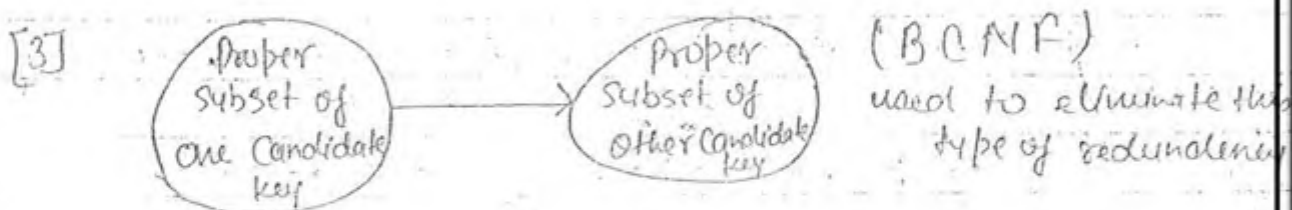
Example - $R(AB CDE) \{ AB \rightarrow C, C \rightarrow D, B \rightarrow E \}$
 [2] case [2] case



Example - $R(ABCD) \{ A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A \}$
 candidate key $\{ A, B, C, D \}$



Example - $R(AB CDE) \{ AB \rightarrow C, C \rightarrow D, B \rightarrow E, C \rightarrow A \}$
 C.K $\{ AB, CB \}$



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

Relation Consists Redundancy becoz of single valued dependency:-

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

Example:-

X	Y
x_1	y_1
x_2	y_1
x_1	y_1

Not Super key

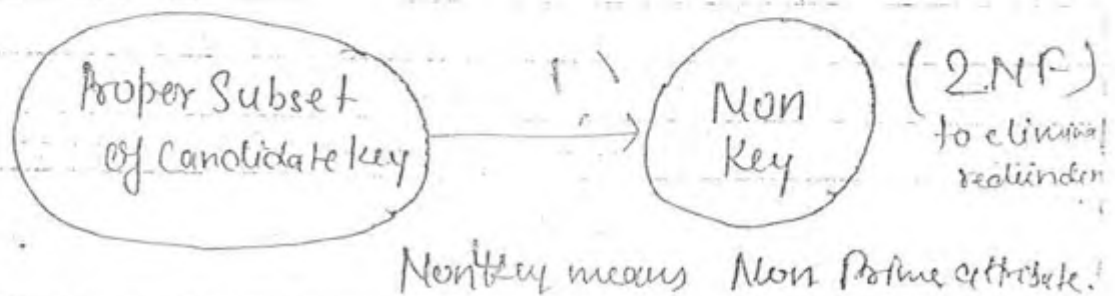
OR in the other way as contradiction of the above statement:-

(if any Non-trivial $X \rightarrow Y$ exists in R then X should be Super key.

Then R is free from Redundancy becoz of single value dependencies.)

Non-Trivial FD which results Redundancy:-

[1]



By Converting the above relation in 1NF: We get

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

Redundancy

(Sid $\rightarrow$ Sname

Sid Cname $\rightarrow$ 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

Solⁿ:-

R(A CD, B) ! (1NF Relation)

primary key = ACD

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

Dependency:-

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

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

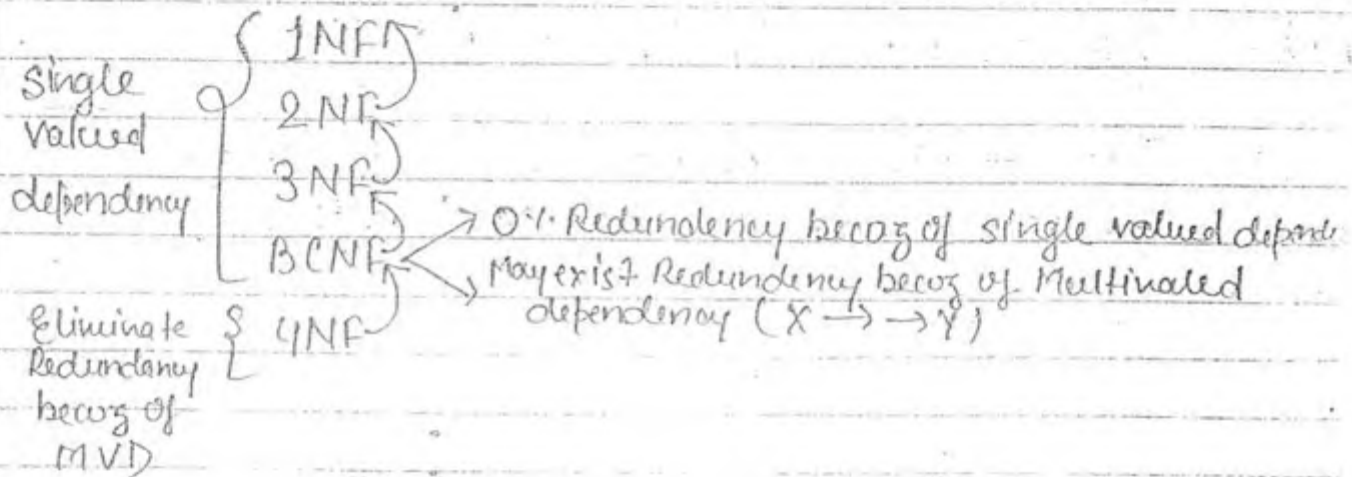
21/08/11

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 exist over R.

Example:-

Sid	Sname	Cname
S1	A	DB/D11
S2	B	DM/BM
S3	A	DB

Multivalued attribute

(NOT in 1NF)

Primary key : Sid

Check for correct decomposition

Ques) $R(ABCD EFGH IJ)$

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

$D_1 = \{ \begin{matrix} ABC, ADE, B \cdot F, FGH, DIJ \\ AB \rightarrow C & A \rightarrow DE & B \rightarrow F & F \rightarrow GH & D \rightarrow IJ \end{matrix} \}$ These two

$D_2 = \{ \begin{matrix} ABCDE, BFGH, DIJ \\ AB \rightarrow C & A \rightarrow DE & B \rightarrow F & F \rightarrow GH & D \rightarrow IJ \end{matrix} \}$ dependency preserving

$D_3 = \{ ABCD, DE, BF, FGH, DIJ \}$

Ans1- Decomposition is proper only if it satisfy
lossless join + Dependency preserving.

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

5) Intersection ($\cap$) - ✓
 $R1 \cap R2$

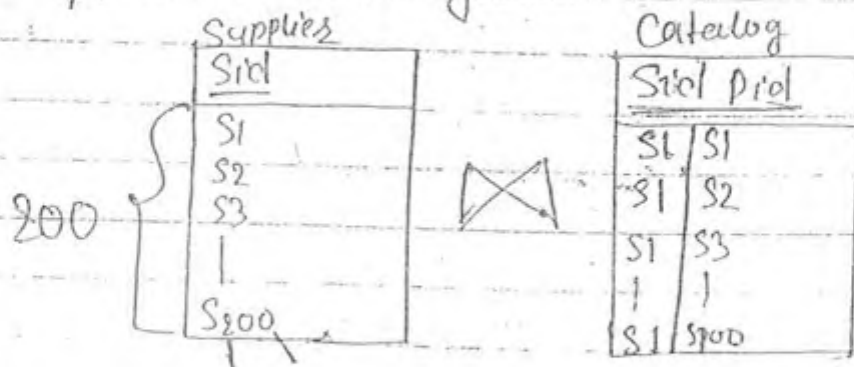
6) Set Difference ($-$) - X

7) / (Division) - X

$$A(X,Y) / B(Y) \neq B(Y) / A(X,Y)$$

8) Natural Join ($\bowtie$) = ✓

Query 1 - Assume Supplier relation consist 200 tuples
 & catalog relation consist 100 tuples.
 How many maximum no. of tuples resulted by
 suppliers $\bowtie$ catalog.



Max no. of tuples in $\text{Sup} \bowtie \text{Catalog}$ = 100
 min no. of tuples in $\text{Sup} \bowtie \text{Catalog}$ = 100 (being cate
 & being sid is foreign key of suppliers Sid so, sid refer to
 minimum 100 tuples is necessary. Suppliers Tab

[04]

$$\pi_{name} \left(\pi_{sid} \left(\pi_{pid} \left(\sigma_{col=Reel} (Parts) \right) \bowtie \pi_{sid, cid} (Catalog) \right) \bowtie \pi_{sid, name} (Supplies) \right)$$

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

Check for Commutative

1) π (Projection) = $\times$

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

2) σ (Selection) = $\checkmark$

$$\sigma_{c2} (\sigma_{c1} (R)) = \sigma_{c1} (\sigma_{c2} (R)) = \sigma_{c1 \wedge c2} (R) = (\sigma_{c1} (R)) \cap (\sigma_{c2} (R))$$

3) Cross Product = $\checkmark$

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

Que) $R(ABCDEN)$

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

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

ABC	ACDE	ADh

$$A^+ = A$$

$$B^+ = BD$$

$$C^+ = C$$

$$AB^+ = ABC$$

$$BC^+ = ABCDEh$$

$$AC^+ = ACBDEh$$

$$A^+ = A$$

$$C^+ = C$$

$$D^+ = D$$

$$AC^+ = ABCDEh$$

$$AD^+ = ADEh$$

$$Ah^+ = Ah$$

$$CD^+ = CD$$

$$Ch^+ = Ch$$

$$Dh^+ = Dh$$

$$ACD^+ = ACDBDEh$$

$$ACH^+ = ACBDEh$$

$$CDh^+ = ACDBDEh$$

$$A^+ = A$$

$$D^+ = D$$

$$Ah^+ = Ah$$

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

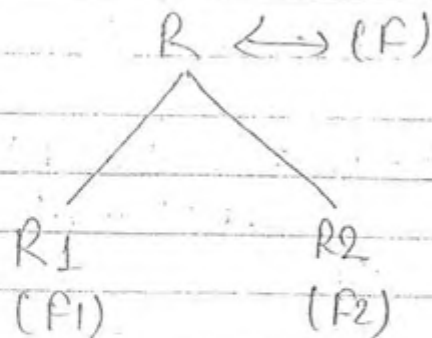
* All the original fun dependency are preserved in it so it is Dependency preserving decomposition

Ques) Check whether decomposition Preserving or Not!
 $R(ABCD E F)$
 $F = \{ AB \rightarrow CD, C \rightarrow D, D \rightarrow E, E \rightarrow F \}$
 $D = \{ AB, CDE, EF \}$

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

Not preserving dependency

$A^+ = A$
 $B^+ = B$
 ~~$C^+ = CDE$~~
 $C^+ = CDEF$
 $D^+ = DEF$
 $E^+ = EF$
 $F^+ = FE$



which of them is Always False

- a) $F_1 \cup F_2 \supset F$
- b) $F_1 \cup F_2 = F$ (Dependency preserving)
- c) $F_1 \cup F_2 \subset F$ (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
 $R = (ABCD)$

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

$$D_2 = \{ AB, BC, CD \}$$

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

Que) $R(ABCD)$

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

$D = \{AB, BCD\}$

Soln:- Identify FD's of Subrelations:-

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

$A^+ = ABCD$

$B^+ = BCD$

$B^+ = BCD$

$C^+ = CD$

$D^+ = D$

$BC^+ = BCD$

$CD^+ = CD$

$BD^+ = BCD$

Apply membership test

$A \rightarrow B$

$B \rightarrow C$

$C \rightarrow D$

$F_1 \cup F_2$ Covers F Then preserving dependency

a) $R_1(AB)$ $R_2(BC)$ $R_3(ABDE)$ $R_4(EH)$

$AB^+ = ABCDEH$

$R_{13}(ABDE)$

$R_2(BC)$

$R_4(EH)$

$R_2(BC)$

$R_{134}(ABDEH)$ $E^+ = EH$

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

b) $R_1(ABC)$ $R_2(A'DE)$ $R_3(ADH)$

$AC^+ = ACBDEH$

$R_{12}(AB(CDE))$

$R_3(ADH)$

$AD^+ = ADEH$
is superkey
for R_3 relation

$R_{12}(AB(CDEH))$

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

Decomposition is lossless only if N Relations Become Single Relation.

Ques 1) $R(ABCDEF)$

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

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

2) $R(ABCDE)$

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

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

3) $R(ABCD)$

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

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

4) $R(ABCDEH)$

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

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

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

Soln 1) Lossy

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

2) $AC^+ = \{AB, CDE\}$ superkey lossless

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

4)

~~Lossy~~

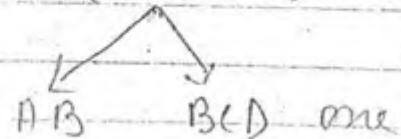
* Algorithm for checking decomposition is lossless or lossy

⇒ 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

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

(because of decomposition R should not lose any attribute)

i.e. if $R(ABCDE)$

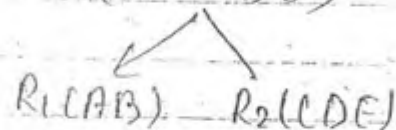


attribute is lost after decomposition.

and [II] let R_i & R_j are two relations can be merge into single relation $R_{ij} = R_i \cup R_j$ only if following conditions are true

$$I. R_i \cap R_j \neq \emptyset$$

i.e. $R(ABCDE)$



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

$$R_1 \bowtie R_2 = R$$

[lossless Join]

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

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

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

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

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

$\Rightarrow$ decomposition is loss-less decomposition.

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

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

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

^{Note}
Ques 1- If $R(ABCD)$ decomposed into

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

given decomposition is lossy.

$$S = R_1 \bowtie R_2$$

which one is true

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

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

$$R_1$$

A	B
1	1
2	1
3	2

$$R_2$$

A	C
1	2
2	1
3	1

$$\pi \left(\begin{matrix} R_1 \times R_2 \\ R_1 \cdot A = R_2 \cdot A \\ R_1 \bowtie R_2 \end{matrix} \right) =$$

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

Arity! - Means addition of attribute of Relations
 Cardinality! - Means multiplication of rows of relation

~~R1 x R2 =~~

(R1 x 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

$R_1 \bowtie R_2 =$

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

$\Rightarrow$

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

Extra Tuple after decomposition

$r_1 \bowtie r_2 \supset \{ \}$
 (lossy Join)

* If $R_1(ABCE)$ $R_2(BCDE)$
 $\pi_{ABCE} (R_1 \bowtie R_2) = \pi_{ABCE} \left(\sigma_{\begin{matrix} R_1.B = R_2.B \wedge \\ R_1.C = R_2.C \end{matrix}} (R_1 \times R_2) \right)$

Example 1-

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

(Sid, Cname) :- Primary key

Sid $\rightarrow$ Sname (Partial Dependency) } Not in 2NF
 Sid, Cname $\rightarrow$ Sname

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

Sid $\rightarrow$ Sname

Candidate key :- (Sid, Cname)

Above Decomposition is

Lossless Join

+

Dependency Preserving

+

2NF (NO partial

Dependency)

Example 1-

$R(AB CDEF)$

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

Check the above relation is in 2NF or not.

Solⁿ -

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

(Partial Dependency)

Candidate Key :- AB

Convert it to 2NF -

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

R_2

B	E	F
---	---	---

$B \rightarrow EF$

B! Candidate key

Put remaining attribute in other relation

R_1

A	C	D
---	---	---

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

R_1

A	B	C	D
---	---	---	---

$AB \rightarrow C$

$C \rightarrow D$

AB! Candidate key

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

R_2

B	E	F
---	---	---

$B \rightarrow EF$

B! Candidate key

Example 1-

$R(AB CDEF)$

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

AB ! Candidate key.

If we put all PD in one relation :

$\boxed{A \ B \ E \ F}$

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

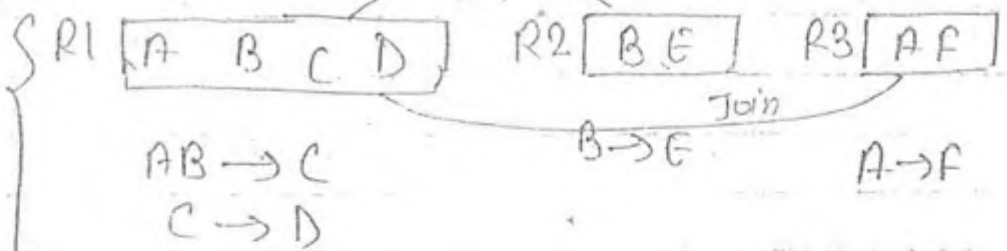
(AB) ! Candidate key.

So put all PD in one relation

Dependency preserving

+ lossless join

+ NO PD (2NF)



* $\Rightarrow$ 2NF Not free from Redundancy bcoz of

1) Nonkey $\rightarrow$ Nonkey not violate the condition of 2NF

2) Proper Subset of Candidate key $\rightarrow$ Proper Subset of attribute 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 1-

It internally
satisfy that
the relation
is in 2NF.

There is no
need to
check for

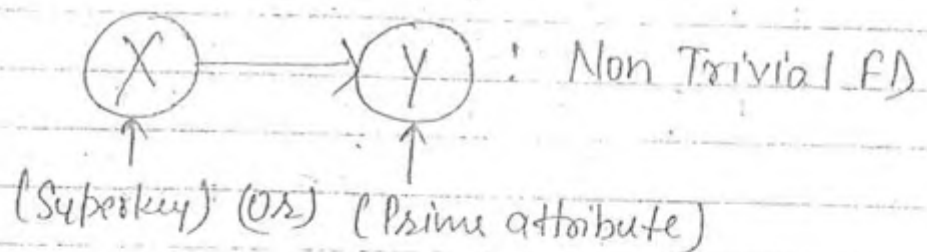
2NF if we

apply these
condition for

checking that the
relation is in
3NF or not.

a) X is Super key
(or)

b) Y is prime attribute.



$\Rightarrow$ 3NF def is also eliminating partial dependence
or
we can say that it satisfy the 2NF.

$\Rightarrow$ 3NF eliminates Nonkey $\rightarrow$ Nonkey
dependencies.

3NF - Relation R is in 3NF only if

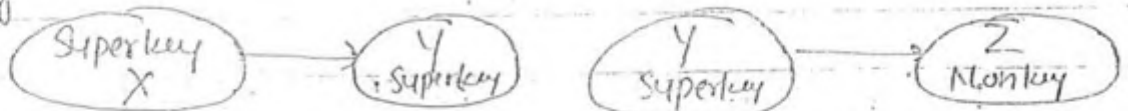
- 1) R should be in 2NF (no partial dependence)
- and 2) Nonkey should not transitively determine by candidate key or Super key.
(No transitive dependence)

Failed i.e.
by 3NF
definition



⇒ Nonkey Z Transitive determined by Superkey X.

3NF
definition
Satisfied



here Z is directly determined by Superkey

Check it is 3NF or not.

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

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

Solⁿ1- First check R is in 2NF or not

$\{ AB \rightarrow CD, D \rightarrow A, C \rightarrow E \}$ } No partial dependence
Candidate key $\{ AB, BD \}$ } So, it is in 2NF

Check for 3NF

$\{ AB \rightarrow CD, D \rightarrow A, C \rightarrow E \}$ } So not in 3NF because
Condition 1 ✓ Condition 2 ✓ X } E is transitively determined by
Superkey i.e. AB

Decompose into 3NF:-

R_1

A	B	C	D
---	---	---	---

R_2

C	E
---	---

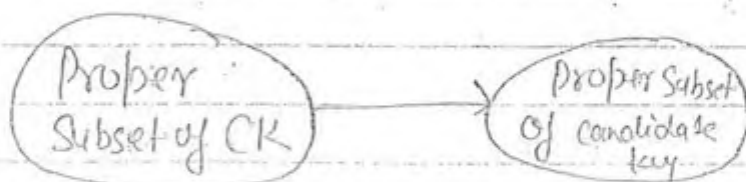
Yet redundancy $AB \rightarrow CD$
ent $\leftarrow D \rightarrow A$

Candidate key $\{ AB, BD \}$

$C \rightarrow E$
 $\{ CE \}$ - Candidate key

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

* 3NF is also not free from Redundancy because



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:-

$R_1(ABCD)$

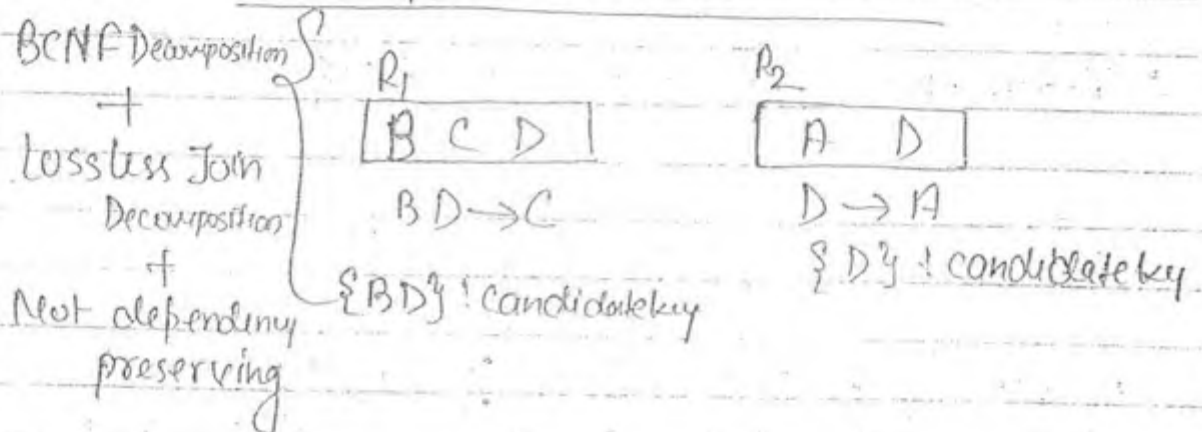
$\{AB \rightarrow CD, D \rightarrow A\}$

Solⁿ 1- $\{AB \rightarrow CD, D \rightarrow A\}$

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

The Relation R is not in BCNF.

Decompose into BCNF:-



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

* If there is any dependency exist like



then there must be dependency preservation is violates.

Example 1-

$R(AB CDE)$

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

Candidate key (AB)

Partial Dependend

Decomposit
in 2NF

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

$B \rightarrow E$
 $\{B\}$

Decompositio
in 3NF

$AB \rightarrow C$

$C \rightarrow D$

$AB \rightarrow C$

$C \rightarrow D$

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 1-

$R(AB CDE)$

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

$R_1(ABC)$

$R_2(CD)$

$R_3(BE)$

$AB \rightarrow C$

$C \rightarrow D$

$B \rightarrow E$

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	2NF	3NF	BCNF
1) 0/1 Redundancy	High	< 1NF	< 2NF	0/1 Redundancy beoz of single valued function dependencies (Redundancy possible beoz of MVD)
2) Lossless Join decomposition	Always	Always	Always	Always
3) Dependency preserving decomposition	Always	Always	Always	} Some Relation may not possible to decompose into BCNF by pres- erving dependencies

* 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-
ency. Then 'R' is in BCNF.

Proof $X \rightarrow Y$ non trivial is in BCNF only if X is super
key.
i.e. If non trivial $\rightarrow$ BCNF
if not non trivial $\rightarrow$ BCNF

T	T	T
F	T	T

⇒ Binary Relation

Relation with Two attributes, always in BCNF

$R(AB)$

a) $A \rightarrow B$

Candidate key $\{A\}$

b) $B \rightarrow A$

$\{B\}$

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

$\{A, B\}$

d) No Non Trivial FD. AB

(Ques) Assume no multivalued attribute

$R(AB C D E F)$

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

Solⁿ:-

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

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

$\boxed{AB C D E}$

✓ $A \rightarrow BCDE$

✓ $BC \rightarrow ADE$

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

$\{A, BC\}$ candidate key

$\boxed{AB C D}$

$A \rightarrow BCD$

$BC \rightarrow AD$

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

$\boxed{D E}$

$D \rightarrow E$

$\{D\}$! Candidate key

$\boxed{B F}$

✓ $B \rightarrow F$

$\{B\}$! Candidate key

$\boxed{B F}$

$B \rightarrow F$

$\{B\}$! Candidate key

2NF

+

Not in

3NF

3NF

+

BCNF

$\overline{B C F} \quad ADE$
 $A \rightarrow DE$

(Ques) $R(AB CDEF GH IJ)$

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

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

Candidate
 Key $\{AB\}$

$\checkmark$

$\checkmark$

$\times$

$\times$

$\times$

~~$\{AB\}$~~

2NF Decomposition

$\boxed{A \ B \ C}$

$AB \rightarrow C$

$\boxed{B \ F \ G \ H}$

$B \rightarrow F$

$\boxed{F \rightarrow GH}$

$\{B\}$

$\boxed{A \ D \ E \ I \ J}$

$A \rightarrow DE$

$\boxed{D \rightarrow IJ}$

$\{A\}$

Candidate key $\{AB\}$

3NF Decomposition

$\boxed{A \ B \ C}$

$AB \rightarrow C$

$\boxed{B \ F}$

$B \rightarrow F$

$\boxed{F \ G \ H}$

$F \rightarrow GH$

$\boxed{A \ D \ E}$

$A \rightarrow DE$

$\boxed{D \ I \ J}$

$D \rightarrow IJ$

It is also in BCNF

Ques) $R(AB C D E F)$

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

R_1
 $\boxed{A B C D E}$

$A \rightarrow BCDEF$

$BC \rightarrow ADE$

R_2
 $\boxed{B F}$

$B \rightarrow F$

R_3
 $\boxed{D F}$
 $D \rightarrow F$

Ques) $R(AB C D E F G H)$

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

Soln-

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

Candidatekey $(ABFH) BCFH, ACFH$

2NF Decomposition

$R_1 \boxed{A B C E F G H}$

$AB \rightarrow C$

$AC \rightarrow B$

$BC \rightarrow A$

$PD \quad \boxed{E \rightarrow G}$

$R_2 \boxed{B D}$

$B \rightarrow D$

$R_3 \boxed{A D E}$

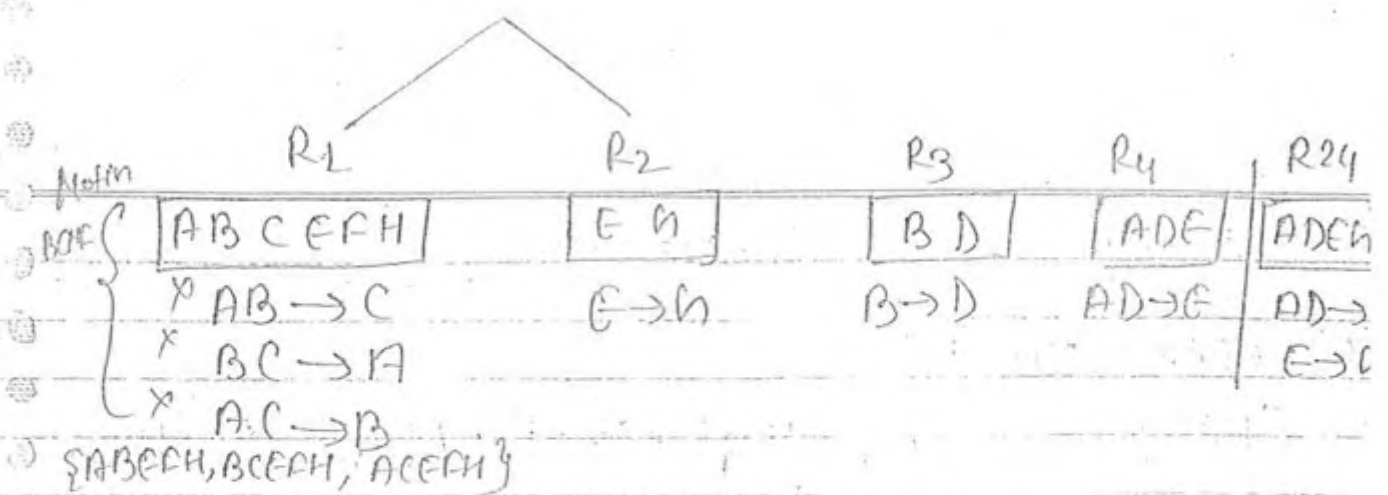
$AD \rightarrow E$

Candidate key $\{ABFH, BCFH, ACFH\}$

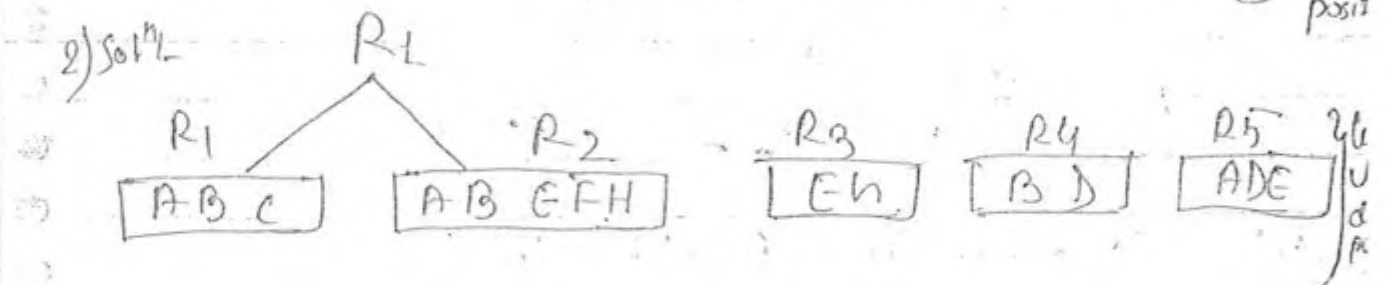
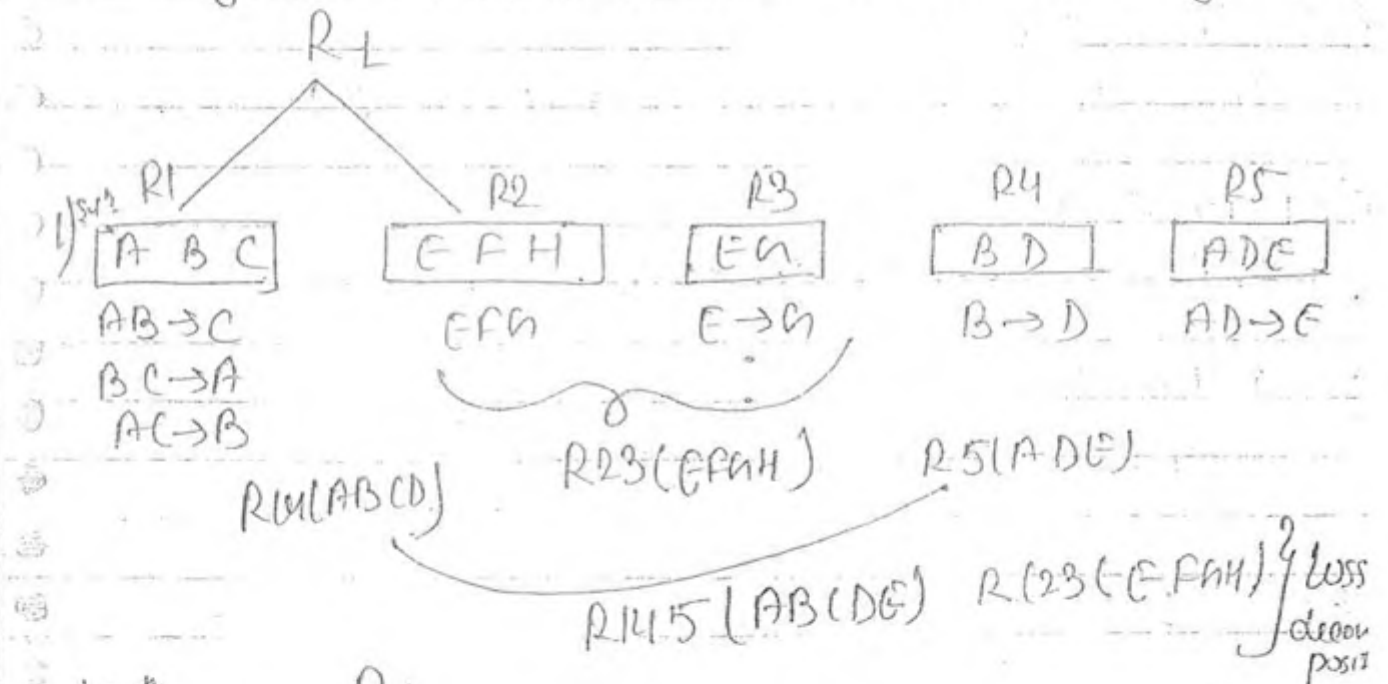
$\{B\}$

$\{AD\}$

R_1 : Again it's not in 2NF



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



22/08/2017

Ques) Identify highest Normal Form

[1] $R(AB C D E F G H)$

$\{ABC \rightarrow DE, E \rightarrow FG, H \rightarrow G, G \rightarrow H, ABED \rightarrow EFG\}$

[2] $R(AB C D E)$

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

[3] $R(AB C D E F G H)$

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

[4] $R(AB C D E F)$

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

[5] $R(AB C D E)$

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

[1] Solⁿ: Candidate key $[AB C]$

2NF $\checkmark$

3NF $\times$

BCNF $\times$

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

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

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

3NF: ✓

BCNF: X

Q Solⁿ - Candidate key: $\{AH, HA, FH, EH, DH, CH\}$

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

Not in 2NF

So, it is in 1NF

Q Solⁿ - Candidate key: $\{AB, FB, EB, DB, CB\}$

Not in BCNF. But it is in 3NF. Because right side of $AB \rightarrow C, C \rightarrow D, D \rightarrow E, E \rightarrow F, F \rightarrow A$ side all prime attribute. So, it is in 3NF

Q Solⁿ - Candidate key: $\{AB, FB, EB, CB\}$

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

Not in 2NF

So, it is in 1NF.

Minimal Cover or Canonical Cover

It is used to eliminate redundant functional dependency.

Example

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

$A \rightarrow C$ is eliminated

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

$AB \rightarrow C$ is eliminated because

Argumentation Property $B \rightarrow C$

$$AB \rightarrow AC$$

$$\therefore AB \rightarrow A$$

$$AB \rightarrow C$$

$$\text{Then } G \subseteq \{ A \rightarrow B, B \rightarrow C \}$$

$$\Rightarrow (F \equiv G)$$

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

Extraneous Attribute-

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

B is extraneous Attribute-

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

Forward h1 - True

Reverse F :- $G = \{ A \rightarrow C \} \models AB \rightarrow C$ True
 $\models A \rightarrow C$ True

By oppⁿ method
 ship. test

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

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

Proof First check A is extraneous

$$F = \{ \overset{x}{A}B \rightarrow C, A \rightarrow B \}$$

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

Forward h1 :- $F = \{ \overset{x}{A}B \rightarrow C, A \rightarrow B \} \models B \rightarrow C$ X
 $\models A \rightarrow B$ ✓

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

$$F \neq G$$

Take B is extraneous attribute

$$F = \{ AB \rightarrow C, A \rightarrow B \} \models A \rightarrow C \checkmark$$

$$\models A \rightarrow B \checkmark$$

$$G = \{ A \rightarrow C, A \rightarrow B \} \models AB \rightarrow C \checkmark$$

$$\models A \rightarrow B \checkmark$$

$$F \text{ covers } G \checkmark$$

$$G \text{ covers } F \checkmark$$

$$(F \equiv G)$$

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

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

Proof:-

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

Combination of transitivity & augmentation

$$\text{if } AB \rightarrow C$$

$$AB \rightarrow BC \text{ if } BC \rightarrow D$$

$$\text{then } AB \rightarrow D$$

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

(Que) Minimal cover of

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

$$\text{Sol}^n - F = \{ A \rightarrow BC^*, B \rightarrow C, C \rightarrow B \} \quad F = \{ A \rightarrow BC, B \rightarrow C, C \rightarrow B \}$$

$$\text{minimal cover}_1 = \{ A \rightarrow B, B \rightarrow C, C \rightarrow B \} \quad \text{minimal cover}_2 = \{ A \rightarrow C, B \rightarrow C, C \rightarrow B \}$$

$$F_1 = F_2 = F$$

$\Rightarrow$ Minimal cover is not unique. But all minimal covers logically equivalent.

(Que) $F = \{ A \rightarrow B, (D \rightarrow E, E \rightarrow D, A \rightarrow B, AC \rightarrow D)^* \}$

$$\text{Sol}^n - \{ AC \rightarrow E, E \rightarrow D, A \rightarrow B, AC \rightarrow D \}$$

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

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

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

Ques) Identify minimal cover:-

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$(A \rightarrow B, A \rightarrow C, B \rightarrow A, C \rightarrow$$

$$\{A \rightarrow B, A \rightarrow C, B \rightarrow A, C \rightarrow A\} \quad (A \rightarrow BC, B \rightarrow A, C \rightarrow$$

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

Identify minimal cover:-

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

$$\text{Soln: } \{A \rightarrow BC, \cancel{D \rightarrow E}, E \rightarrow C, D \rightarrow AEH, \cancel{ABH \rightarrow BD}, DH \rightarrow BC\}$$

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

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

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

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

To check minimal cover is correct or not:-

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

[2] Minimal cover logically equivalent to original cover. i.e. we obtained all the FD from minimal cover FD, ~~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 1- $R(\underline{A}, B, C)$ C: Multivalued attribute
A: is primary key

R Not in 1NF

Convert into 1NF = $R(\underline{A}, \underline{C}, B)$
primary key $\{A, C\}$

⇒ $R(\underline{A}, B, C, D)$ C, D: Multivalued attribute
A: is primary key

R Not in 1NF

Convert into 1NF = $R(\underline{A}, \underline{C}, \underline{D}, B)$

Primary key $\{A, C, D\}$

⇒ $R(\underline{A}, B, C, D, E, F)$
 $\{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A\}$
and E, F are Multivalued attribute
Then Candkey :- A, B, C, D

Not
is
1NF

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

1NF relation $\{ R(\underline{A}, B, C, D, E, F) \}$ // No multivalued attributes.
 $\{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A\}$
Cand key :- AEF, BEF, 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 $\{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A\}$

If candidate key is $\{A, B, C, D\}$

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 there. So, it is not even in 1NF. Candidate key not determine E & F this implies E & F are multivalued attribute.

$\Rightarrow$ If Relation consist two or more multivalued attribute while conversion of Relation into 1NF then MVD is possible.

Example - $R(\underline{A}, B, C, D)$

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 multivalued attribute.

Multivalued attribute Relation

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

Teacher's : Primary key

$\{ \text{Teacher's, Course} \}$ } dependent relation
 $\{ \text{Teacher's, Book} \}$ }
 $\{ \text{Course, Book} \}$ } 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
 depending each
 other which is
 incorrect.

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

BCNF

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

Cite. To Avoid
 data loss requires
 to add more
 tuples)
 => There exist redundant
 data b/w (C) & (B)

⇒ Candidate Key: TCB

⇒ Highest Normal Form of R is BCNF.

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

Example -

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

Total (15) tuples

[TC]: repeated 3 times

[TB]: repeated 5 times

Definition of MVD:-

Let R be the Relational Schema X, Y are attributes of R and Z is $R - (XUY)$

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

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

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

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

* To show MVD, minimum 4 tuples are needed as per definition of 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	f.e.			
t ₂	X ₁	Y ₁	Z ₂				
t ₃	X ₂	Y ₂	Z ₁				
t ₄	X ₂	Y ₂	Z ₂				

$X \twoheadrightarrow Y$

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 ₁	Z ₂		X ₁	Y ₂	Z ₁
t ₃	X ₁	Y ₂	Z ₁		X ₁	Y ₁	Z ₂
t ₄	X ₁	Y ₂	Z ₂		X ₁	Y ₂	Z ₂

$X \twoheadrightarrow Y$

$X \twoheadrightarrow \bar{Z}$

① Complementation Rule:-

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

② Trivial MVD:- (Reflexive MVD)

if $X \supseteq Y$ (or) $XUY = R$ Then $(X \twoheadrightarrow Y)$ is trivial

Example:- $R(AB) \begin{cases} A \twoheadrightarrow A \\ A \twoheadrightarrow B \end{cases}$ Trivial

L

* To get redundancy due to single valued dependency we need at least two attributes 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:-

$$\text{if } X \twoheadrightarrow Y \text{ \& \& } Y \twoheadrightarrow Z \text{ Then } X \twoheadrightarrow (Z-Y)$$

$\downarrow$
ABC

$\downarrow$
CDE

$\downarrow$
DE

If there is no any common attribute
then $X \twoheadrightarrow Z$

[4] Augmentation:-

if $X \twoheadrightarrow Y$ and $W \supseteq Z$ then $XW \twoheadrightarrow YZ$.

Example:-

if $A \twoheadrightarrow B$
Then $ACD \twoheadrightarrow BC$

if $A \twoheadrightarrow B$
Then $ACD \twoheadrightarrow BCD$

* Every trivial multivalued dependency is also Reflexive multivalued dependency

[5] Replication:-

if $X \twoheadrightarrow Y$ Then $X \twoheadrightarrow Y$

* (Union / Splitting Not allowed)

Example 1-

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

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

$(A \rightarrow \rightarrow B)$ is trivial

while z 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 $C \rightarrow \rightarrow A$, $CB \rightarrow \rightarrow A$ (Due to argumentation)

$CB \rightarrow \rightarrow AB$

if $B \rightarrow A$ then $B \rightarrow \rightarrow A$, $BC \rightarrow \rightarrow A$ (Due to argumentation)

$BC \rightarrow \rightarrow AC$

$BC \rightarrow A$ 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 $M \twoheadrightarrow Y$ with X should be Superkeys.

Note:-

Superkeys or minimal superkeys or candidate keys identified by Functional Dependency only. Not from Multivalued Dependency.

Example:-

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

Candidate key (TCB)

Highest NF: BCNF

$\{T \twoheadrightarrow C, T \twoheadrightarrow B\}$
↑ ↑
Superkey Superkey

⇒ It is in BCNF not in 4NF.

Decompose it to be in 4NF.

T	C
A	DB
A	C
B	DB

(TC): Candidate key

$T \twoheadrightarrow C$
Trivial
MVD

T	B
A	K
A	N
B	K

(TB): Candidate key

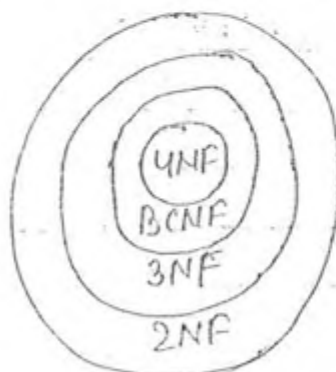
$T \twoheadrightarrow B$
Trivial
MVD

NO Non
trivial
MVD
4NF

Properties Results:-

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

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



Query Languages:-

1) Relational Algebra

2) SQL

3) Tuple Relational Calculus

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

Example

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

Query on R is

Query is a question on R

Query:- Retrieve Student whose marks greater than 50

Relational Algebra

SQL

Tuple Relational Calculus

$\pi, \sigma, \cup, -$
etc.

SELECT from
WHERE clause

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

Relational Algebra:-

* Always Eliminates duplicate tuple from the result

Ex. If

A	CS
A	CS
A	IT

It gives o/p only

A	CS
A	IT

Basic Operations:-

π : Projection

σ : Selection

$\times$: cross product

$\cup$: Union

$-$: Minus

ρ : Rename

Derived operations:-

$\cap$: Intersection

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

$\bowtie$: Join

$\div$: Division

Set operations:-

$\cap, \cup, -$: To apply these relation 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 R_1 is 2 & no. of attribute in R_2 is 3 then $\cup, \cap, -$ is not applicable there.

Ex:-

Sid	Sname		Sid	Sname	branch
S1	A		S1	A	C3

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

* not only the range of attribute of R_1 & R_2 are same but also the name of the attributes of R_1 & R_2 must be same for $\cup, \cap, -$.

Sid	Sname		Sid	branch
	↓			↓
	Varchar(2)			Varchar(2)

Sid	Sname	$\cup$	Sid	Sname
	↓	$\cap$		↓
	Varchar(2)	$-$		Varchar(2)

R & S cardinalities are m & n each.

$R \cup S$: [Range of Tuples] $\{ \max(m, n) \text{ to } m+n \}$

$R \cap S$: " $\{ \phi \text{ to } \min(m, n) \}$

$R - S$: " $\{ \phi \text{ to } m \}$

Cross Product:-

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

then $R \times S$ $\begin{cases} \text{Cardinality} : m * n \\ \text{Arity} : x + y \end{cases}$

Example:-

R		
A	B	C
1	2	3
2	2	1

S		
B	C	D
2	3	3
2	4	1

R x S				
A	B	C	B	C
1	2	3	2	3
1	2	3	2	4
2	2	1	2	3
2	2	1	2	4

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

⇒ Join 1 - Cross product followed by selection and then projection.

~~Cross~~

* Conditional Join :- ($\bowtie_c$)

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

$$R \bowtie S = \pi_{C_1, C_2} (\sigma_{C_1 < C_2} (R \times S))$$

Example: $R \bowtie S = \pi_{A, B, C, D} (\sigma_{R.A < S.B} (R \times S)) \Rightarrow$

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

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

A	B	C
1	2	3

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

* Left Outer Join:- $\bowtie$:-

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

$R \bowtie S$

A	B	C	D
1	2	3	3
2	2	Null	

* Right Outer Join:- $\bowtie$

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

$R \bowtie S$

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

* Full Outer Join:- $\bowtie$ S

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

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

- * (some/any/at least one) used as same in DBMS.
- * (All/every) used as same in DBMS.

DIVISION OPERATOR:-

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

Enrolled

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

Course

Cid	Cname
C1	-
C2	-
C3	-

Query1- Sid's who are enrolled some course.

Sid
S1
S2
S3

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

Query1- Sid's who are enrolled all courses:

O/p:- S1

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

* 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 Q Rel

$\pi_{sid}(\text{Enrolled})$ —

$$\frac{\pi_{sid, cid}(\text{Enrolled})}{\pi_{cid}(\text{course})} = \frac{\pi_{sid}(\pi_{sid, cid}(\text{Enrolled}) \times \text{course} - \text{Enrolled})}{\pi_{cid}(\text{course})}$$

Sid	Cid
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		

$$\frac{\pi_{sid, cid}(\text{Enrolled})}{\pi_{cid}(\text{course})}$$

$$\pi_{sid}(\text{Enrolled}) - \left[\pi_{sid}(\pi_{sid, cid}(\text{Enrolled}) \times \text{course} - \text{Enrolled}) \right]$$

(Universal)

(Every student Enrolled Every Course)

(Actual Enrolled)

(Student who are

Enrolled Some course)

(At least one course)

$$= \begin{array}{|c|c|} \hline Sid & Cid \\ \hline S2 & C3 \\ S3 & C2 \\ S3 & C3 \\ \hline \end{array} = \begin{array}{|c|} \hline S2 \\ S3 \\ \hline \end{array}$$

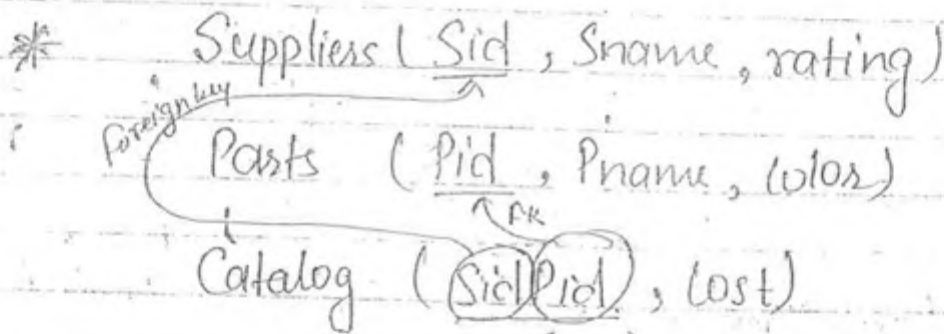
(Complement Data)

(Students who are not Enrolled all courses)

disqualify dot

23/08/11

*



Query 1 - Sid's of Suppliers whose rating greater than 10.

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

Query 1 - Sid's of suppliers whose rating greater than 10 or who supplied some part.

Suppliers

Sid	rating
S1	5
S2	15
S3	7

Catalog

Sid	Pid	
S1	P1	-
S2	P1	-
S1	P2	-

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

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

Quesy1- Retrieve Name's of Suppliers whose rating > 10 (and) who are also supplied some part.

Suppliers

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

Catalog

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

By join we eliminate S3 because it not supply any part because S3 is not in Catalog sid but it is in supplier sid

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

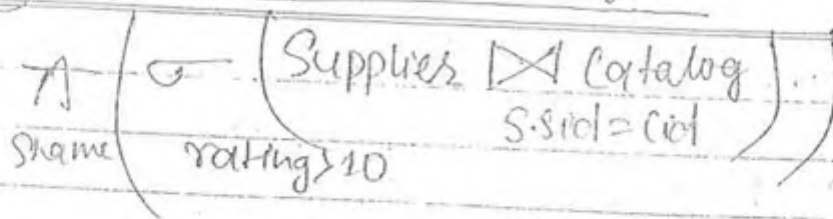
OR

By using natural join

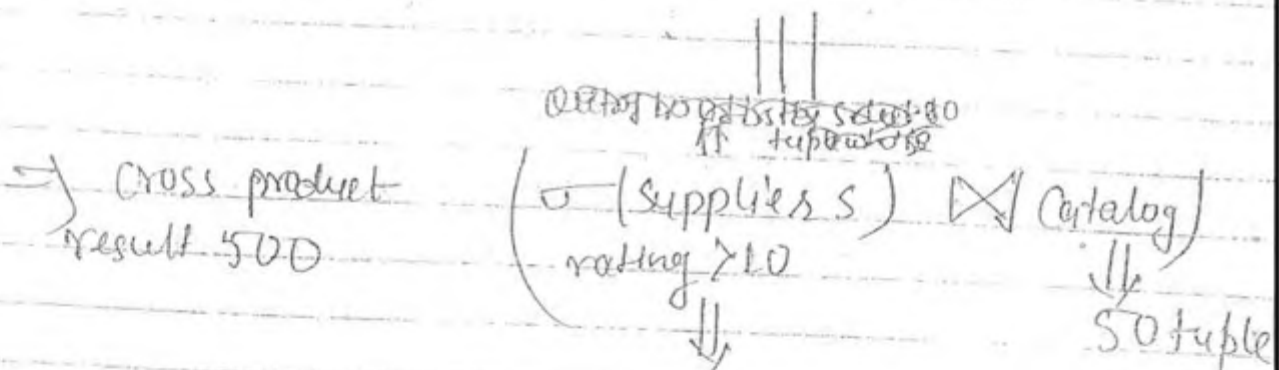
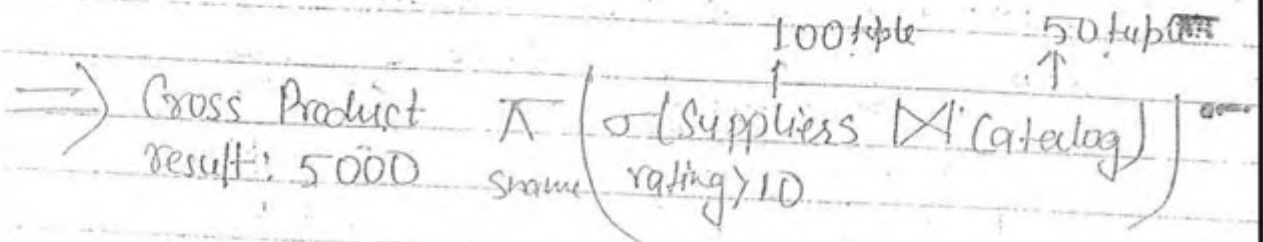
[Q2] $\pi_{sname} \left(\sigma_{rating > 10} (suppliers \bowtie catalog) \right)$

[Q3] $\pi_{sname} \left(\sigma_{s.sid = c.sid \wedge rating > 10} (suppliers \bowtie_{\text{apply conditional join}} catalog) \right)$

[04] By apply conditional join:-



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



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

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

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

|||

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

because rating attribute is only belong to the suppliers 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 cost of the whole relation.

*

Let $A \in \text{Relation } R \text{ only.}$

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

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

~~$\sigma_{(Suppliers \bowtie Parts)}$~~

[Q1] $\pi_{Name} \left(\sigma_{(Suppliers \bowtie Catalog \bowtie Parts)} \right)$
 $Color = Red$

|||

[Q2] $\pi_{Name} \left(\sigma_{(Suppliers \times Catalog \times Parts)} \right)$
 $S.Sid = C.Sid \wedge$
 $C.Pid = P.Pid \wedge$
 $Color = RED$

|||

[Q3] $\pi_{Name} \left(\pi_{Sid} \left(\sigma_{(Parts)} \bowtie Catalog \right) \bowtie Suppliers \right)$
 $Color = RED$

|||

if foreign key constraints is not there then

max no. of Tuples in sup $\bowtie$ catalog = 100
 min " " " " " " " = 0

Ques) $R(ABCD)$ $S(BDE)$

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

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

How many max no. of tuples in $R \bowtie S$.

Solⁿ:- Maximum no. of tuples = 1000.

Rename Operator:-

Two syntax for Rename operator

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

T		
Sid	Prd	C

 2) Rename the Column $\rho(\text{Catalog})$

Catalog		
S	P	C

 $\rho(\text{Catalog})$

Catalog		
S	P	Cost

$\rho(\rho(T, \text{Catalog}))$
 $\rho(\text{Catalog})$

$\rho(T)$

T		
S	P	C

* Use of Rename Operator:-

$$\sigma_{\text{Catalog.Pid} = \text{Parts.Pid} \wedge \text{col} = \text{RED}} (\text{Catalog} \times \text{Parts}) \equiv \sigma_{\{ \rho(c, \text{Catalog}) \times \rho(p, \text{Parts}) \}} \\ \text{C.Pid} = p.\text{Pid} \wedge \text{col} = \text{RED}$$

$$\equiv \sigma_{\text{Picl} = p.\text{Picl} \wedge \text{col} = \text{RED}} (\text{Catalog} \times \rho(\text{Parts}))$$

Query 1 - Retrieve Sid of the Suppliers whose supplies some red part or some green part.

[Q1]

$$\pi_{\text{Sid}} \left(\text{Catalog} \bowtie \sigma_{\substack{\text{col} = \text{GREEN} \\ \text{col} = \text{RED}}} (\text{Parts}) \right)$$

By using union operator:-

$$\rho \left(T_1, \pi_{\text{Sid}} \left(\text{Catalog} \bowtie \sigma_{\text{col} = \text{RED}} (\text{Parts}) \right) \right)$$

$$\rho \left(T_2, \pi_{\text{Sid}} \left(\text{Catalog} \bowtie \sigma_{\text{col} = \text{GREEN}} (\text{Parts}) \right) \right)$$

$$T_1 \cup T_2$$

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

$$\Rightarrow [Q1] \{ (T1, \pi_{sid} (catalog \bowtie \sigma_{(parts)}))$$

$$\{ (T2, \pi_{sid} (catalog \bowtie \sigma_{(parts)}))$$

$$T1 \cap T2 \Rightarrow (T1 \cup T2)$$

$$\times [Q2] \pi_{sid} (catalog \bowtie (\sigma_{(parts)} \wedge \text{color} = \text{GREEN} \wedge \text{color} = \text{RED}))$$

* [Q1] is true But [Q2] is not correct.

* Without using the union operator:-

$$\{ (P1, Parts)$$

$$\{ (P2, Parts)$$

$$\{ (C1, Catalog)$$

$$\{ (C2, Catalog)$$

$$\pi_{C1.sid} \left(\sigma_{C1.sid = C2.sid} \left((C1 \bowtie \sigma_{(P1) \wedge \text{color} = \text{RED}}) \times (C2 \bowtie \sigma_{(P2) \wedge \text{color} = \text{GREEN}}) \right) \right)$$

By using natural join :-

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

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

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

Sid of the suppliers who supplied every red parts.

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

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

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

Example:-

Catalog

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

Part

Pid	Color
P1	R
P2	G
P3	R

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

$$\frac{\pi_{\text{pidsid}}(\text{Catalog})}{\pi_{\text{sid}}(\text{Supplies})} =$$

$$\pi_{\text{pid}}(\text{Catalog}) - \pi_{\text{pid}} \left(\pi_{\text{pid}}(\text{Catalog}) \times \pi_{\text{sid}}(\text{Supplies}) - \pi_{\text{pidsid}}(\text{Catalog}) \right)$$

These are parts not supplied by every supplier.

Query: - Retrieve Sid of the Suppliers who supplied at least two parts.

$$\sigma_{\text{pid} \neq \text{pid}}(\pi_{\text{pid}}(\text{Catalog}))$$

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

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

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

$$\pi_{\text{sid} = \text{sid}}(\pi_{\text{pid}}(\text{Catalog}) \times \pi_{\text{pid}}(\text{Catalog}))$$

$\rho(T1, \text{Catalog}) \quad \rho(T2, \text{Catalog})$

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

|||

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

Query 4: Retrieve Sid of the Supplier who supplied at least Three parts.

$\rho(T1, \text{Catalog}) \quad \rho(T2, \text{Catalog}) \quad \rho(T3, \text{Catalog})$

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

Query 1: Retrieve Sid's of suppliers who supplied exactly two parts.



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

$\pi_{sid}(\text{suppliers}) - \left| \begin{array}{l} \text{sid's of suppliers who} \\ \text{supplied at least three parts} \end{array} \right|$

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.

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

$\pi(T1, \text{Catalog})$
 $\pi(T2, \text{Catalog})$

$\pi_{T1.sid, T2.sid} \left(\begin{array}{l} \sigma(T1 \times T2) \\ T1.cost > T2.cost \wedge \\ T1.pid = T2.pid \end{array} \right)$

~~$T1.sid \neq T2.sid$~~ No need to write this condition because of Key constraints because sid & pid are Primary key. So, sid must not be repeated.

Ques) Retrieve Srid of the Suppliers who supplied most expensive cost.

Srid	Prid	Cost		Srid	Prid	Cost
S1	P1	10	X	S1	P1	10
S2	P1	20		S2	P1	20
S3	P2	30		S3	P2	30

Srid	Prid	Cost	Srid	Prid	Cost
S1	P1	10	S1	P1	10
S1	P1	10	S2	P1	20
S1	P1	10	S1	P2	30
S2	P1	20	S2	P1	10
S2	P1	20	S2	P1	20
S2	P1	20	S1	P2	30
S3	P2	30	S1	P1	10
S3	P2	30	S2	P1	20
S3	P2	30	S1	P2	30

(S1 P1 10)
(S2 P1 20)

Suppliers Info who
doesn't supplied
most exp part

(S2 P1 20)
(S1 P2 30)

Suppliers Info
who doesn't supplied
least expensive part

$$\rho(T1, \text{Catalog}) \quad \rho(T2, \text{Catalog})$$

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

$$\pi_{Sid} (\text{Catalog} - T3)$$

⇒ Suppliers who supplied least expensive part:

$$\rho\left(T3, \pi_{T1.Sid, T1.Cost} \left(\sigma_{T1.Cost > T2.Cost} (T1 \times T2) \right) \right)$$

$$\pi_{Sid} (\text{Catalog} - T3)$$

OR

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

$$\pi_{Sid} (\text{Catalog} - T3)$$

Query1- Retest the supplier who supplied 2nd most expensive part.

$\rho(T1, \text{catalog}) \quad \rho(T2, \text{catalog})$

$\rho\left(T3, \begin{array}{c} \overline{\wedge} \left(\begin{array}{c} \sigma(T1 \times T2) \\ T1 \cdot \text{cost} < T2 \cdot \text{cost} \end{array} \right) \\ T1 \cdot \text{sid}, T1 \cdot \text{pid}, \\ T1 \cdot \text{cost} \end{array} \right)$

$\rho(T4, T3)$

$\rho\left(T5, \begin{array}{c} \overline{\wedge} \left(\begin{array}{c} \sigma(T4 \times T3) \\ T3 \cdot \text{cost} < T4 \cdot \text{cost} \end{array} \right) \\ T3 \cdot \text{sid}, T3 \cdot \text{pid}, \\ T3 \cdot \text{cost} \end{array} \right)$

$\overline{\wedge}_{\text{sid}} (T3 - T5)$

T3		
Sid	Pid	Cost
S1	P1	10
S2	P1	20

T5		
Sid	Pid	Cost
S1	P1	10
S2	P1	20

(Ques) Enroll (sicd, cicd)

Student (sicd , name, sex)

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

what this relation algebra query represent

$$\pi_{cicd} \left(\left(\pi_{sicd} \left(\sigma_{sex=f} (Student) \right) \right) \times \pi_{cicd} (Enroll) \right) - enroll$$

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

$$\pi_{cicd} \left(\left(\pi_{sicd} \left(\sigma_{sex=f} (Student) \right) \right) \times \pi_{cicd} (Enroll) \right) - enroll$$

All Female Student All Course

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

(every 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		
S2	X	C2	S1	C2	S1	C1
S3	C3	✓	S1	C3	S2	C1
			S2	C1	S3	C1
			✓ S2	C2	S1	C2
			S2	C3	S2	C3
			S3	C1	S3	C3
			✓ S3	C2		
			S3	C3		

Course

C2 $\Rightarrow$ {S1}

C3 $\Rightarrow$ {S2 S3}

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

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

$$\pi_{sid} (\sigma_{sex=female} (student)) = \pi_{sid} (student \bowtie \rho (student))$$

sex = 'female'

X = 'male'

marks $\leq$ 100

All Female student

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

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

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

Student	Sid	marks	Sex
S1	10	M	
S2	20	F	

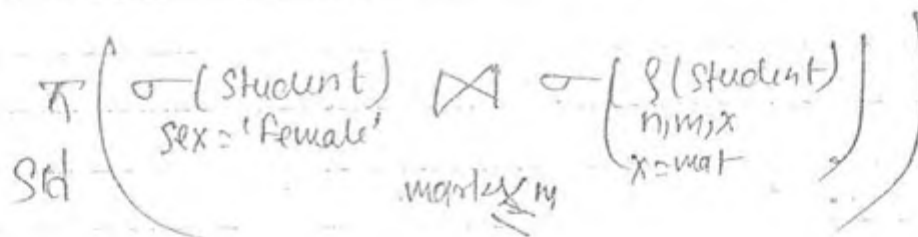
Student	n	m	x
S1	10	M	
S2	20	F	
S3	30	F	

S2	20	F
S2	20	F

S1	10	M
S2	20	F
S3	30	F

S1
S1
S2
S1
S2

Solⁿ 1-



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) " " not less than some male student
 c) " " more marks than all male student
 d) " " more than equal to " " "

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

$$* \{ \forall, > \} \Leftrightarrow (\text{Universal Set}) - \{ \exists, < \}$$

$$② \{ \forall, < \} \Leftrightarrow (\text{Universal Set}) - \{ \exists, > \}$$

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

$$④ \{ \exists, < \} \Leftrightarrow (\text{Universal Set}) - \{ \forall, > \}$$

$$* > = \text{All} \Leftrightarrow \text{not} < \text{Some}$$

$$> \text{All} \Leftrightarrow \text{not} < = \text{Some}$$

$$> = \text{Some} \Leftrightarrow \text{not} < \text{all}$$

$$> \text{some} \Leftrightarrow \text{not} < = \text{all}$$

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

Ques) Which one of the following Query are equivalent

✓ 1) $\pi_P(R \bowtie S)$

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

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

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

1) $\pi_P \left(\begin{array}{l} \sigma(R \times S) \\ R.P = S.P1 \\ R.Q = S.Q \end{array} \right) \quad \begin{array}{|c|} \hline P \\ \hline 1 \\ \hline \end{array}$

R	S
P Q	P Q
1 2	1 2
2 3	2 4
3 3	4 3

2) $\pi_{R.P} \left(\sigma \left(\pi_P(R) \times \pi_P(S) \right) \right) \quad \begin{array}{|c|} \hline P \\ \hline 1 \\ 2 \\ \hline \end{array} \quad \begin{array}{l} 1 \\ 2 \\ 3 \end{array} \bowtie$

24/08/2011

SOL- (Structured Query Language)-

Sublanguage of SOL-

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 Suppliers)

(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

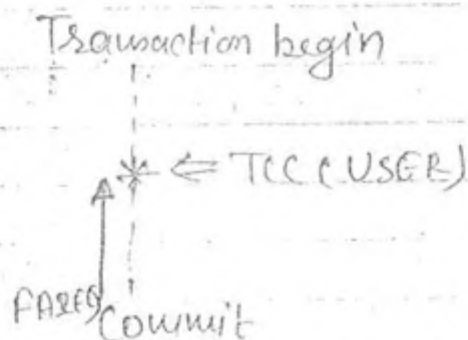
* Independent of physical & conceptual schema.

3) DCL (Data Control Language):-

a) Roll back / Abort

b) Commit

c) Transaction consistency check point:-



4) Data Query Language:-

Used to retrieve data from DB.

a) Select,

b) From,

c) Where,

Query:-

SELECT DISTINCT A1, A2, ..., AN

FROM R1, R2, ..., Rm

where P

* To eliminate duplicate Tuple from result use (DISTINCT)

* whatever the data is coming from R1 is equal to (R1 x R2 x ... x Rm)

* whatever the condition is selected is comes under where

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

$\pi_{A1, A2, \dots, AN} \left(\sigma_P (R1 \times R2 \times \dots \times Rm) \right)$ $\rightarrow$ By default it eliminates the duplicate tuples in the result.

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

* SELECT & FROM clause are mandatory clause. ^{operator in Relational Algebra}
 Basic SQL Query: (Operators) :- ^{But using these clause we can't create any SQL query.}

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

[1] FROM R1, R2, ..., Rm

[2] [WHERE P]

[3] [GROUP BY (Attribute)]

[4] [HAVING condition]

[ORDER BY (Attribute) [DESC]]

[1] FROM Clause - Cross Product of 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 result (table) based on specified attribute.

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

Q.3 Group by (Branch) : It group the entire table based on Branch.

O/P:-

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

Aggregation Operators:-

⇒ COUNT ([DISTINCT] Attr)

⇒ SUM ([DISTINCT] Attr)

⇒ AVG ([DISTINCT] Attr)

⇒ MIN (Attr)

⇒ Max (Attr)

⇒ (Aggregation always discard Null value
i.e. in every aggregation Null value is not counted.)

* If count is based on single attribute, then it checks for null value.
If count is based on entire table, it counts all the tuple of the table.

$$\text{count}(\text{marks}) = 5$$

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

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

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

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

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

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

* Arithmetic operation with Null results Null.

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

Ques)

RNO	Marks
1	10
2	20
3	30
4	NULL

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

What the o/p of the Select Statement.

Soln:-

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

* (AS) (rename operator):-

Rename for either

attribute or Table.

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

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

O/p:-

minmarks
50

Ques:-

Syntax
Error

Select Branch, min(marks)
From student.

O/p:- CS 50
IT 50
EE 50

It is not able to display o/p in tabular form. So, it can't be displayed by SQL. So, Syntax Error

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

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

O/p:- CS 50 Here min(marks) is applied on every
ET 50 group of the table.
EC 80

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

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

O/p:- 50
50
80

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

Query1- Select Sid, marks O/p

① FROM student	S1	50
② GROUP BY (Branch)	S4	70
	S2	60
	S3	50
	S5	80

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

} It is not allowed

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

Synotatically allowed

Query 1 - Select Sid, ^{branch}marks, Avg(marks)
From Student
GROUP BY (Sid, branch)

} Allowed

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

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

Q/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 1 - Retrieve Students whose group average marks more than 50.

```

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

O/P1 - Avg	SS1	50	CS	1st group	} Satisfy the Having Clause Condition
60	SS4	70	CS		
80	SS5	80	CS	2nd group	

Example 1 - Employee (name, sex, salary, dname)
SQL Query -

```

① Select deptName
② From Employee
③ Where sex = 'M'
④ Group By dname
⑤ Having Avg(sal) > 10000
  
```

Select (Avg(salary))
 FROM Emp
 company avg salary

What this SQL Query Return?

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

Retasnt

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

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

SET OPERATORS:-

UNION / UNION ALL
INTERSECT / INTERSECT ALL
MINUS (US) EXCEPT / MINUS ALL \

$$\pi_{Pname} \left(\sigma_{(Catalog \times (parts))} \right) \cup \pi_{Pname} \left(\sigma_{(Catalog \times (parts))} \right)$$

C.Pid = P.Pid C.Color = RED C.Color = GREEN

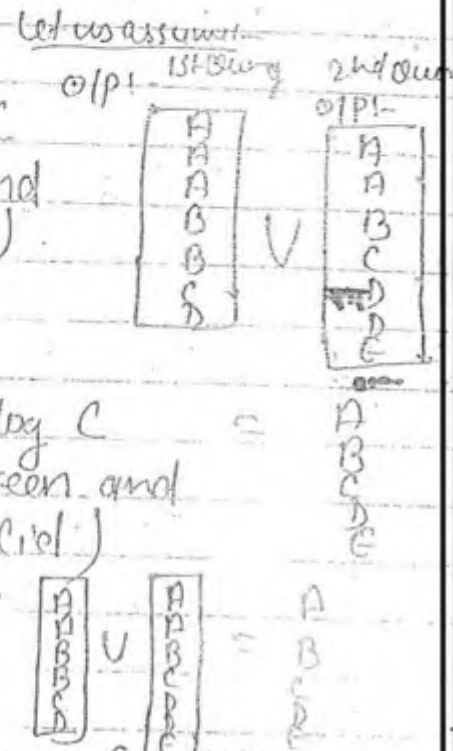
Query 1 - Retrieve Pnames of the red parts or green parts that are supplied by some supplier

(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.Pid)

UNION : discards duplicates in Result.



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

SQL 1999! Standard! Having clause Condition can be applied every tuple of each group by using two more clause { ANY, EVERY }

Ex 1

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

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

[04]

$$\pi_{\text{name}} \left(\pi_{\text{sid}} \left(\pi_{\text{pid}} \left(\sigma_{\text{col}=\text{Reol}} (\text{Parts}) \right) \bowtie \pi_{\text{sid}=\text{cid}} (\text{Catalog}) \right) \bowtie \pi_{\text{sid}=\text{same}} (\text{Supplies}) \right)$$

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

Check for Commutative:-

1) π (Projection) = $\times$

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

2) σ (Selection) = $\checkmark$

$$\sigma_{c2} \left(\sigma_{c1} (R) \right) = \sigma_{c1} \left(\sigma_{c2} (R) \right) = \sigma_{c1 \wedge c2} (R) = \left(\sigma_{c1} (R) \right) \cap \left(\sigma_{c2} (R) \right)$$

3) Cross Product = $\checkmark$

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

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

5) Intersection ($\cap$)!- ✓
 $R1 \cap R2$

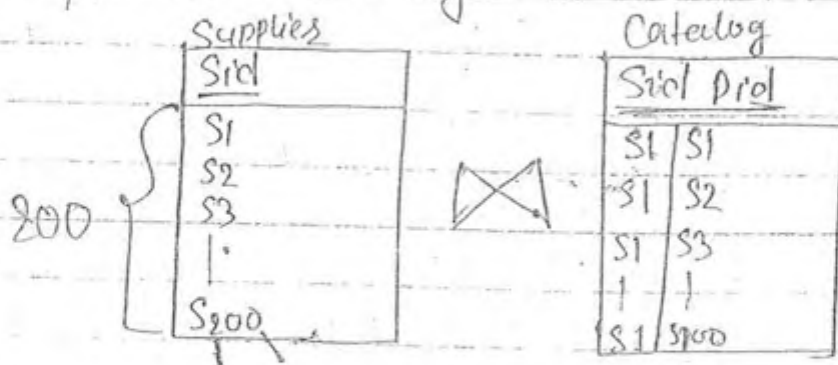
6) Set Difference ($-$)!- X

7) / (Division)!- X

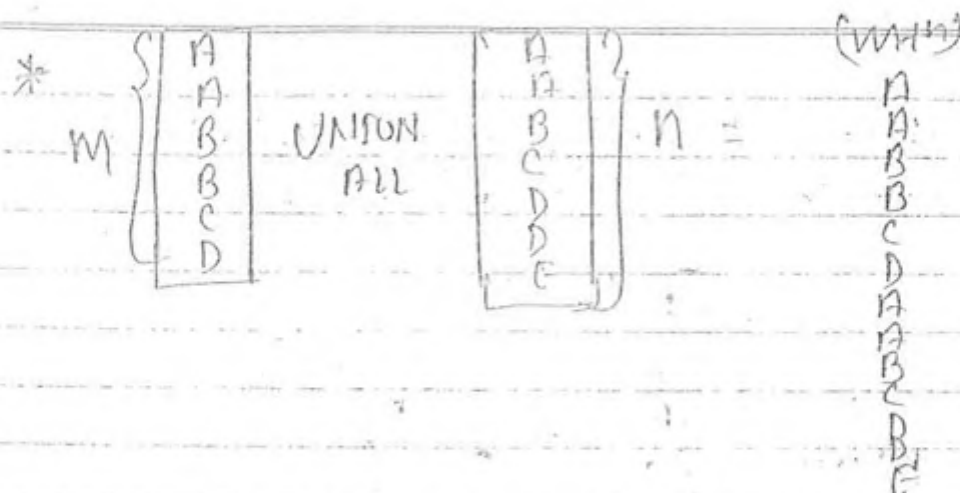
$$A(X,Y)/B(Y) \neq B(Y)/A(X,Y)$$

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.



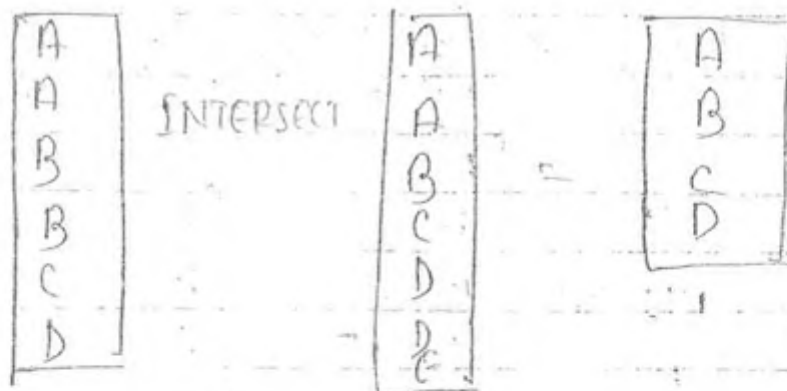
max no. of tuples in $\text{Sup} \bowtie \text{Catalog} = 100$
 min no. of tuples in $\text{Sup} \bowtie \text{Catalog} = 100$ (beoz cataly
 beoz sid is foreign key of suppliers Sid so, sid reference
 minimum 100 tuples is necessary. Suppliers Table)



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

* INTERSECT

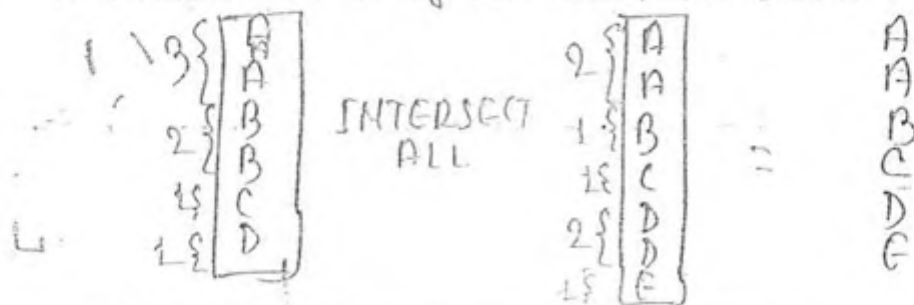
Discards duplicates in result



* Common tuple of both the relation with duplicate tuples.

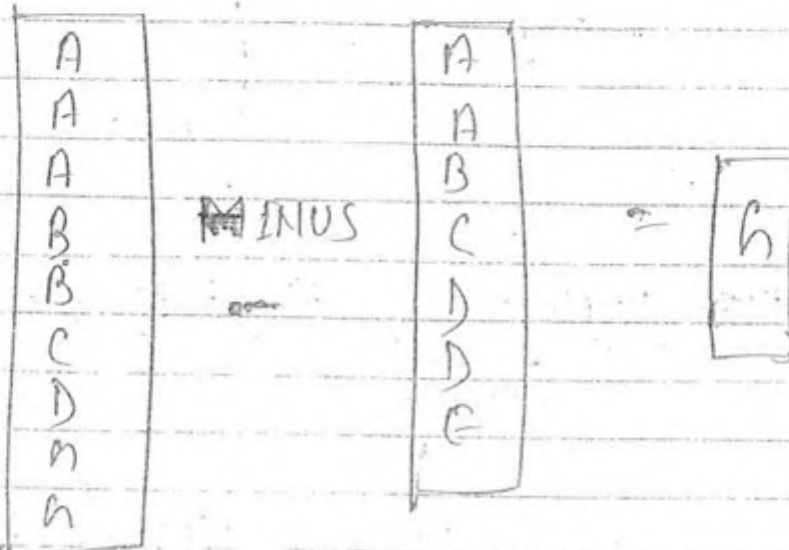
* INTERSECT ALL

* ~~Discards~~ ^{Maximum} no. of common in both the set.

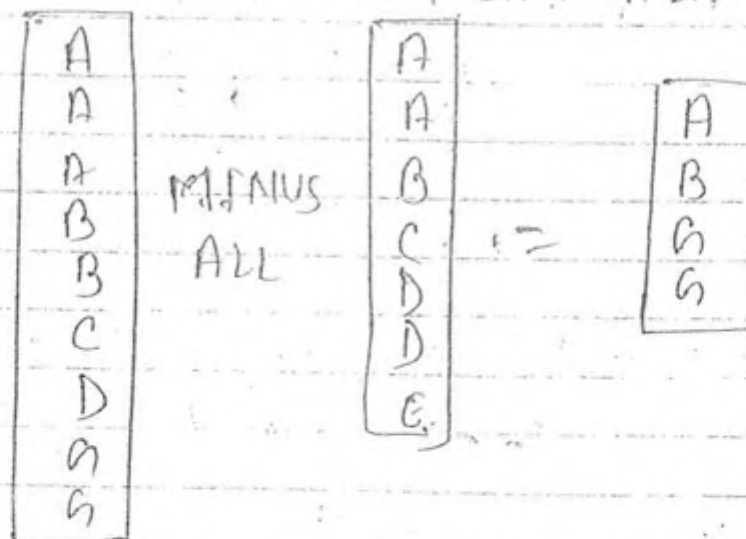


MINUS

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



MINUS ALL — ^{duplicates} No. of repetition in the first relation - no. of
duplicates in 2nd relation.



Nested Queries:-

SQL supports Query inside From

- 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:-

Independent
Nested Query



(Inner Query independent
of outer Query)

Correlated
Nested Query



(Inner Query uses attribute
that is specified in the
outer Query)

Example:- Retrieve Sid of the Suppliers 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

O/P:

S1
S1
S3

//

Sid	Pid
✓ S1	P1
X S2	P2
✓ S1	P3
✓ S3	P5

SELECT P.D
FROM Parts
where Color = Red

//

Pid
P1
P3
P5

SOL 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

FROM Parts

where Color = Red) P

where S.Pid = P.Pid

Quesy:- Emp(Eno, Ename, rating)

Dependent (Eno Dname, age)

Retrieve Employee names who have no dependent.

[Q1]

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

[Q2]

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

- [a] Q1 correct but not Q2
[b] Q2 " " " Q1
[c] Q1 & Q2 correct
[d] Q1 & Q2 incorrect

Example:

Eno
E1
E2
E3
E4

Eno	Dname
E1	A
E1	B
E2	A

O/p Q1

[E1
E2
E3
E4]

[Q2]

[E3
E4]

Set Comparison operators:-

$x \text{ IN } \{ \dots \} \Rightarrow y \Rightarrow x \text{ is in given set as not}$
if x is in set then it True
Otherwise False.

$\Rightarrow$ Op ANY

$\Rightarrow$ Op ALL

$\Rightarrow$ Op $<, <=, >, >=, =, \neq$
(not equal to)

Example:-

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

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

$x > \text{ALL } \{ \dots \}$

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 Sid
FROM SUPPLIER
WHERE rating > (SELECT rating
FROM SUPPLIER S, Catalog C, Parts P
WHERE S.Sid = C.Sid and
P.Pid = C.Pid and
C.Wlos = Red)~~

Select Sid

FROM SUPPLIER

WHERE rating > ANY (Select rating

FROM SUPPLIERS, Catalog C, facts

Where S.Pid = C.Sid and

P.Pid = C.Pid and

Color = RED.)

~~Query!~~ Retrieve Sid of the Suppliers whose group average rating is more than the suppliers rating of every group of rating. Age = max(age)

~~Query!~~ Retrieve Sid of the Supplier whose ^{max}age is greater than all ~~FROM~~

Query Sid's of Suppliers who are highest age.

Suppliers (Sid, Sname, Age)

age = $\max(\text{age})$
Select Sid, age

FROM Suppliers

where age = (Select max(age)
FROM Suppliers)

* Aggregation is done on Attribute

age = max(age) is not correct

* Modification only create 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			Returns Sid, Sname of the suppliers who supply some parts.
Sid	Pid	Cost	
✓ S1	S1	P1	
✓ S2	S1	P2	
X S3	S2	P3	
X S4			

* It is work as nested for loop 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 false then supplier tuple is not select.

EXISTS (class)

Returns true if inner Query results non-empty.

Query

Student (Sid, Sname, marks)
Enrolled (Sid, Cid)

No Foreign Key, No NULL value in the DB.

Query 1)

Select Sid
FROM student
where Sid in (Select sid
from Enrolled)

1 X S1
S2

S1
S2
S3

S2
S3

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

S1

S1

S2

[Query 3] Select S.Sid
 FROM Student S, Enrolled E
 Where S.Sid = E.Sid

S1

S1

S2

[Query 4] Select S.Sid
 FROM Student S
 Where EXISTS (Select *
 From Enrolled E
 Where E.Sid = S.Sid)

S1

S2

Which is true:-

- [a] all Queries returns identical row for any DB.
☒ [b] Q1 not same as Q2 for some DB but Q2 & Q3 same for every DB.
☐ [c] Q1 & Q2 same for any DB and Q4 returns fewer row set than Q3.
☐ [d] Q4 encounters integrity violation at run time.

Student
Sid
S1
S2
S3
S4

Enrolled
Sid Cid
S1 C1
S1 C2
S2
S5

Ques) movie (id, title, yr)
 actor (id, name)
 casting (movieid, actorid)

Write an SQL Query to return titles of the all 1959 Monroe films.

Soln:-

Select Titles
 From Movie, actor a, casting c
 where m.id = movieid and
 a.id = actor id 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
T1	40
T3	50
T4	60

20

(B)	Title	Price
X	B1	10
X	B2	20
✓	B3	30
✓	B4	40
✓	B5	50
✓	B6	60
✓	B7	70

* If there is no any data then "Count" returns "zero"

Que) 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 correct but Q1 results correct for some database but both are incor.

d) Q1 & Q2 returns identical rowset for some database

Input		o/p	
Cid	Balance	Cid	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	C1	✓	10 20 ✓
C3	20		C3	20	C1	✓	10 20 ✓
C4	30		C4	30	C1	✓	10 30 ✓
					C2	X	20 10
					C2	✓	20 20
					C2	✓	20 20
					C2	✓	20 30 ✓
					X	20	10
					C3	✓	20 20
					C3	✓	20 20
					C3	✓	20 20
					C3	✓	20 30 ✓
					X	30	10
					X	30	20
					X	30	20
					C4	✓	30 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 ~~True~~ "non empty".

Similarly

"NOT EXISTS" return True when inner condition is ~~True~~ "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.Sid = S.Sid and
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	P2
	S3 P2	

(Sid) — { All parts supplied — (enroll)
all suppliers

O/p! - It returns all the suppliers

* This above program is just work as division operation.

25/08/2011

SOL1:-

Supports pattern matching of strings.

"0/*" $\Rightarrow$ used to denote 0 or more characters

"_" $\Rightarrow$ used to denote Any single character

Example:- "Names" Starts with 'D'

'D_*_*.A'

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

'D_*_*.A'

(Ques) If we want to retrieve 'MADE*EASY' string.

All use Escape character '\'

used treat
patter character as like normal character

$\Rightarrow$ 'MADE\EASY'

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

Select *

From Student

whose Name 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.)

Empno	Empname	PPno
E1	A	P1
E2	A	NULL
E3	B	P2

Query:- Retrieve Employee who have no passport.

```
Select *  
FROM Employee  
Where PPno like NULL
```

} It is not
compare with
null value

So we use IS/ISNOT clause

⇒ IS/ISNOT! - Compare with NULL values.

```
Select *  
FROM Employee  
Where PPno is NULL
```

There are three truth value in DBMS

- True $\Rightarrow$ If it is '0'
- FALSE $\Rightarrow$ If it is '1'
- UNKNOWN: Comparison with Null value is treated as Unknown.
then it is $\frac{1}{2}$ (It is not true, not false, it is between true & false)

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

$\Rightarrow$	OR	T	F	U	AND	T	F	U
	T	T	F	T	T	T	F	U
	F	T	F	U	F	F	F	F
	U	T	U	U	U	U	F	U

$\Rightarrow$ NOT UNKNOWN = UNKNOWN

Ques) Relation can contain Null values. Suppose all comparison of with Null value are treated false.

Which of the following pairs is not equivalent?

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

b) $x = 5$ $x > 4$ 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.H.S & R.H.S if the above questions we obtain the values)

a) $x = 5$ $\boxed{5}$

$\text{not}(\text{not}(x = 5))$ $\boxed{5}$ value of x is 5

b) $x = 5$ $\boxed{5}$

$x > 4$ and $x < 6$ $\boxed{5}$ value of x is 5

c) $x \neq 5$ $\boxed{6}$

$\text{not}(x = 5)$ $\boxed{6}$

Transaction and Concurrency Control

Transaction

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

Example

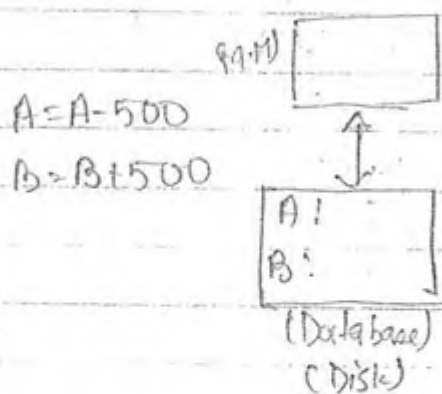
T1: Transfer 500 Rs from Account A to Account B

Read(A)

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

Write(A)

Updates data items A into the database.



begin Trans

$R(A)$

$A = A - 500$

$W(A)$

$R(B)$

$B = B + 500$

$W(B)$

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 becoz of

- 1) Power failure
- 2) S/W failure
- 3) H/W failure
- 4) Disk Crash. Etc.

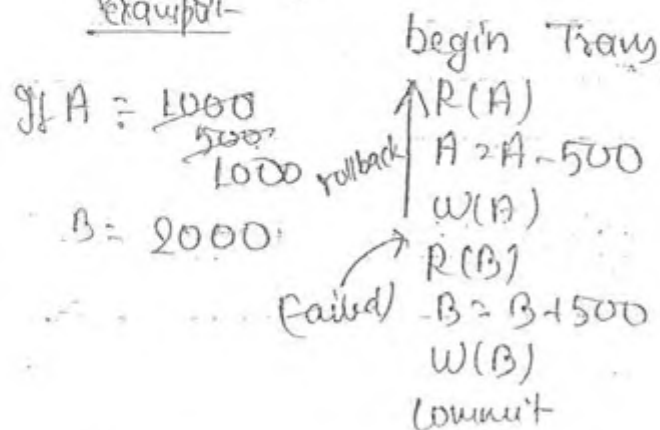
⇒ Recovery Management component performs the Rollback or Abort.

⇒ 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 arbit.

Example:-



i.e. rollback is done with the help of Transaction Log file.

Transaction Log:-

File created by DBMS for every Transaction to keep track of Transaction Operations.
Used to perform Rollbacking.

⇒ The main purpose of transaction log is Rollbacking

Undo! operation:-

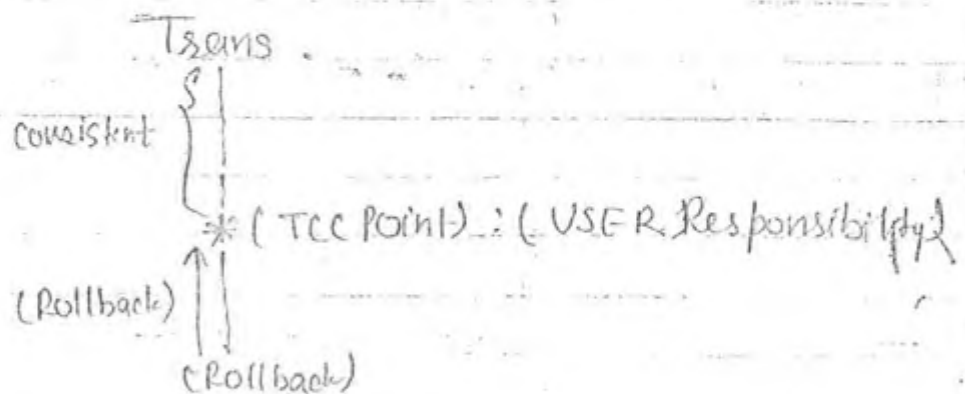
Database is updated by old values of the log file.

Redo! operation:-

Updating the database by new values.
Database updated with New Values.

⇒ Log file retained in the DB until transaction committed or rollbacking completed, then ^{after} the log file is automatically removed.

Transaction Consistent Check Point:-



C: Consistency:-

DB should be consistent before & after execution of the transaction.

⇒ Consistency is checked by Users 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 transactions.

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:-

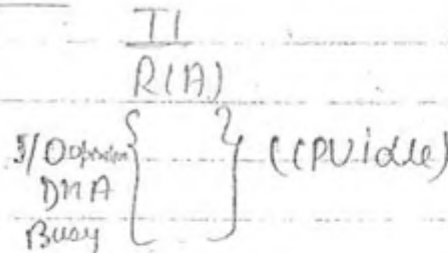
T1	T2	T1	T2
R(A)		R(A) : 1000	
W(A)		R(B) : 2000	
R(B)		3000	
W(B)		R(A)	
		W(A)	
	R(A) : 500	R(B)	
	R(B) : 2500	W(B)	

T1: T2 3000 T2: T1

⇒ If n Trans then
n! Serial schedule

⇒ All serial schedules are consistent.

⇒ Less throughput:-



because of this i.e. because most of the CPU is idle due to the S/O operation! —

⇒ Poor Resource Utilization

⇒ More Response time.

Concurrent Schedule:-

Interleaved Execution or simultaneous Execution of two or more transac

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)			2500
	R(B) 2500		
	3000		

[consistent] (Equal to T1 T2 serial)

[Inconsistent]

T1 → T2 ✓
T2 → T1 ✗

Advantages:-

- 1) More throughput
- 2) less Response time
- 3) Better Resource Utilization.

DisAdvantages:-

Concurrent Execution May result in con-
fency.

(Ques) How many Total Schedules are possible in
T1 & T2 Transaction.

$$\frac{6!}{4!2!}$$

$$\frac{6!}{4!2!} \text{ (By Permutation) i.e. } 6C4 \times 2$$

$$= 6C4 \times 2$$

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.

Isolation:-

Concurrent execution of two or more transaction should be equal to any serial schedule then it's consistent. (we called it as (Serializable Schedule).

then it's ~~are~~ taken by

$\Rightarrow$ concurrency control management component

D: Durability:-

Database should be recoverable under any kind of failure.

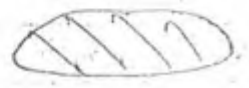
RAID 1 - Redundant Array of Independent Disks.

RAID 0 - Redundant Array of Independent Disks
not durable

RAID 1 - Mirror Image



1st Disk



2nd Disk

It is durable

Disadvantage: write operation is too costly in RAID 1.

Problems Because of Concurrent Execution:-

[1] WR Problem :- [Read After Write Problem]

T1	T2
W(A)	
	R(A): Uncommitted Read (or) Dirty Read

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

Inconsistent (WR Problem)

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

Consistent (No Problem)

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 find out that which problem is exist in this transaction due to concurrency. Problem is not just found after seeing the transaction. Firstly we have to check if that if transaction is consistent then no problem, but if the transaction is not consistent then after we check, due to which problem it is inconsistent. 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)
Rollback	

T1 performed Read(A) before commit/rollback of T1, trans T2 updated data item A.

Example:-

A: No. of copies of Textbook. To issue one book from A book then what operation are required during transaction.

T
R(A)
if (A > 0)
A = A - 1
W(A)
commit

if Transaction T1 & T2 both simultaneously request for the same book. the transaction operation

T1	T2
R(A)	
if (A > 0)	
A = A - 1	
	R(A)
	if (A > 0)
	A = A - 1
	W(A)
	commit
W(A)	
commit	

if initially A = 10 (due to RW prot)
↑
due to transaction T2 due to transaction T1

Inconsistent (RW Problem Exist)

	T1	T2
A = 10	R(A)	
		W(A)
A = 9	R(A)	

(Temporary Read Problem) read by the Transaction (T1)

OR
(Phantom phenomena Problem)

[S] WW Problem :- [Write After Write Problem]

T1	T2
W(A)	
	W(A)

(Write-Write Problem is possible)

Example:-

T1: Set A, B values as 1000 (W(A), W(B))

T2: Set A, B values as 2000 (W(A), W(B))

If T1: T2 Serial: A = B = 2000
(Firstly Trans T1 execution then T2 executed)

T2: T1 Serial: A = B = 1000
(Firstly Transaction complete its execution then T1 starts)

T1	T2
$w(A)$	
	$w(A)$
	$w(B)$
$w(B)$	

A = ~~1000~~
2000

B = ~~2000~~
1000

(Non-Serializable) or
Inconsistent
Because of ww Problem

T1	T2
$w(A)$	
	$w(A)$
$w(B)$	
	$w(B)$

A = ~~1000~~
2000

B = ~~1000~~
2000

(Serializable)
(No concurrent
Execution Problem)

* when the transaction is non-serializable then only
WR Problem, RW Problem, ww Problem are reasons
for non serializable inconsistency.

* Some times ww Problem is LOST UPDATE PROBLEM

⇒ LOST UPDATE PROBLEM:-

T1	T2
$R(A)$	
$w(A)$	
Rollback	
(Failed)	

A = ~~0, 10, 20~~ 0

$w(A)$ ← (lost update problem because of
T1 Rollback)

* Even if the schedule is serializable, lost Update Problem is still possible.

26/08/2011

Problem becoz of concurrent execution:-

⇒ if the schedule is non-serializable then RW, WR, WW Problems may exist.

⇒ if the schedule is serializable then schedule free from RW, WR, WW Problems. But lost Update Problem may be possible.

Serializable Schedule:-

⇒ Schedule which satisfies isolation property.

⇒ Concurrent Execution b/w two or more transactions should be equal to any serial schedule.

Classification of the Schedule:-

Two classification are possible. Based on:-

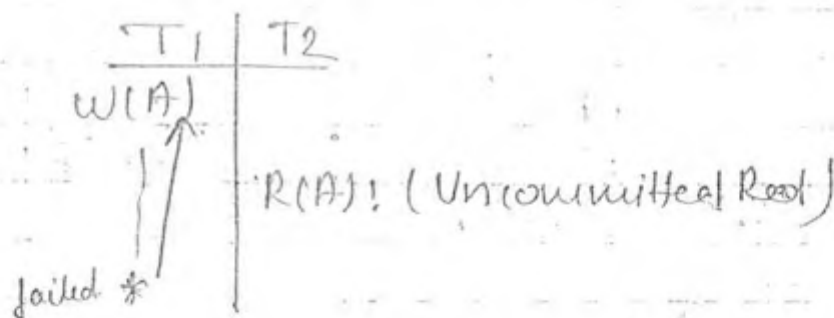
* Recoverability

* Serializability

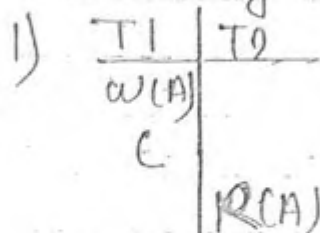
* Recoverability:-

If Trans T2 read the data item A that is updated by Uncommitted Trans T1

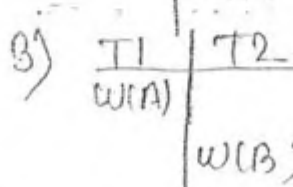
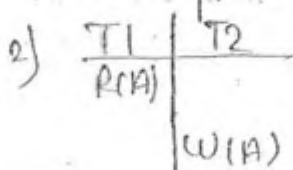
$\Rightarrow$ (T2 depending on T1 result)



Which of these are depending on other



NO dependency
b/w Transaction



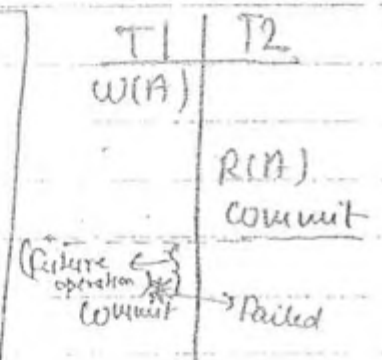
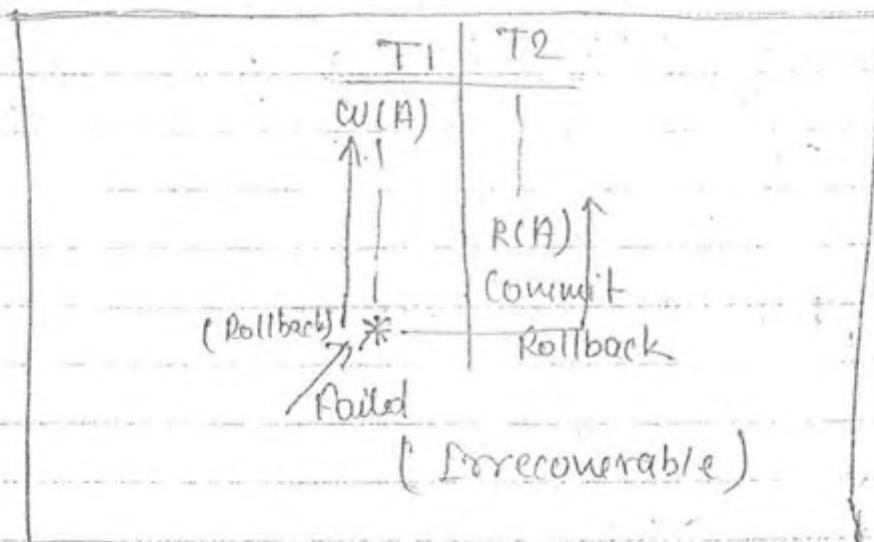
(If two transaction updated simultaneously that does not mean that there is a dependency between them)

T1	T2
w(A)	
	R(A)
	w(B)
R(B)	

Dependency exist

II Forerecoverable Schedule:-

Rollbacking of committed transaction.



T1	T2
w(A)	
	R(A)
	Commit
Commit/Rollback	

Trans-F Transaction T2 Reads the data item A updated by some other Transaction T1 if commit operation of T2 before the Commit/Rollback of T1 then schedule is irrecoverable.

[2] Reasonable schedule:-

Transaction.

Rollbacking of only uncommi

T1	T2	T1	T2	T1	T2
w(A)		w(A)		w(A)	
	R(A)		R(A)		R(A)
		Rollback		Rollback	
Commit/Rollback		Commit		C/R	
	Commit		Commit		Commit

Recoverable Recoverable

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 irrecoverable.

T1	T2
W(A)	R(A) $\Rightarrow$ It is recoverable
R(B)	W(B) $\Rightarrow$ It is not recoverable because it not satisfy the recoverable condition
Commit	Commit

(Irrecoverable)

T1	T2
R(A)	
W(A)	R(A)
	W(B)
R(B)	
W(B)	
C1	C2

(Dependency is present)
 Recoverable but not Serializable (WR Problem)

T1	T2
W(A)	
	W(A)
W	W(B)
W(B)	
C1	C2

(NO Dependency)
 Recoverable schedule but not Serializable schedule
 (WW Problem Exist)

T1	T2
R(A)	
	R(A)
	W(A)
	Commit
W(A)	
Commit	

No Dependency So it is Recoverable schedule but not Serializable (RW Problem)

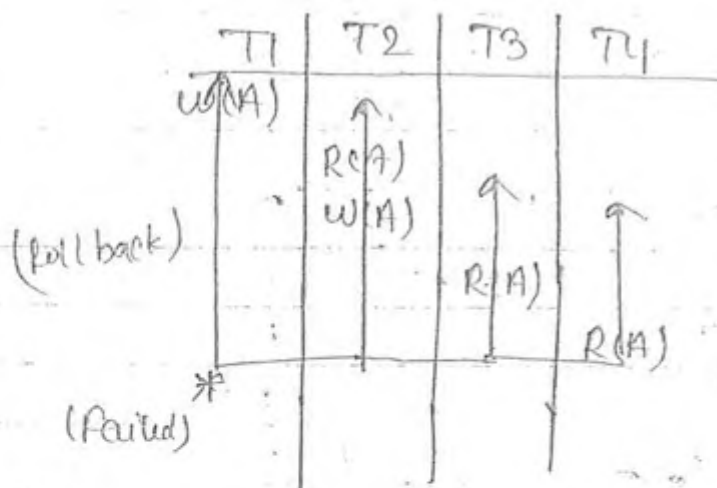
T1	T2
W(A)	W(A)
Rollback	Commit/Rollback

Recoverable

Schedule but lost Update Problem Exist.

⇒ Recoverable Schedule May not free from
 RW, WR, WW Problems and lost update Problem and Cascading Rollback Problem

Cascading Rollbacking:-



(Recoverable but Cascading Rollback)

Here T2 depending on T1, T3 depending on T2 & T4 depending on T3. If there is any failure occur in T1 before the commit then we roll back T1 along with we roll back all the dependending transaction must be roll backed.

⇒ it results wastage of CPU execution time and I/O operations

[8] Cascadless Rollbacking Recoverable Schedule:-

To remove cascading rollbacking problem we need the cascadless rollbacking i.e. no transaction is depending on other transaction.

T1	T2
w(A)	
c/R	
	R(A)

{ Uncommitted Read not allowed
In cascadless Rollbacking
Recoverable Schedule }

T1	T2
w(A)	
	w(A)
	w(B)
w(B)	
c1	
	c2

(WW Problem)
Is exist

T1	T2
R(A)	
	R(A)
	w(A)
w(A)	
c1	
	c2

(RW Problem)
is exist

T1	T2
w(A)	
	R(B)
	R(C)
	w(C)
w(C)	
c1	
	c2

⇒ WR Problem not possible here and cascading Rollback are not possible but still WW, RW are possible. Lost update problem is still possible.

[4] Strict Recoverable Schedule:-

To remove the lost update problem Strict Recoverable schedule is come, in this schedule we avoid simultaneously writes of any two transactions. Due to this WW problem also recoverable removed.

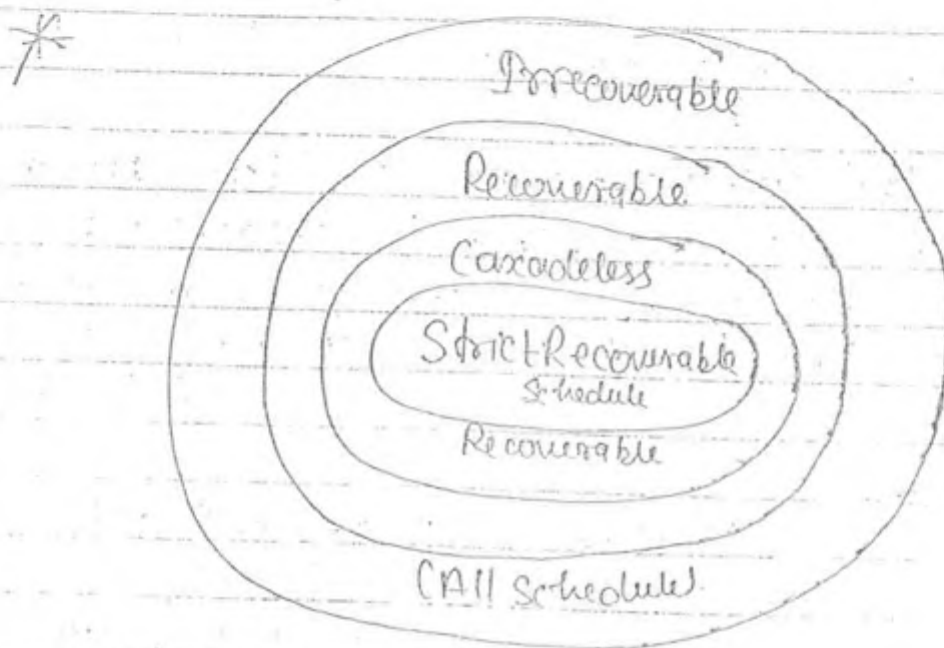
T1	T2
w(A) C/R	R(A)/w(A)

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
<u>R(A)</u>	R(A) <u>w(A)</u> C2
w(A) C1	

This schedule satisfied Strict Recoverable condition but non-serializable schedule due to RW problem.

⇒ Strict Recoverable Problem is free from ww , wR , Problem and also last update, cascading Rollback problem.
 But RW problem ^{may} also exist here.



* $\begin{matrix} \text{operation} \\ \downarrow \\ w_1(A) w_2(A) w_2(B) w_1(B) c_1 c_2 \end{matrix}$ $\begin{matrix} \text{data item} \\ \downarrow \\ \end{matrix}$
 $\downarrow$
 Transactions No

Que) SI: $s_1(x) s_2(z) s_1(z) s_3(z) s_3(y) w_1(x) w_3(y) s_2(y) w_2(z) w_3(y) c_1 c_2 c_3$
 Check which higher recoverability is present here.

T_1	T_2	T_3
$s_1(x)$	$s_2(z)$	
$s_1(z)$		$s_3(z)$
		$s_3(y)$
$w_1(x)$		
	$s_2(y)$	$w_3(y)$ $\rightarrow$ depending exist
	$w_2(z)$	
	$w_2(y)$	
c_1		
	c_2	c_3

$\Rightarrow$ It is irrecoverable.

Ques) $w_1(x) w_3(y) (s_3 s_2 y) w_2(z) w_2(y) c_1 c_2$

$\Rightarrow$ Recoverable $\checkmark$
 Causal $\checkmark$
 Strict Recoverable $\checkmark$

Ques) S21 - $s_1(x) s_2(z) s_3(x) s_4(z) s_2(y) s_3(y) w_1(x) c_1 w_2(z) w_3(y) w_2(y) c_3 c_2$

$w_1(x)$	
c_1	
$w_2(z)$	$w_3(y)$
$w_2(y)$	c_3
c_2	

Causal but not strict

Ques 3: $S_3(x) S_1(x) W_3(x) S_2(x) W_1(y) S_2(y) W_2(y) C_3 C_1 C_2$

RECOVERABLE ✓

Deadlock X

S4: $S_1(x) W_1(x) W_2(x) S_2(x) S_1(y) W_2(y) C_1 C_2$

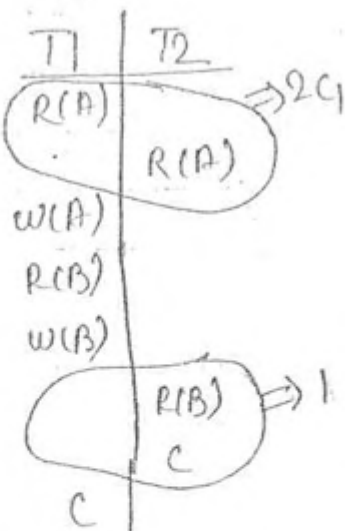
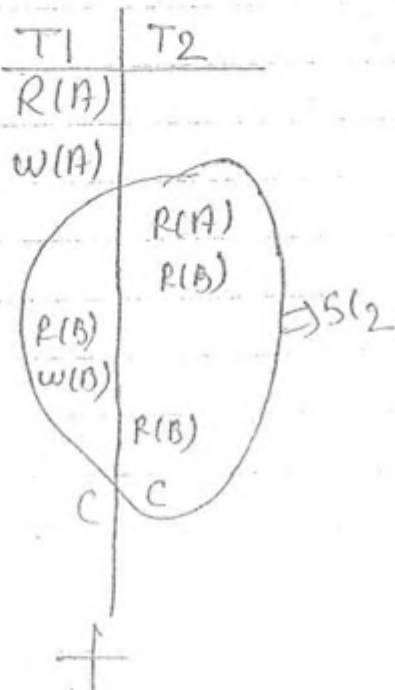
T1	T2	
$R_1(x)$		No dependency
$W_1(x)$		
	$W_2(x)$	REC ✓
	$S_2(x)$	CAS ✓
$S_1(y)$		Strict X
	$W_2(y)$	
C_1		
	C_2	

Ques) T1 operations is	T1	T2 operations is	T2
	$R_1(A)$		$R_2(A)$
	$W_1(A)$		$S_2(B)$
	$R_1(B)$		C_2
	$W_1(B)$		
	C_1		

$$\text{Total no. of schedules} = \frac{8!}{5!3!} = \frac{8 \times 7 \times 6 \times 5!}{5! \times 3!} = \frac{8 \times 7 \times 6}{3 \times 2 \times 1} = 56$$

No. of recoverable schedule:-

Example 1:-



= All schedule - Irrecoverable schedule

$$= 100 - 56 = 44$$

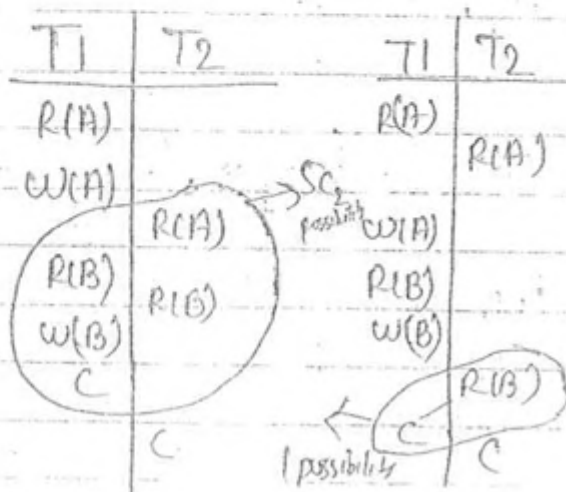
$$= 0C_5 - 5C_2 - (2C_1 * 1)$$

$$46 - 2$$

$$= 44$$

Alg Concaelless Reconesable Schedule:-

= (Reconesable Schedule) - (Concaelless Roll back)



⇒ 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 1:-

	T1	T2	T3
w(A)			
		w(A)	
			w(A)

A = 10
20
30

(S1) A1T3

T1	T2	T3
w(A)	w(A)	
		w(A)

A = 20
10
30

(S2) A1T3

S1 and S2 are ~~not~~ logically equivalent because S2 resulted after swapping conflict pairs in S1.

So it is Not conflict equivalent ~~but~~ S1 & S2 are equivalent Schedule.

Que

T1	T2
R(A)	
w(A)	
	R(A)
	R(B)
R(B)	
w(B)	

S1

T1	T2
R(A)	
w(A)	
	R(A)
R(B)	
	R(B)
w(B)	

S2

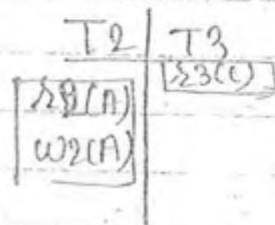
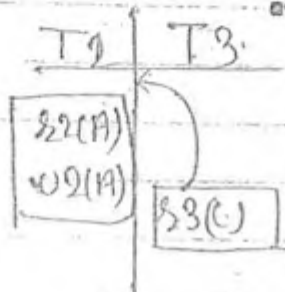
S1 is logically equivalent to S2 because we are swapping two consecutive non-conflict pairs in the schedule.

Ques 2008

S1: $\boxed{r_2(A)} \boxed{w_2(A)} \boxed{r_3(C)} \boxed{w_2(B)} \boxed{w_3(A)} \boxed{w_3(C)} \boxed{r_1(A)} \boxed{r_1(B)} \boxed{w_1(A)} \boxed{w_1(B)}$
 S2: $\boxed{r_3(C)} \boxed{r_2(A)} \boxed{w_2(A)} \boxed{w_2(B)} \boxed{w_3(A)} \boxed{r_1(A)} \boxed{r_1(B)} \boxed{w_1(A)} \boxed{w_1(B)} \boxed{w_3(C)}$

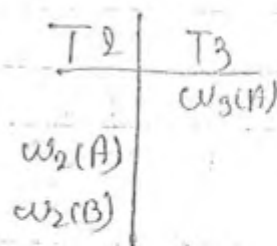
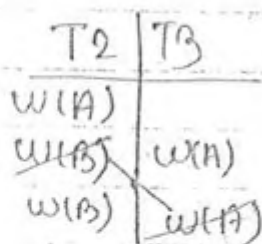
S3: $r_2(A) r_3(C) w_3(A) w_2(A) w_2(B) w_3(C) r_1(A) r_1(B) w_1(A) w_1(B)$

Which one of them are conflict equivalent schedules



$\Rightarrow$ S1 & S2 are conflict equivalent

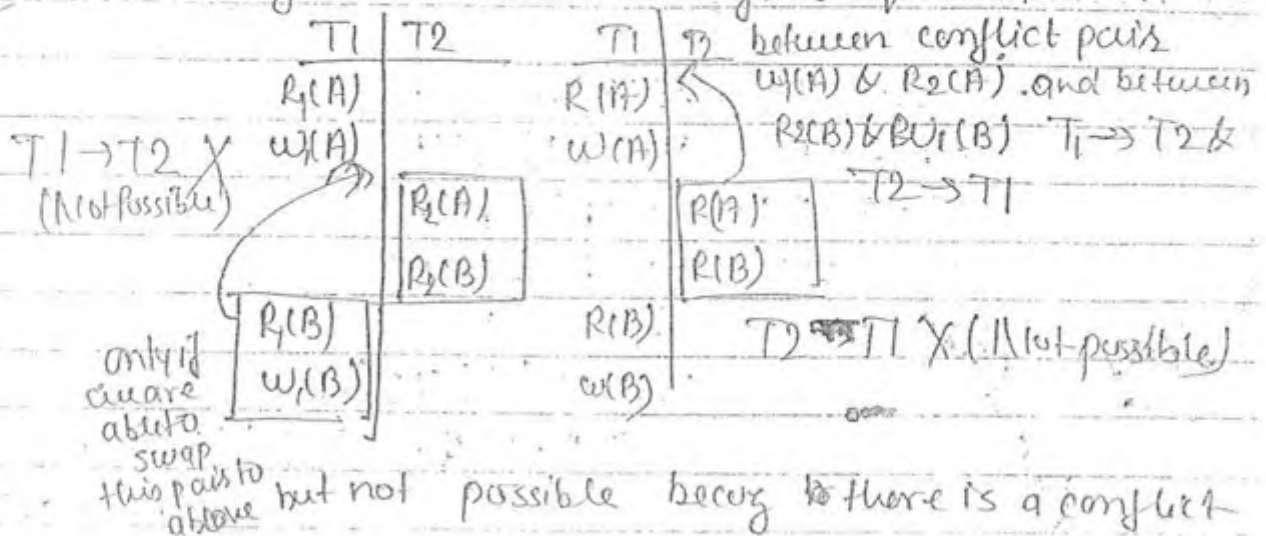
S2: $\boxed{r_3(C)} \boxed{r_2(A)} \boxed{w_2(A)} \boxed{w_2(B)} \boxed{w_3(A)} \boxed{r_1(A)} \boxed{r_1(B)} \boxed{w_1(A)} \boxed{w_1(B)} \boxed{w_3(C)}$
 $\boxed{r_2(A)} \boxed{r_3(C)} \boxed{w_3(A)} \boxed{w_2(A)} \boxed{w_2(B)} \boxed{w_3(C)} \boxed{r_1(A)} \boxed{r_1(B)} \boxed{w_1(A)} \boxed{w_1(B)}$
 (Conflict pair)



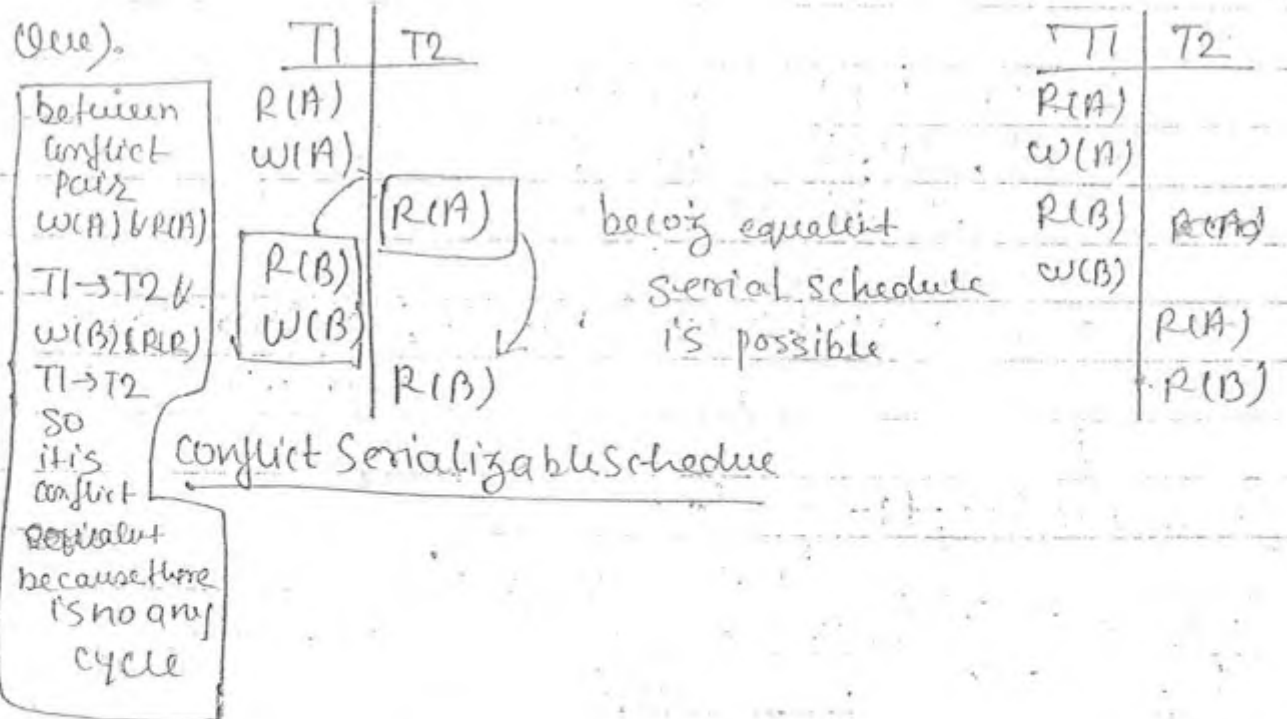
Conflict Serializable Schedule

Schedule is said to be Conflict Serializable schedule only if there should be conflict Equivalent Serial Schedule.

Ques) Check the given schedule is conflict equivalent:-



(T_1 followed by T_2) is $T_1 \rightarrow T_2$ X not possible
 (T_2 followed by T_1) is also not possible i.e. $T_2 \rightarrow T_1$ X.



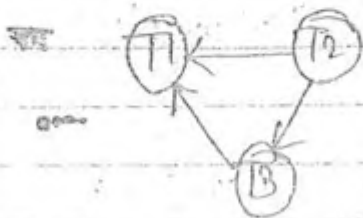
Ques) $R_2(A) W_2(A) R_3(C) W_2(B) W_3(A) W_3(C) R_1(A) R_1(B) W_1(A) W_1(B)$

$T_2 \rightarrow T_3$ $R_2(A) \& W_3(A)$

$T_2 \rightarrow T_1$ $W_2(A) \& R_1(A)$

$T_2 \rightarrow T_1$ $W_2(B) \& R_1(B)$

$T_3 \rightarrow T_1$ $W_3(A) \& R_1(A)$



$T_2 \rightarrow (T_1 T_3)$

$T_2 \rightarrow T_3 \rightarrow T_1$ (conflict)

Serializable
Schedule

Ques) $R_2(A) R_3(C) W_3(A) W_2(A) W_2(B) W_3(C) W_1(A) R_1(B) W_1(A) W_1(B)$

$R_2(A) \& W_3(A)$ $2 \rightarrow 3$

$W_3(A) \& R_1(A)$ $3 \rightarrow 1$

$W_3(A) \& W_2(A)$ $3 \rightarrow 2$

$2 \rightarrow 3 \& 3 \rightarrow 2$ (non-conflict Serializable
Schedule)

Precedence Graphs:-

Graph $G = (V, E)$

Vertices (V): Transaction of the schedule.

Edges (E): $T_i \rightarrow T_j$

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 preceded
then T_j .

if $T_i: R(A) \quad T_j: W(A)$

$T_i: W(A) \quad T_j: R(A)$

$T_i: W(A) \quad T_j: W(A)$

Testing Condition

if Precedence graph is Cyclic $\Rightarrow$ Non conflict
Serializabl

if Precedence graph is Acyclic $\Rightarrow$ Conflict
Serializabl

$\Rightarrow$ Equivalent Serial Schedule based on topological (sequence) order of acyclic precedence graph.

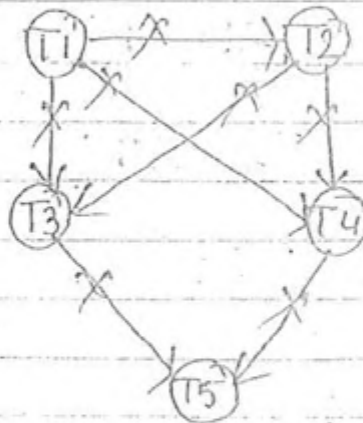
T

27/08/11

Topological Order

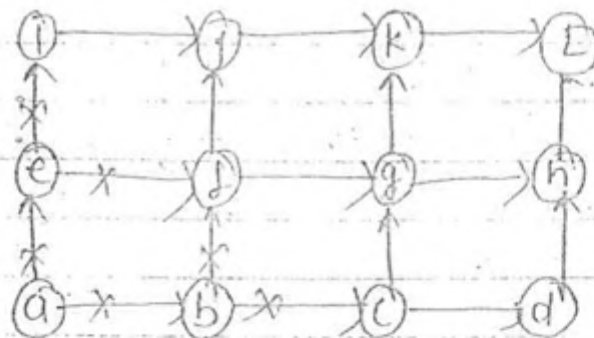
[1] Visit vertex (v) with indegree '0' and delete visited vertex (v) along with outdegree edges of v.

[2] Repeat [1] until graph becomes empty.



These are the two topological sequences of given graph.

$\left\{ \begin{array}{l} T1 : T2 : T3 : T4 : T5 \\ T1 : T2 : T4 : T3 : T5 \end{array} \right.$



a : b : d : c : d : h :

a : b : c :

a : e : d :

a : e : i :

a : b : f : c : g : k :

a : b : f : c : g : h :

Based on Serializability:-

- 1) Conflict Serializable:-
- 2) View Serializable:-

Conflict Serializable Schedule:-

Non Conflict Pairs:-

- After swapping of two operation of two different transaction, then there is no any ^{due to the resultant schedule} change in old schedule. Then that operation is no conflict pair.

[A] $T_1: R(A) \quad T_2: R(A)$: Non conflict Pair

$$\left(\begin{array}{c|c} T_1 & T_2 \\ \hline R(A) & R(A) \end{array} \right) = \left(\begin{array}{c|c} T_1 & T_2 \\ \hline R(A) & R(A) \end{array} \right)$$

$S_1 \qquad S_2 \quad \text{becoz} \Rightarrow S_1 = S_2$

[B] $T_1: R(A) \quad T_2: W(A)$: Conflict Pair

$$\left(\begin{array}{c|c} T_1 & T_2 \\ \hline R(A) & W(A) \end{array} \right)$$

S_1

$$\left(\begin{array}{c|c} T_1 & T_2 \\ \hline R(A) & W(A) \end{array} \right)$$

$S_2 \Rightarrow S_1 \neq S_2$

[c] $T_1: w(A) \quad T_2: R(A) \quad \therefore$ Conflict Pair

T_1	T_2
$w(A)$	$R(A)$

[d] $T_1: w(A) \quad T_2: w(A) \quad \therefore$ Conflict Pair

T_1	T_2
$w(A)$	$w(A)$

T_1	T_2
$w(A)$	$w(A)$

Final updation A is done
by T_2

Final updation of A is
done by T_1

$\Rightarrow S_1 \neq S_2$

[e] $T_1: R(A)/w(A) \quad T_2: R(B)/w(B) \quad \therefore$ Non Conflict

Conflict Pairs:-

Pair of operations said to be conflict only if

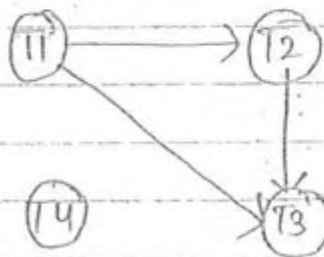
- 1) At least one write operation
- and 2) on same data item
- and 3) b/w different transaction

Conflict Equivalent Schedule:-

S_1 & S_2 said to be conflict equivalent only if S_2 resulted after swapping consecutive non conflict pairs from S_1 .

Ques) Identify Conflict Equivalent Serial Schedule:-

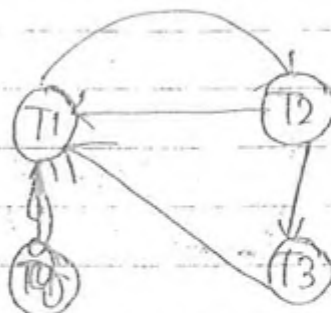
[1] $R_1(A) R_2(A) R_3(A) R_4(A) W_1(B) W_2(B) W_3(B)$



$T_4 : T_1 : T_2 : T_3$
 $T_1 : T_4 : T_2 : T_3$
 $T_1 : T_2 : T_4 : T_3$
 $T_1 : T_2 : T_3 : T_4$

} Conflict
Equivalent
Serial Schedule

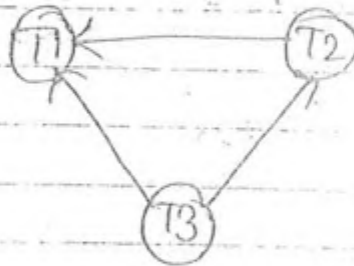
[2] $R_2(z) R_2(y) W_2(y) R_3(y) R_3(z) R_1(x) W_1(x) W_3(y) R_2(x)$
 $R_1(y) W_1(y) W_2(x)$



Non Conflict
Equivalent Schedule

[3] $R_1(A) R_2(A) R_3(B) W_1(A) R_3(C) R_2(B) W_2(B) W_1(C)$

$R_2(A) \rightarrow W_1(A)$
 $R_3(B) \rightarrow W_2(B)$
 $R_3(C) \rightarrow W_1(C)$



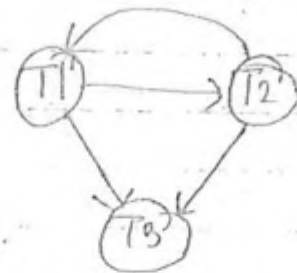
Conflict Serializable Schedule

$T3 : T2 : T1$

* If conflict schedule is serializable then only we can say that it is conflict serializable schedule.

Ques)

	T1	T2	T3
SV	R(A)		
		W(A)	
	W(A)		
		S1	W(A)



Non conflict Serializable schedule:-
If we execute this schedule in serial way:-

S2!

	T1	T2	T3
	R(A)		
	W(A)		
		W(A)	
			W(A)
		S2	

$S1 = S2$ i.e.
 schedule are equal but
 not conflict equivalent.

S_2 is equivalent serial schedule of S_1 so, it is equivalent serial schedule $\Rightarrow$ (Serializable Schedule).

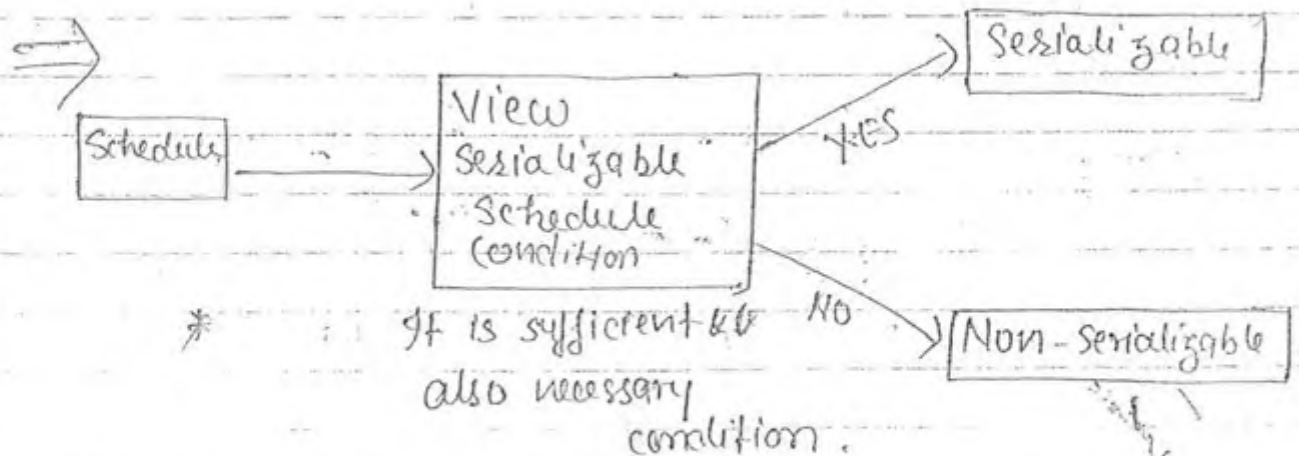
$\Rightarrow$ if (Acyclic Precedence graph)

- { conflict serializable (Serializable)
- } // It is sufficient to say that schedule is serializable

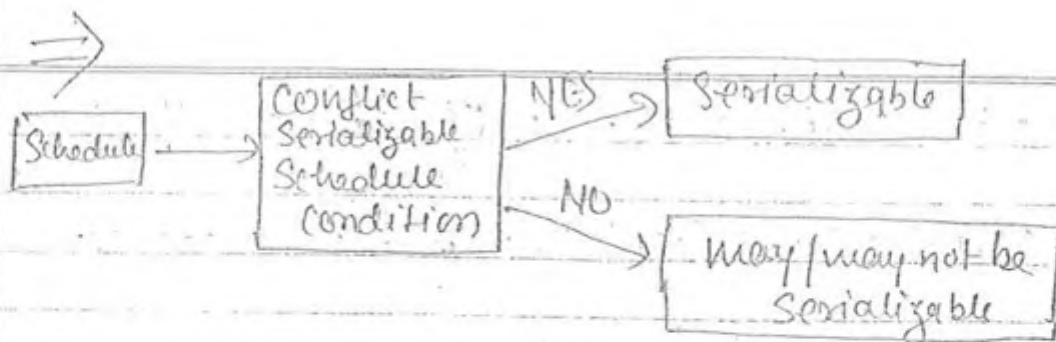
else

- { Not conflict Serializable Schedule
- } may / may not be serializable
- } // Not Necessary Condition to say non serializable

* It is only sufficient but not necessary.



Exponential Time complexity
(NP-Complete problem)



⇒ Polynomial Time Complexity $O(n^2)$

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) $\swarrow$ T_1
 $R(A)$

(S1)

(Initial DB) $\swarrow$ T_1'
 $R(A)$

(S2)

Example:-

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

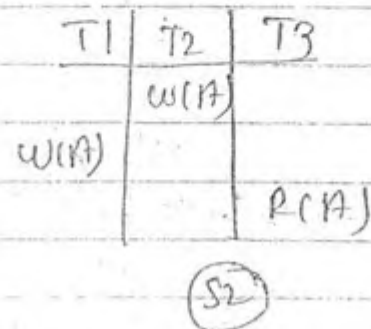
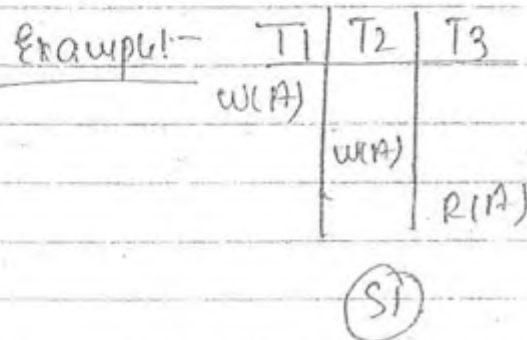
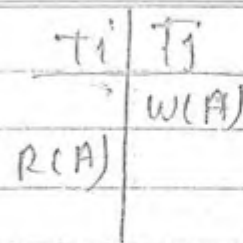
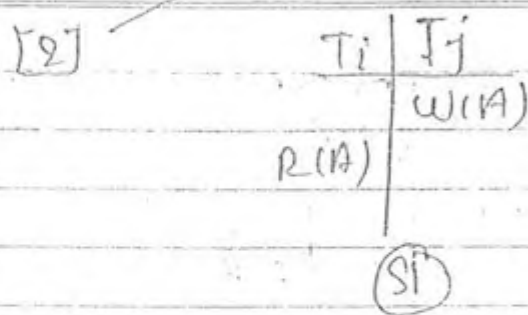
(S1)

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

(S2)

S_1 & S_2 are not equivalent becoz in S_1 $R_1(A)$ & $R_2(A)$ are initially read data item from DataBase but in S_2 $R_2(A)$ read the data item updated by $T_3(A)$. So it's not equivalent $S_1 \neq S_2$

Write, Read Sequence should be same in the S1 & S2 schedule

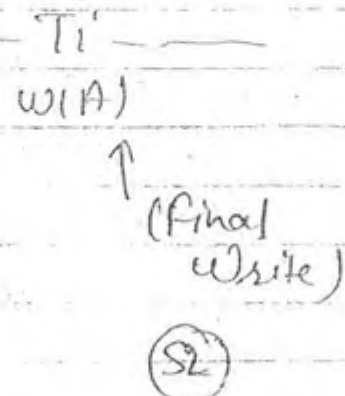
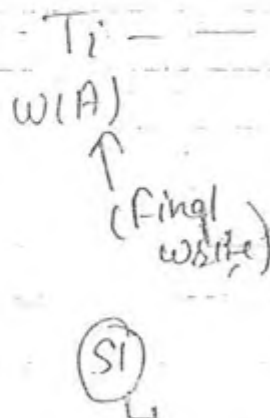


$\Rightarrow S1 \neq S2$ becoz here ~~R3(A)~~ in S1 $R3(A)$ read the data item updated by $W2(A)$

But in S2 $R3(A)$ read the data item updated by $W1(A)$.

So, $S1 \neq S2$ because write & Read sequence should not same.

[3] Final Write. should be same.



Example:-

T1	T2	T3
w(A)		
	w(A)	
		R(A)

(S1)

T1	T2	T3
	w(A)	
w(A)	.	
		R(A)

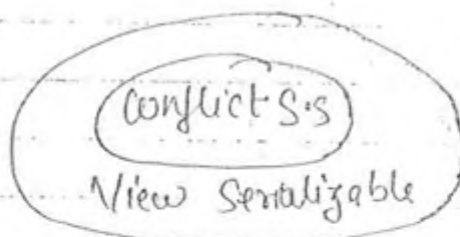
(S2)

$S1 \neq S2$ 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	T1	T2	T3
R(A)			R(A)		
	W(A)		W(A)		
W(A)				W(A)	
		W(A)			W(A)

(S1) Not SS $\Leftrightarrow$ 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)	
			W(A)

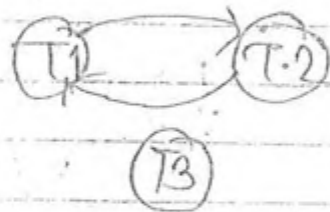
T1:T2:T3:T4
(Conflict SS)

There is 6 different serial schedules are possible for it, because only we have to save the final updation of data for transaction. We can change the order of initial three transaction in any order. Because they do not change the serial of any serial.

(T1 T2 T3 T4)

T1 T3 T2 T4
 T2 T3 T1 T4
 T2 T1 T3 T4
 T3 T1 T2 T4
 T3 T2 T1 T4

(Ques) $S: R_2(B) R_2(A) W_1(A) R_3(A) W_1(B) W_2(B) W_3(B)$
 Identify all view equivalent Serial Schedule.



i.e. Cycle is there so it is

Non conflict serializable schedule

For checking View Serializable Schedule:-

[1] Final Updation:-

A: $\bar{T}_1$

B: $T_1 \quad T_2 \quad \bar{T}_3 \quad (T_1 \quad T_2) \rightarrow T_3$

$\Rightarrow$ 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: T_2

B: T_2

Write

$T_1 \Rightarrow T_2 \rightarrow T_1$

$T_1, T_2, T_3 \Rightarrow T_2 \rightarrow T_1$

$T_2 \rightarrow T_2$

$T_2 \rightarrow T_3$

[3] WR Sequence:-

$W_1(A) : R_3(A) \Rightarrow T_1 \rightarrow T_3$

Model -

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 conclusion i.e.

(T1 T2) → T3

T2 → T1 T3

T1 → T3

T2 → T1 → T3

are one by the view equivalent
serializable schedule.

Ques) S: $r_2(A) r_1(A) w_1(C) r_3(C) w_1(B) r_4(B) w_3(A)$
 $r_4(C) w_2(B) r_2(B) w_4(A) w_4(B)$

For checking View Serializable schedule:-

[1] Final Update:-

A: T3 (T1)

T3 → T4

B: T1 T2 (T4) (T1 T2) → T4

C: (T1)

[2] Initial Read:-

Initial Read Write

A: T1 T2

T3 T4

⇒ ~~S: $r_2(A) r_1(A) w_1(C) r_3(C) w_1(B) r_4(B) w_3(A)$~~

B: ~~---~~

T1 T2 T4

⇒ ~~$r_4(C) w_2(B) r_2(B) w_4(A) w_4(B)$~~

C: ~~---~~

T1

⇒ ~~$r_2(A) r_1(A) w_1(C) r_3(C) w_1(B) r_4(B) w_3(A)$~~

	Initial Read	Write
A	T1 T2	T3 T4
B	---	T1 T2 T4
C	---	T1

[3] WR Sequences-

- 1) $w_1(C) \quad r_3(C) \quad T_1 \rightarrow T_3$
 - 2) $w_1(B) \quad r_4(B) \quad T_1 \rightarrow T_4$
 - 3) $w_1(C) \quad r_4(C) \quad T_1 \rightarrow T_4$
 - 4) $w_2(B) \quad r_2(B) \quad T_2 \rightarrow T_2$
- } $T_1 \rightarrow (T_3 T_4)$

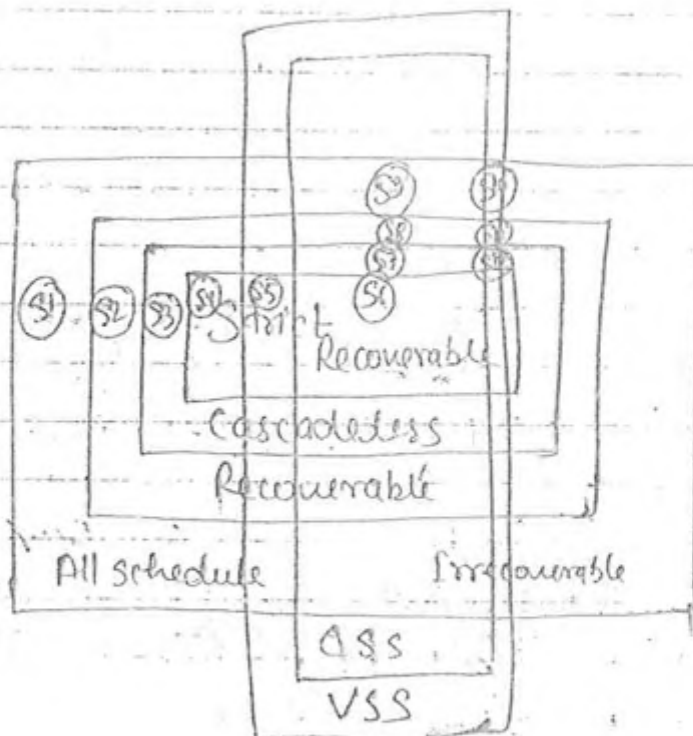
[1] $(T_1 T_2 T_3) \rightarrow T_4$ } $(T_1 T_2) \rightarrow T_3 \rightarrow T_4$

[2] $(T_1 T_2) \rightarrow (T_3 T_4)$

[3] $T_1 \rightarrow (T_3 T_4)$

$T_2 \rightarrow T_1 \rightarrow T_3 \rightarrow T_4$

Two view equivalent
Schedules are
possible.



Schedule is Irrecoverable & not Serializable:-

T1	T2
R(A)	
W(A)	
	R(A)
	R(B)
R(B)	
W(B)	
	C2
C1	

(SI)

Schedule is Recoverable but not Consistent

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
if the schedule is failed it rollback and updation done by T2 is removed	$w(A)$ $w(B)$ *	$w(A)$ $w(B)$

(NO WW Problem
But Lost Update Problem may possible)

⇒ Strict Recoverable schedule (lost update problem not exist)

* Types of Protocol -

- a) locking protocol.
- b) 2 Phase locking protocol.

[A] Locking Protocol:-

lock:- lock is a variable which is used to identify the status of the data-item.

Legal Transaction:-

Transaction is said to be legal transaction if transaction should lock the data item before usage of data item & release data item after usage.

=> Every transaction in the schedule is legal transaction

Ex:-
L(A)
R(A)
W(A)
V(A)

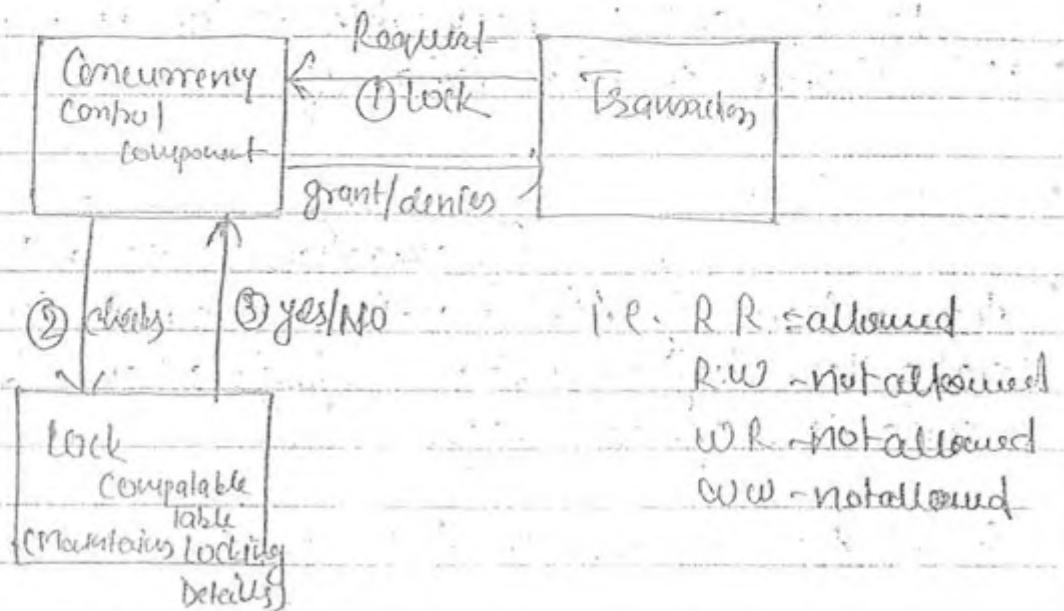
* 'Lock' work as same like "critical section" in Operating system. That in Operating system critical section is used for "shared Resources".

Critical Section - It is a shared resource for O.S.
For this we use Semaphore : Mutex that also share resource.

Process
{
if (mutex = 0)
 mutex = 1
 CS
 mutex = 0
}

if (mutex = 0)
{
}
else
{ (P2 leave to wait) }
until mutex become 0 (means

Components of legal transaction of 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]

Transaction locked the data item A in Shared mode means that transaction can Read data item A.

A

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)!

Transition T locked the data item A in Exclusive mode means that T can either Read/Write the Data item A.

T
X(A)
R(A)
W(A)

Lock Compatible Table!

(Hold T_i)

X(A) ← Requesting T_j

	S	X	⇒ Hold T_j
S	YES	NO	
X	NO	NO	

Requested

Example!

T_1	T_2
X(A)	
R(A)	
W(A)	
U(A)	
	S(B)
	R(B)
	U(A)
	S(B)
	R(B)
	U(B)
X(B)	
R(B)	
W(B)	
U(B)	

Transaction is legal and also it follows the compatible table.

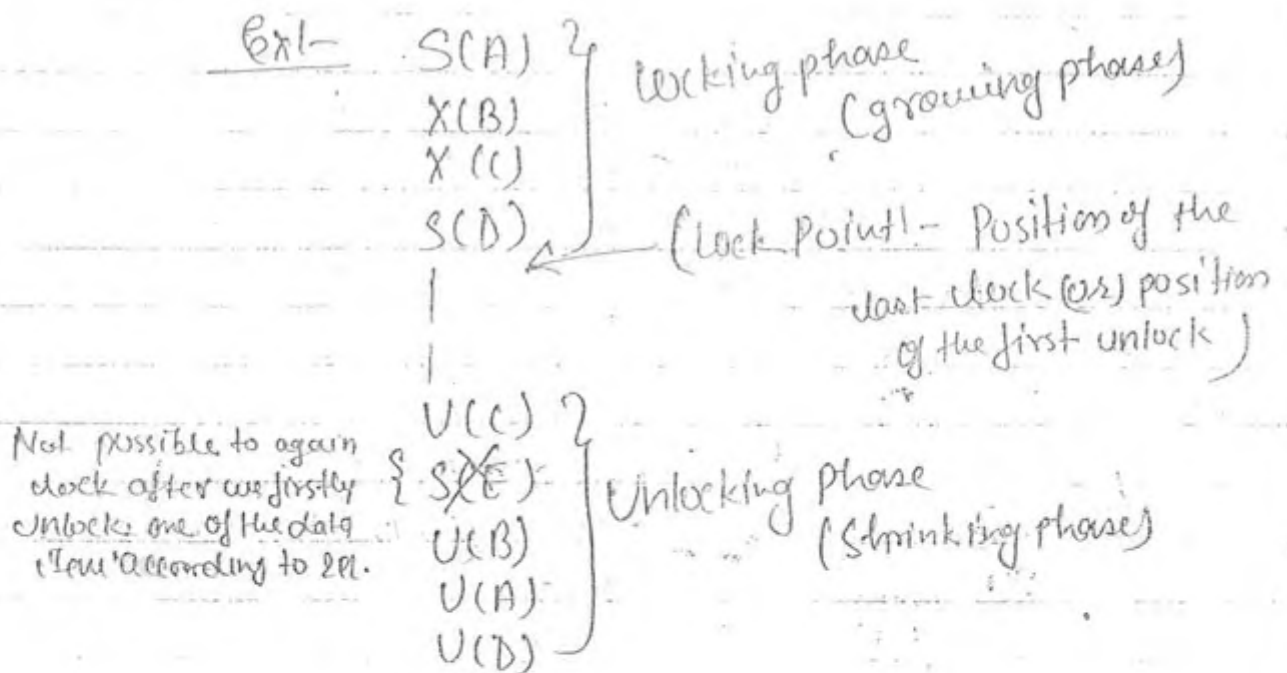
⇒ legal Trans
by compatibility satisfied b/w S/X locks.
but non-serializable schedule

⇒ (S/X may not free from Non Serializability)

i.e. locking protocol is not free from serializability
So put some restriction on the locking protocol
to remove the above problem. We can call
Such protocol as Two Phase locking Protocol

[B] 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(A)	
R(A)		R(A)	
W(A)		X(B)U(A)	S(A)
V(A)	(denied)		R(A)
	S(A)		S(B)
	R(A)		R(B)
	U(A)	R(B)	
	S(B)	W(B)	
	R(B)		
	V(B)		
denied X(B)			
R(B)			
W(B)			
V(B)			

Not possible to execute using 2PL.

⇒ 2PL! Always ensure conflict serializability. But reverse is not true.
(Not possible to execute any non-serializ schedule)

⇒ If it is 2PL then it is serializable. But reverse is not always true i.e. every serializable is execute 2PL.

⇒ Equivalent Serial Schedule based on the position of clock point.

	T1	T2	T3
X(A)	X(A)		
w(A)	R(A)		
(lock part) X(B) U(A)	S(A) R(A)	<u>X(A)</u> W(A)	This excludes lock is not allowed!
W(B) V(B)	X(B) ← lock point W(B) V(B) V(A)	W(A)	
Allocated To			W(A)

$\Rightarrow$ Not possible using 2 pr.

$$T_1:T_2$$

T1	T2	T3
	X(A) R(A)	
X(B) R(B)		
X(A)	W(A) X(B) X(A)	X(A) → Not possible i.e. Exclusive locks not allowed R(A)
W(B)		W(A)
	R(B)	
	W(B)	

$$T_1 \rightarrow T_2 \rightarrow T_3$$

Conflict Serializable
Schedule but not
possible using 2PM.

$\Rightarrow$ Every conflict serializable schedule may not be possible to execute by using 2PL.

* $\Rightarrow$ All conflict serializable schedule may not possible to execute even by using lock upgrading of 2PL.

Que) 2PL Allowed to execute?

T1	T2	T3
S(A) R(A)		S(A) R(A)
S(B) R(B) U(A) U(B)	X(B) R(B) W(B)	
	S(X) R(C)	X(A) W(A) U(X)
		W(C)

$\Rightarrow$ Not allowed in 2PL.

lock upgrade technique using 2PL allowed or Not?

T1	T2	T3
S(A) R(A)		R(C)
(lock point) $\Rightarrow$ S(B) R(B) U(A) U(B)	S(B) R(B) X(B) W(B)	
	(lock point) $\Rightarrow$ S(C) R(C) U(C) U(B)	X(A) W(A)
		(lock point) $\Rightarrow$ X(C) U(C) U(A)

$\Rightarrow$ allowed to execute using lock upgrade technique.

Equivalent serial schedule:

T1 $\rightarrow$ T2 $\rightarrow$ T3

2PL with S/X locking:-

Limitation of 2PL:-

[A] Basic 2PL:-

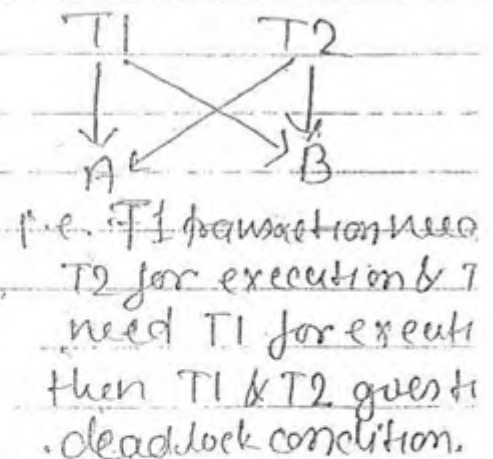
T1	T2
X(A)	
W(A)	
R(B) V(A)	S(A)
	R(A)
W(B)	
V(B)	S(B)
	R(B)
	V(B)
	Commit
Commit	

⇒ Serializable schedule & allowed to execute by 2PL.

but IRRECOVERABLE Schedule.

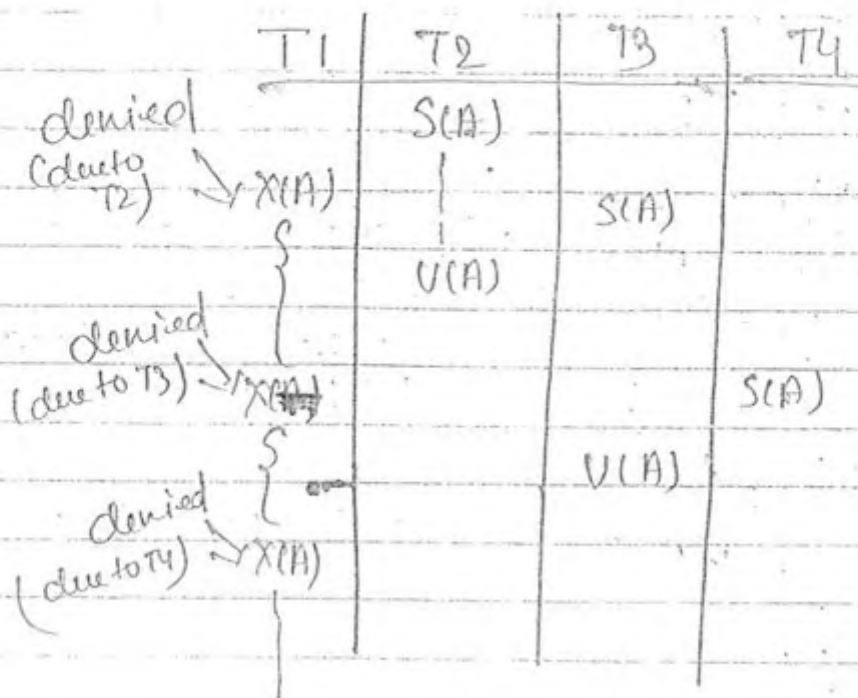
[B]

T1	T2
X(A)	
W(A)	
(Denied)	
X(B)	
W(B)	
waiting	
	X(B)
	W(B)
	X(A)
	W(A)
	waiting



* 2PL may results the deadlock also as may not free from Deadlock.

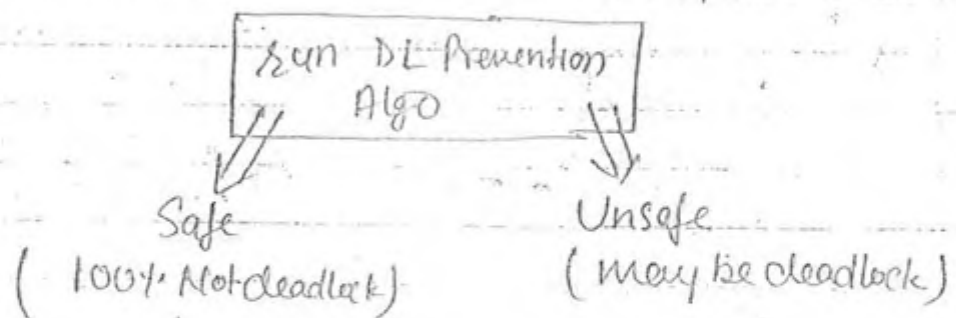
Starvation:- If only one process goes to infinite block of data item then starvation is occur. one transaction permanently block due to unavailability of resources but other transaction is executed this scenario is called starvation.



* May not free from starvation.

Deadlock Prevention:- not allow to occur the deadlock.
(OS) means not to get the deadlock

Process P1: requested R1

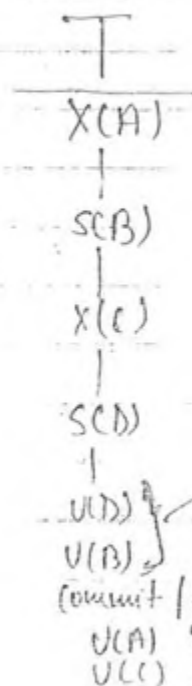


T1	T2	T1	T2
W(A)		W(A)	
	R(A)		R(A)
	C		Abort
R/C		C	
(Irrecoverable)		(Recoverable)	

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 ho until commit Rollback



we may unlock the shared lock before the commit or rollback

we always unlock the Exclusive lock after the commit or rollback of transaction.

Commit / Rollback

Example 1-

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 becoz of holding Exclusive locks until commit.
- * Becoz 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] Regressive 2PL Protocol:-

Basic 2PL + All Exclusive locks / shared lock should be hold until commit.

T1
S(A)
X(A)
|
commit
V(A)
V(A)

- * Difference between Strict 2PL & Regressive 2PL is that in strict 2PL only all exclusive lock should be hold until commit but in Regressive all exclusive lock as well as all strict lock also should be hold after commit.

- ⇒ Ensuring strict Recoverability
- ⇒ Ensuring Serializability
- ⇒ Only difference is that Equivalent Serial schedule based on order of commit.

T1	T2	T3
	C2	
C1		
		C3

⇒ T2, T1, T3

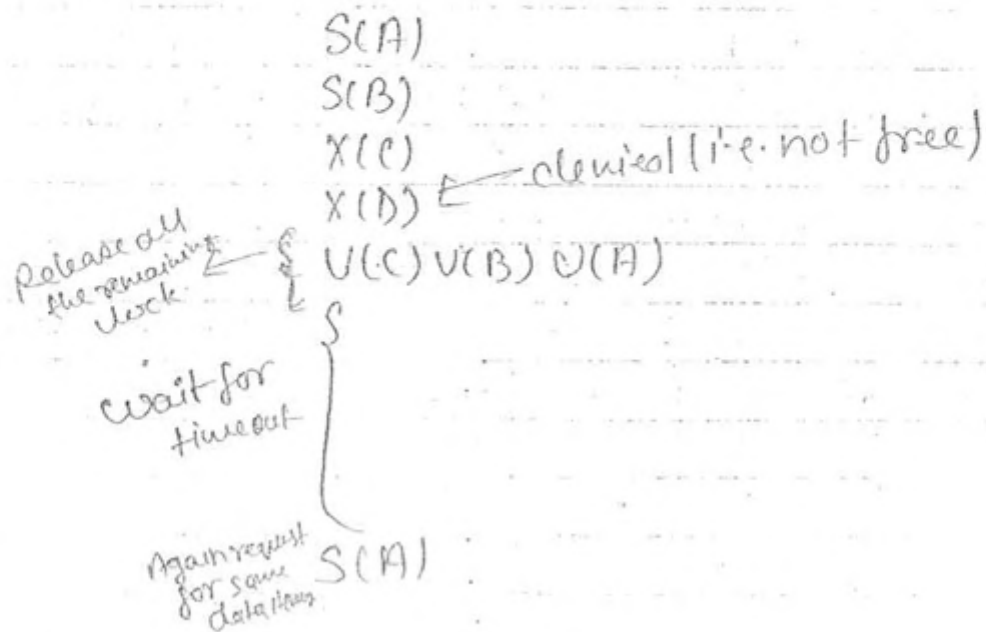
Conservative 2PL Protocol:-

- ⇒ This protocol is basically made for To Avoid Deadlock.
- ⇒ It says that lock the all data item that are required to execute by the transaction, before beginning of the execution of Transaction. If any data item is not free release the existing lock and repeat the same after time out

S(A)	S(B)
X(C)	X(D)

→ R(A)
R(B)
W(C)
W(D)

1) Some data item is not free -



Disadvantages of Conservative 2PL

- Starvation Possible
- Irrecoverability is possible.

Example

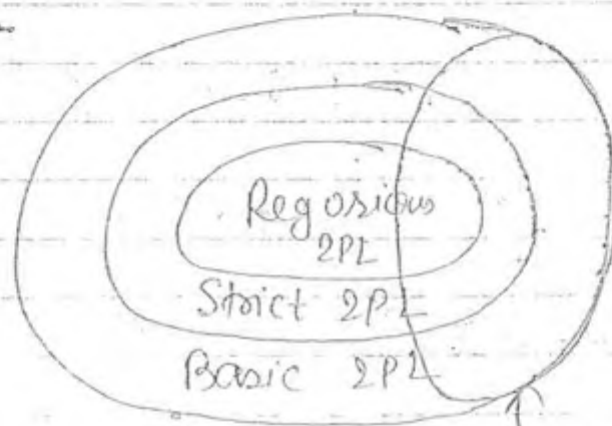
T1	T2
X(A)	
X(B)	
W(A)	
U(A)	S(A)
	R(A)
	U(A)
W(B)	
W(C)	
	Commit
Commit	

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 holded long before the requirement] becoz of this is less concurrency is occur, so throughput is less.

[d] Prediction of locks required by the transaction is too difficult.



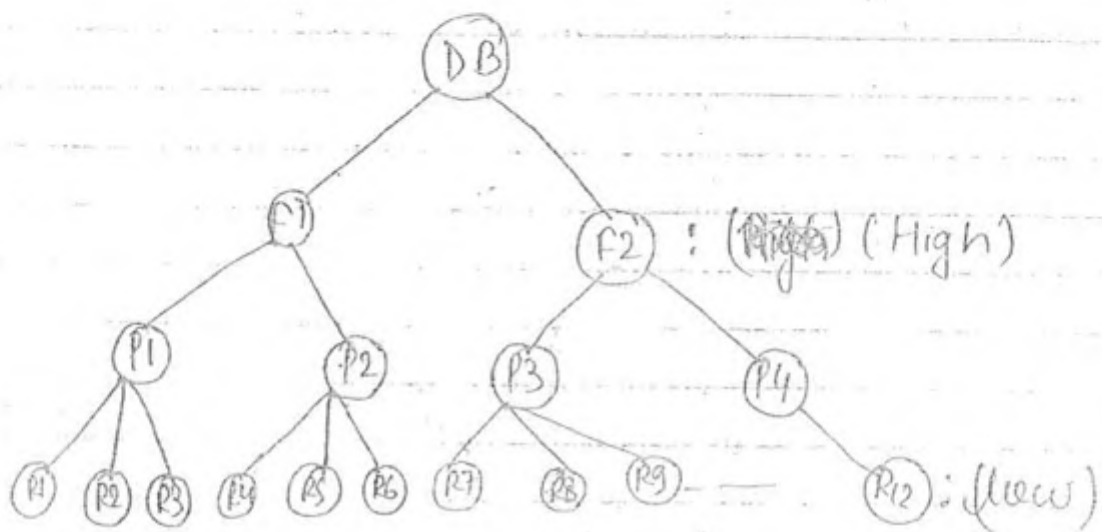
Ques)	T1	T2	T3
	S(A) R(A)		
	X(B) U(C)	S(C) R(C)	X(A) W(A)
	W(B) U(B)	S(A) R(A) U(A) U(C)	U(A) (cleared in strict 2PL)
	Commit	Commit	Commit

- a) Not in 2PL
- b) 2PL but not strict 2PL
- c) Strict 2PL but not Regorous
- d) Regorous 2PL

Multilevel Granularity Protocol! - (Tree Protocol!)

Granularity!

Granularity means lock in the DB.



There are two types of lock on the basis of Granularity

- 1] High level locking
- 2] Low level locking

Example! - If T_1 : Update $R_1, R_2, R_3, R_4, R_5, R_6$
 T_2 : update R_2 & R_{12}

[1] By using High level locking! - $*T_1: (L_1(F_1))$

Advantage! No. of locks are very less
 (lock compatible Table
 Easy to maintain)

$*T_2: (L_1(F_1) L_2(F_2))$

Disadvantage! - results less concurrency
 because Transition may
 hold Un-necessary resource

[2] low level locking:-

Disadvantage:-

* $T_1: \{l_1(R_1) l_1(R_2) l_1(R_3) \dots l_1(R_6)\}$ } complex to maintain
lock compatibility

Advantage

* $T_2: \{l_2(R_2) l_2(R_{12})\}$ } (More concurrency level)

[3] Multilevel granularity:- (Best possible lock depends on ^{Transit} ^{read} ^{require})
Depends on transaction type
Should allow to lock at any level.

* $T_1: l_1(F_1)$

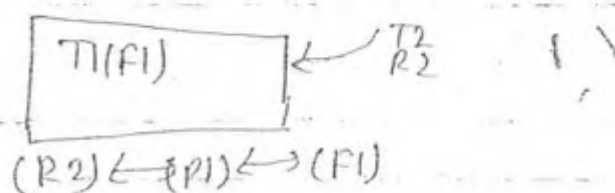
* $T_2: l_2(F_2) l_2(R_{12})$

* Firstly file is locked then after request for locking second?

Example:-

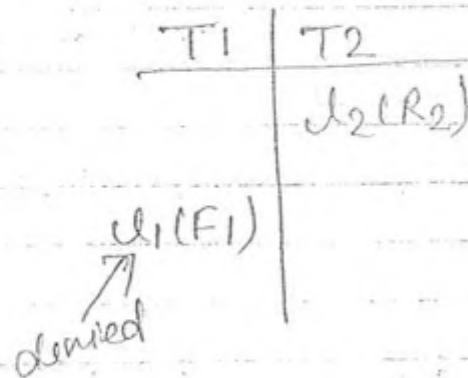
T_1	T_2
$l_1(F_1)$	$l_2(R_2) \leftarrow \text{denied}$

then how to check which file is locked
then how the concurrency control do for grant ordering the ^{permission}



* No. of comparison = height of Tree
= constant. (To grant or to deny the permission)

⇒ If firstly ^{record} record is locked & we request for locking the file in which lock record is belongs to -



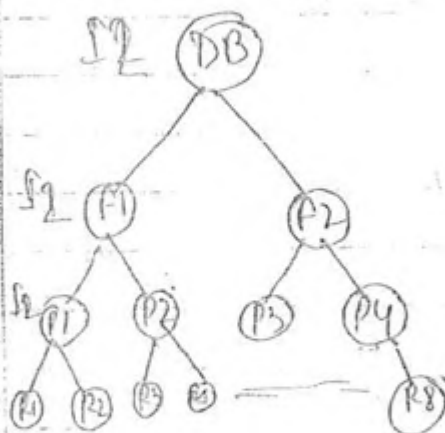
Difficulty comes for concurrency control to allow or grant the permission or deny the permission -

L₂(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 Intentional lock at every level i.e.



I₂(DB) I₂(F₁) I₂(P₁) L₂(R₂)

30/08/11

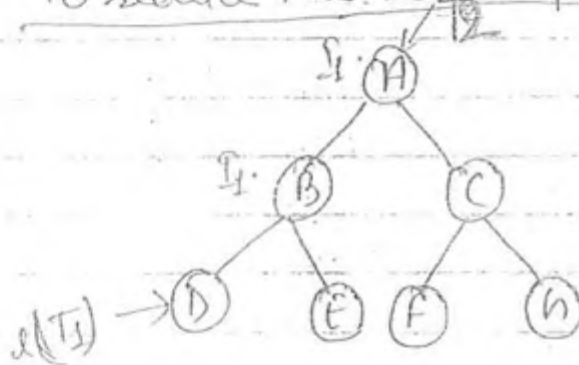
Tree Protocol:-

a) Multilevel Granularity Protocol:-

Intension lock:-

A Node N can be locked by Trans T in Intension mode means that trans T can allowed to request lock on Any descendent of N

To rectify the below problem we use Intension lock



T1	T2
$l_1(D)$	

$l_2(A)$ (exponential time)

becoz lock status requires to check Any descendents of A .

Intension lock

$l_1(B)$
 $l(D)$
 $R(D)$
 $l(E)$
 $w(E)$

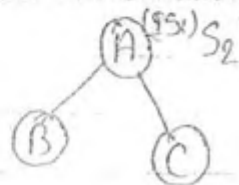
Direct lock

l_1
 $l(B)$
 $R(D)$
 $w(E)$

There are following Intension locks:-

[1] Intension Shared lock [IS]:- A Node N locked by Transaction T in intension shared (IS) mode means that any descendent of N can be request shared lock (S) by transaction T.

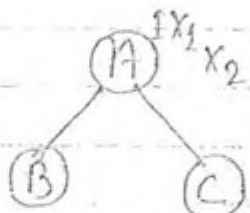
⇒ If parent is Intension shared lock it only allow the the read the data item.



T1	T2
IS(A)	S(A) (only shared lock)
S(B)	R(B)
R(B)	R(C)
S(C)	W(C)
R(C)	W(C)
W(C)	

[2] Intension Exclusive lock [IX]:-

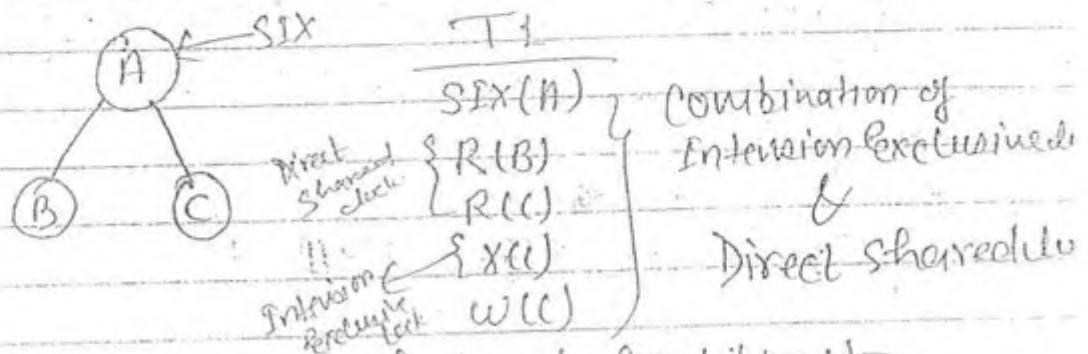
A node N locked by Transaction T in intension Exclusive mode means that Any descendent of N can be request shared / Exclusive lock by transaction T.



T1	T2
IX(A)	X(A) (only Exclusive lock)
S(B)	R(B)
R(B)	W(C)
X(C)	
W(C)	

B) Shared Intension Exclusive Lock [SIX]-

It's Combination of Intension Exclusive lock & Shared lock



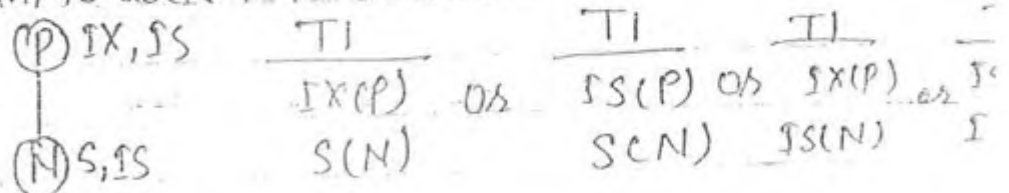
Multilevel Granularity Protocol Conditions:-

- [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 detail 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 element in the array. then what will be the value of count after execution of the program
a = 4 4 5 5 6 6 6 7 7 7 8 8 8
count = 0

```
for (i = 1 to a)
    if (a[i] = a[i+1]) then
        count = count + 1
    else if
        i++;
```

Ques) a) 3 b) 5 c) 6 d) 7

Ques) float a = 63.75 b = 35.75
if float a = 35.75 b = 71.50
b = a
if (a = b)
b = 2a
b = b/2

then what will the value of b

Ques) which join is used to select only distinct column

a) Natural join b) Outer join c) equijoin d) none of these

Ques) which one of them is so important in Digital signature

a) Hash fun b) Encryption c) data hiding d) Decryption

Ques) Int p, q then which of them is not correct operation
a) $p+q$ b) $p-q$ c) $p*q$ d) p/q

Ques) In the given sorted array if ~~there are~~ two element are same 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 VRL

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, x, y, y, y and z, z, z are element arranged in order then in preorder which element is taken first

a) x, x, x b) y, y, y c) z, z, z d) None of these

Ques) Calculate the value of $-a/b + -c/d$ if $a=2, b=4, c=3, d=5$

Ques) what is the full form of SSL in ~~sec~~ 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) If in a ER Model which of them is not so important to taken

a) Entity b) Attribute c) Primary key d) Relationship

Ques) No. of minimum interchange required to keep the maximum no. of element at the root in the tree.

89, 17, 12, 15, 2, 3, 4, 79,

Ques) If $(x \geq y \geq z \geq 200:300)$ if $x=4, y=4, z=6$

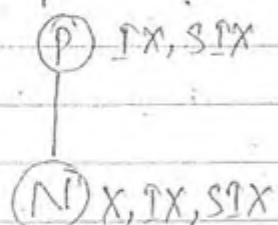
a) 200 b) 300 c) 100 d) None of these

Ques) $S = \{a^m b^n \mid m \neq n\}$ which of them is not belongs to this language

a) $aabb$ b) $abab$ c) $aaabbb$ d) $aabab$

T_1	T_2
$X(A)$	
$X(B)$	$X(A)$
$W(B)$	$X(B)$
	$W(A)$

- [3] A Node N can be locked by Transaction T in "X, IX, S, SX" mode only if parent of N is a locked by Transaction T in "IX, SX" mode.



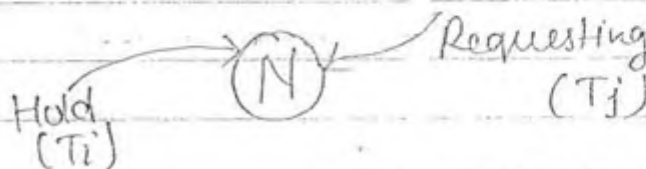
- [4] A node can be request for any lock on Node only if Transaction T not performed any unlock operation (2PL)
(This condition is required to ensure serializability)

- [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 ^{done} from bottom Top.)

[6] Compatible Table:-

Compatibility:-

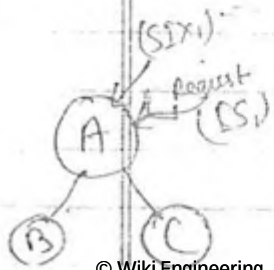
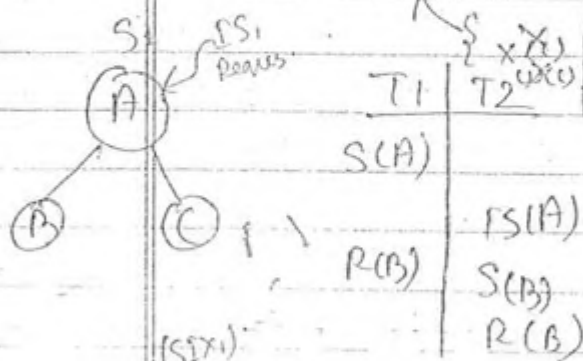
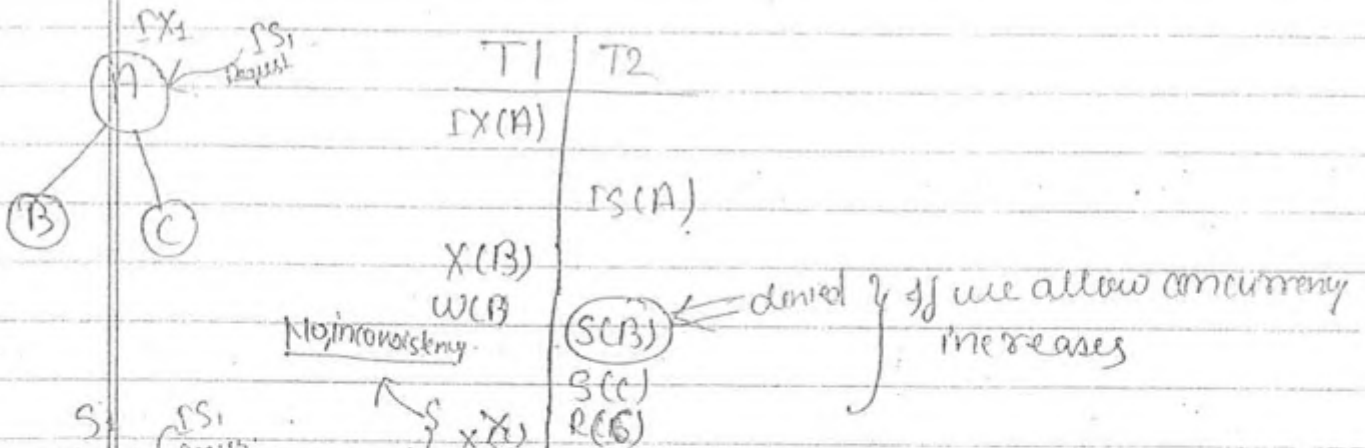
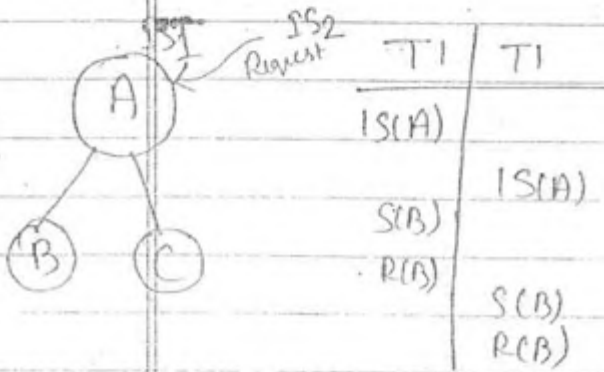
If Node N is locked by Transaction (T_i) & another transaction (T_j) request for locking the same node then compatibility that whether to grant the permission or deny the request of transaction (T_j) .



where $i \neq j$

T_j	IS	IX	S	SIX	Hold T_r
IS	YES	YES	YES	YES	No
IX	YES	YES	No	No	No
S	YES	No	YES	NO	NO
SIX	YES	NO	NO	NO	NO
X	No	No	NO	NO	NO

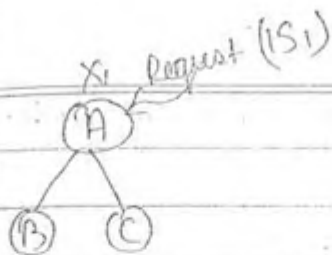
Request T_j



$SIX \leftrightarrow IS$

$S = IS \Rightarrow YES$

$IX = IS \Rightarrow YES$



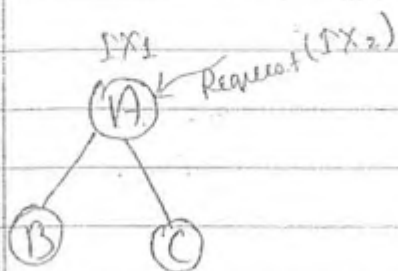
T1	T2
X(A)	

IS(A) ← if we grant

SC(B) → if we grant then RW problem may possible

R(B)

W(B)



T1	T2
S(A)	

$IX(A)$ ← grant

X(B)

W(B)

R(B)

SIX — IX → S-IX(N)
IX-IX(Y)

SIX — IS → S-IS(Y)
IX-IS(Y)

* Intension — Intension lock

IX	—	IX	} Compatible
IX	—	IS	
IS	—	IX	
IS	—	IS	

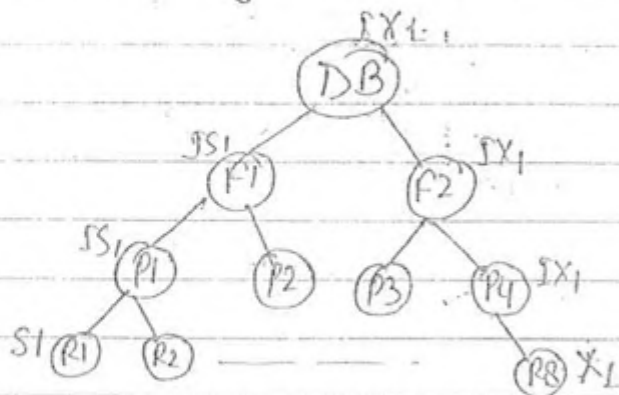
Simultaneous Read Compatible:-

S	—	IS	} Compatible
S	—	S	
IS	—	S	

* If the transaction satisfy all the above six condition then only it is called multilevel granular

(Ques)

T1: Read R1 & update R8 then what will be the locking pattern.



T1

IX₁(DB)

IS₁(F₁)

IS₁(P₁)

S₁(R₁)

Read(R₁)

IX(F₂)

IX(P₄)

X(R₈)

write(R₈)

U(R₈)

U(P₄)

U(F₂)

U(R₁)

U(P₁)

U(F₁)

U(DB)

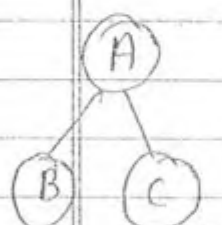
⇒ X

Multitenet Granularity Protocol ensure serializability.

(Equivalent Serial schedule based on lock points)
(becoz it is in 2PL)

⇒ Irrecoverability Possible becoz there is no any restriction for commit of any transaction

Example - check multilevel granularity is possible or not:



T1	T2
IX(A)	
X(B)	
W(B)	
U(B)	
U(A)	
	IS(A)
	SC(B)
	R(B)
	U(B)
	U(A)
	Commit
Commit	

i.e. multilevel granularity is possible but it is Irrecoverable schedule

⇒ Deadlock & Starvation possible.

* To remove the Irrecoverability from Multilevel granularity protocol we put some restriction on it then it become called as strict Multilevel granularity protocol.

Strict Multilevel Granularity protocol:-

Multilevel Granularity protocol + all Exclusive locks should be hold until commit/rollback

⇒ Ensure Serializability & Strict Recoverability

⇒ Even deadlock & starvation is possible.

Timestamp Ordering Protocol:-

One difference between locking & Timestamp protocol is that timestamp is a protocol which is always free from Deadlock.

→ (Free from Deadlock)

Timestamp Value:-

Timestamp is a unique value assigned by DBMS to every transaction in ascending order.

Advantages of Timestamp value:-

1) Because it is unique value, TS is used for identification.

2) Because it is in ascending order, TS is used for to set priority.

(Transaction with less TS value gets high priority).

Example:-

10	11	12	13
T1	T2	T3	T4

older younger

* Every data in the DB maintain two TS values

1) Read TS(A):-

Highest Transaction TS value that has performed R(A) operation successfully.

10	20	30	40
T1	T2	T3	T4
R(A)		R(A)	
	R(A)		

$$RTS(A) = \cancel{10} 30$$

2) Write TSCA:-

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 stamp ordering

(T1:T3:T2)

* Concurrency of the Timestamp protocol is less than locking protocol because in timestamp protocol concurrent execution should be equal to only one serial schedule based on timestamp ordering but in locking protocol concurrent execution should be equal to any serial schedule which satisfy 2PL condition. So, Timestamp protocol is less concurrent.

Example:-

10	30	20
T1	T2	T3
	R(A)	
R(B)		
	w(B)	
		w(B)
		w(A)

rollback
↑
w(B)
↑
w(A)

It violates the serial i.e. T1 → T3 → T2
So T3 is rolled back

If given serial is T1 → T3 → T2

⇒ T1 → T3
T1 → T2
T2 → T3

* $T2 \rightarrow T1$ is possible but $T1 \rightarrow T2$ is not allowed i.e. cyclic dependency is not possible so it is deadlock free protocol.

Possibility of Rollback 1-

1)

10 T1	20 T2
$R(A)$	$W(A)$

i.e. $T1 \rightarrow T2$

Transaction
"If younger updated data item. Then older not allowed to Read" being conflict pair from $T2 \rightarrow T1$.

(Rollbacking T1)

Transaction $T1$ issues $R(A)$ operation:-

$WTS > TS(T1)$
Rollback $T1$

2)

10 T1	20 T2
$W(A)$	$R(A)$

i.e. $T1 \rightarrow T2$

"If younger transaction read data item A. Then older transaction not allowed update A" being conflict pair order is from $T2 \rightarrow T1$.

(Rollback T1)

Transaction $T1$ issues $W(A)$ operation:-

$RTS > TS(T1)$
Rollback $T1$

3)

10 T1	20 T2
$W(A)$	$W(A)$

i.e. $T1 \rightarrow T2$

"If younger transaction updated data item. Then older transaction not allowed to update" being conflict pair order is from $T2 \rightarrow T1$.

(Rollback T1)

* When conflict pair is based on timestamp order then there is no need to rollback. But when conflict pair is not based on time stamp order then we have to rollback the transaction.

classmate
Date _____
Page _____

Transaction T issues W(A) operation:-

$$WTS(A) > TS(T)$$

Rollback T

Basic Time Stamp Ordering Protocol:-

a) Transaction T issues R(A) operation:-

[I] if $WTS(A) > TS(T)$ Then Rollback T

[II] Otherwise allowed to execute R(A) operation by Transaction T

Then Set $RTS(A) = \max\{RTS(A), TS(T)\}$

Ex:-

10	20	30
T1	T2	T3

R(A)

$$RTS(A) = \cancel{10} 30$$

R(A)

R(A) { Here is no any conflict pair then we allow this trans.

b) Transaction T issues W(A) operation:-

[I] if $RTS(A) > TS(T)$ Rollback T

[II] if $WTS(A) > TS(T)$ Rollback T

[III] Otherwise allowed to execute W(A) operation by Transaction T

then

$$Set\ WTS(A) = TS(T)$$

Ex:-

10	20	30
T1	T2	T3

W(A)

$$WTS(A) = \cancel{10} 30$$

W(A)

Rollback
W(A)

* By default we take transaction no. as timestamp value if the timestamp value is not given.

(Que) which of the time stamp values allow to execute the schedule S

S: $r_1(A) r_2(B) w_1(C) r_3(B) r_3(C) w_2(B) w_3(A)$

a) { 10, 20, 30 }

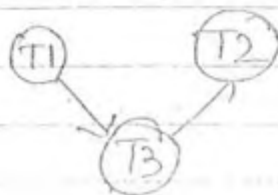
b) { 30, 10, 20 }

c) { 10, 30, 20 }

d) { 30, 20, 10 }

$r_1(A) : w_3(A) \rightarrow 1 \rightarrow 3$

$r_3(B) : w_2(B) \rightarrow 3 \rightarrow 2$



(Que) if time stamp {10, 20, 30} are given then which transaction is roll back.

{10, 20, 30}

$T1 \rightarrow T2 \rightarrow T3$

$T1 \rightarrow T2$

$T2 \rightarrow T3$

$T1 \rightarrow T3$

then check in the transaction one by one by taking operation from above schedule and check by taking previous operation, that where conflict pair is and is the conflict pair satisfy above precedence of the given question.

S: $r_1(A) r_2(B) w_1(C) r_3(B) r_3(C) w_2(B) w_3(A)$

$T3 \rightarrow T2 \neq T2 \rightarrow T3$

So, $T2$ transaction roll back

Once you discard the one transaction then discard all the operation related to that transaction for further checking that whose other transaction will be rolled back.

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! $S_1(A) S_2(B) W_1(C) S_3(B) S_3(C) W_2(B) W_3(A)$

$W_1(C)$ is conflict with $S_3(C) \Rightarrow 1 \rightarrow 3$ not allowed

So, we discard all operation of T
 $\Rightarrow$ Rollback the Transaction T_3 .

c) T_1, T_2, T_3
{ 30, 20, 10 }

$T_3 \rightarrow T_2 \rightarrow T_1$

$T_3 \rightarrow T_2$

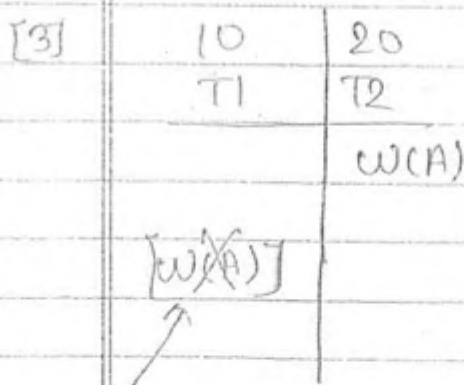
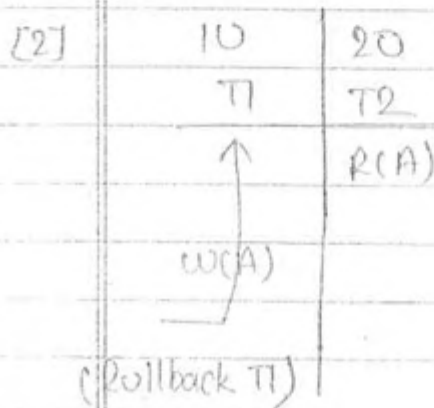
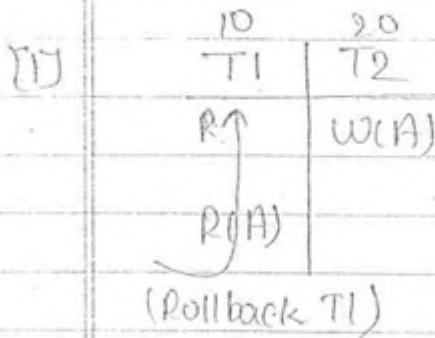
$T_2 \rightarrow T_1$

$T_3 \rightarrow T_1$

S! $S_1(A) S_2(B) W_1(C) S_3(B) S_3(C) W_2(B) W_3(A)$

$S_1(A) \rightarrow W_3(A) \Rightarrow 1 \rightarrow 3$ (not possible)

So, T_3 is rolled back.

THOMAS WRITE TIMESTAMP ORDERING PROTOCOL-Possibilities of Rollback:-

el If younger transaction updated data item then older transaction updations are ignored"

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 $WTS(A) = TS(T)$

Ques)

10 T1	20 T2	30 T3
R(A)		
w(A)	w(A)	
Rollback		w(A)

Basic TO protocol:- (close to conflict Serializable)

Rollback T₁ becoz of w₁(A)

TWR TO protocol:- (close to view serializable sch)

10 T1	20 T2	30 T3
R(A)		
w(A)	w(A)	
Rollback		w(A)

No Rollback

(w₁(A) ignored)

$\Rightarrow T1 \rightarrow T2 \rightarrow T3$ is equal to view

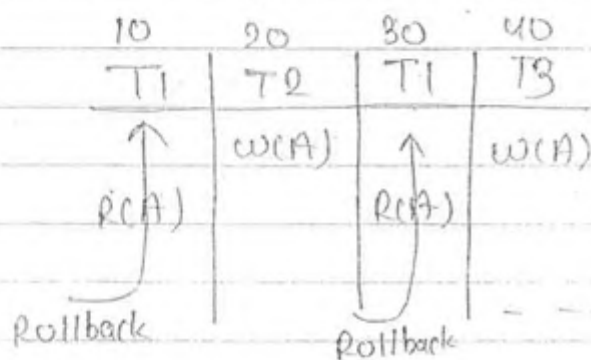
serializable

RULE of Thomas Write Time Stamp Ordering protocol & Basic TS ordering:-

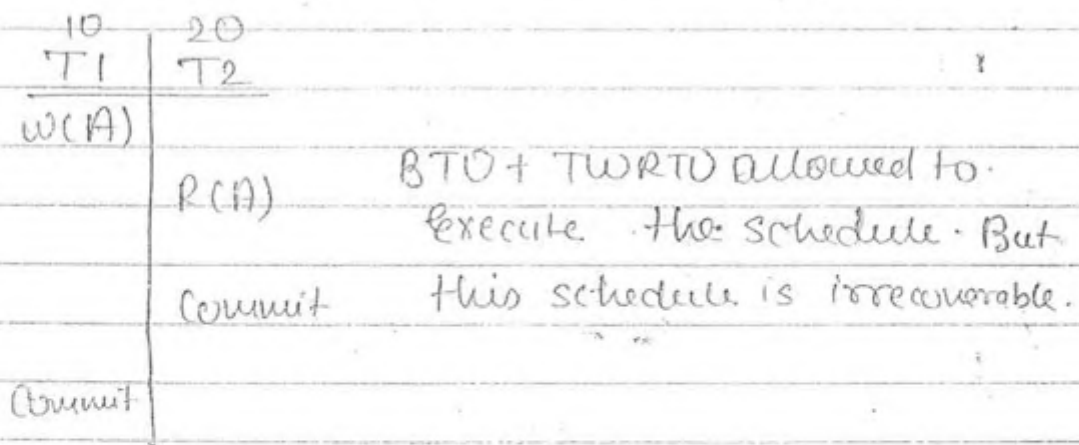
[1] Serializability
(Equivalent Serial based on order of TS values)

[2] Free from Deadlocks

[3] Not free from Starvation



[4] May not free from irrecoverability



Strict Recoverability

10	T1	T2	20

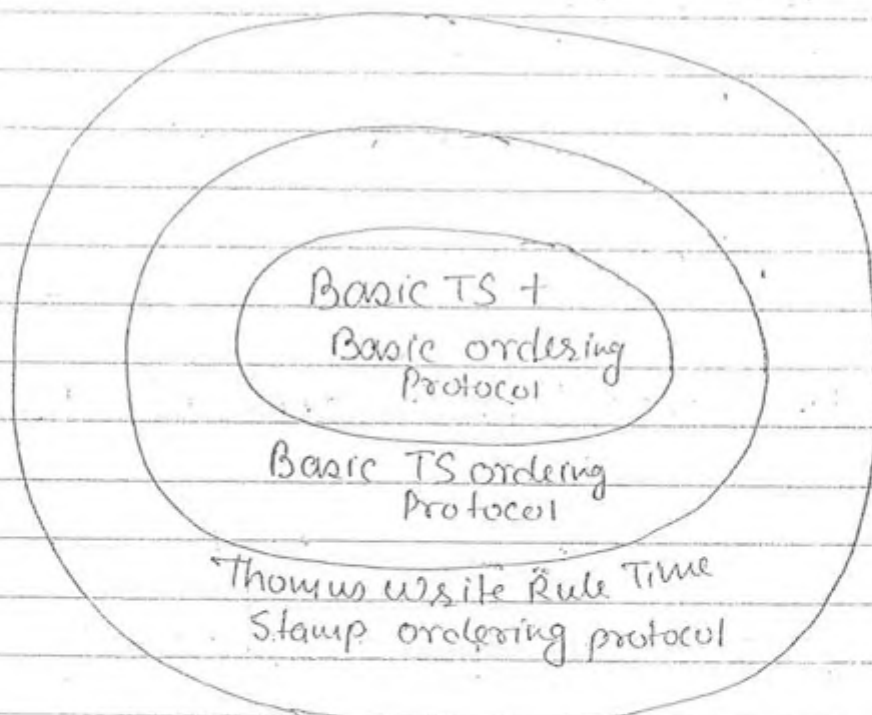
10	20
T1	T2
w(A)	
C/R	R(A)/w(A)

i.e. $WTS(A) < TS(T2)$

Strict Timestamp Ordering Protocol:-

Transaction T_2 issues $R(A)/w(A)$ operations if $WTS(A) < TS(T_2)$. Then transaction (T_2) which issuing $R(A)/w(A)$ operation should be delayed until commit/Rollback of T_1 that is already performed write operation.

*



31/08/2011

classmate

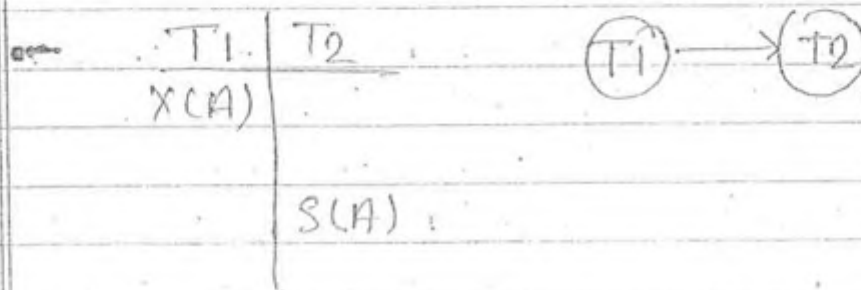
Date _____

Page _____

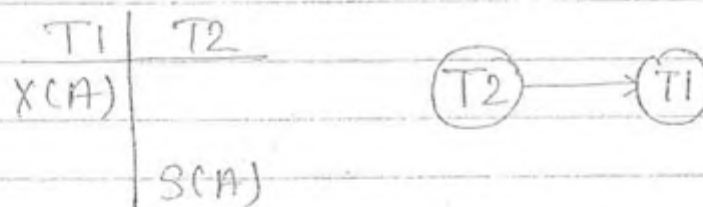
Deadlock Prevention Algorithm:-

The basic purpose of deadlock prevention algorithm is to prevent the deadlocks in lock based protocol using Time Stamp Values.

Precedence Graph:-



Dependency Graph:-



⇒ i.e. dependency graph is opposite of Precedence graph.

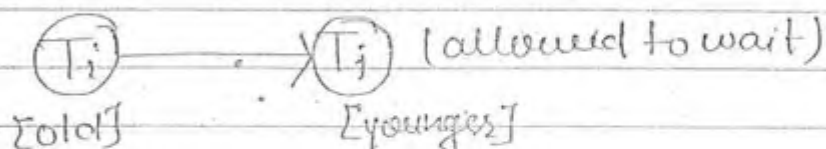
These are two types of protocol:-

[I] Wait-Die Protocol:-

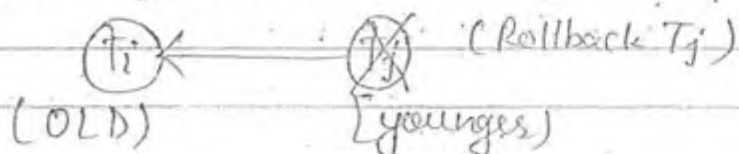
a) Assume $TS(T_i) < TS(T_j)$

b) If Transaction T_i required resources

that is holded by T_j , Then T_i allowed to wait until T_j unlocks.



[ii] If Transaction T_j required resource that is hold by T_i , then Rollback T_j (& to remove starvation) Restart with same Timestamp value.



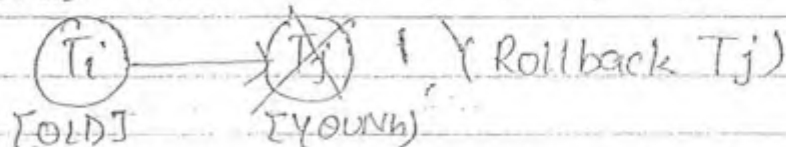
⇒ Deadlocks are not possible here becoz we are not allowed to form cyclic dependency.



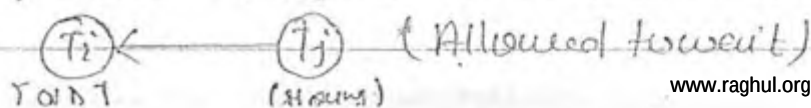
[2] WOUND-WAIT Protocol

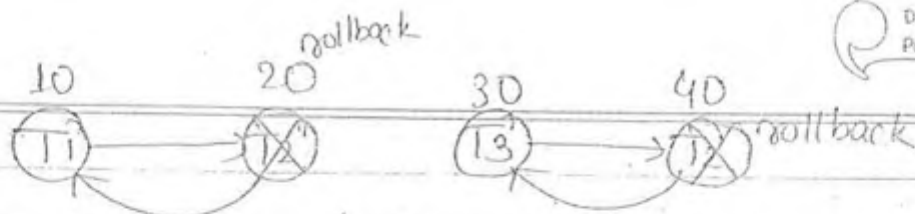
a) Assume $TS(T_i) < TS(T_j)$

b) [i] If Transaction T_i required resource that is holded by T_j , then Rollback T_j (& to remove starvation) Restart with same Timestamp value.

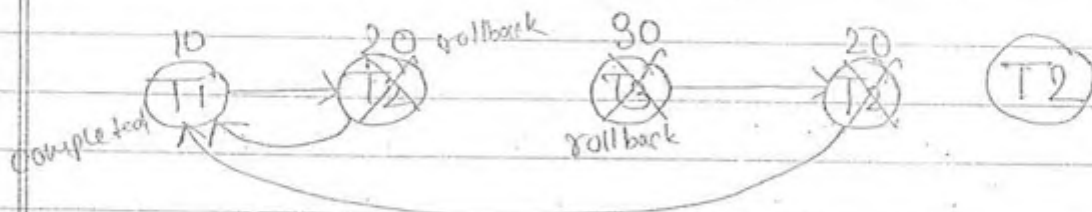


[ii] If Transaction T_j required resource that is holded by T_i , then T_j allowed to wait until T_i unlocks.





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 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

- RW
- WR
- WW

→ Classification of schedule

- Recoverability
- Serializability

→ Protocols

- ***
- locking
- Timestamp

Indexing & File Organization:-

Database files Divided into Blocks or pages.
Page divided into records.

i.e. Databas file collection of Blocks.
Each Block collection of Records.

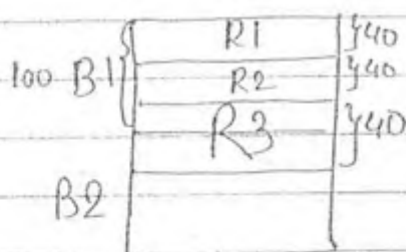
Records are stored in a Block in Two ways:-

[I] Spanned Organization:-

[II] Unspanned Organization

[I] Spanned Organization:-

Record can be spanned b/w the blocks.



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 fragments.

Disadvantage:-

⇒ More I/O cost.

[2] Unspanned Organization:-

Entire record should be accommodate in single block.

B1	R1	40
	R2	40
	//////	20
B2	R3	
	R4	
	//////	

Block Size = 100 bytes.

Record Size = 40 bytes

$$\text{Block Factor} = \left\lfloor \frac{\text{Block Size}}{\text{Record Size}} \right\rfloor$$

$$= \frac{100}{40} = 2$$

Disadvantage:-

⇒ Internal fragmentation Possible.

Advantage:-

⇒ less I/O cost.

becoz

Secondary memory $\xrightarrow{\text{(block by block)}}$ Main memory

⇒ 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 or search the data.

Select *
From Emp
where Ppno = Y
Search key

Search key is a key use to compare the data of Ppno.

Page

Worst Case I/O cost:-

if the file is ordered file: I/O cost:- $\lceil \log_2 N \rceil$

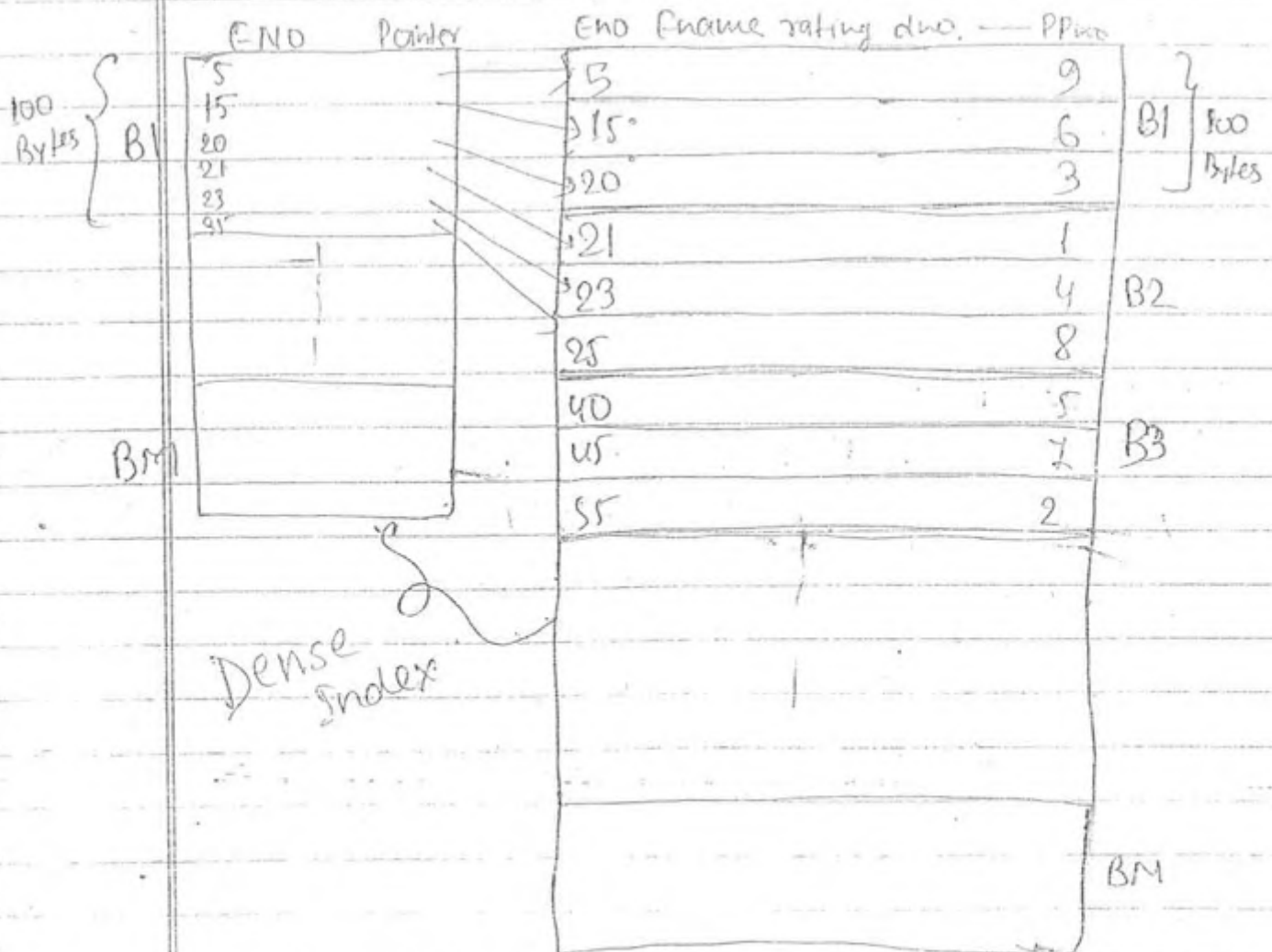
if the file is un-ordered: N blocks

To reduce the I/O cost from $\lceil \log_2 N \rceil$ to N we go for Indexing:

INDEXING:-

Idea 1:- [To reduce I/O cost]

Size of DB Block = Size of Index block
(Equip DB Ark)



⇒ Entry of the Index File!- Two fields

← Search key, Pointer

ex!- ← ENO, Pointer

⇒ Entry size of Index File $\ll$ 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 $\ll$ Total No. of DB File Blocks.

$$(M \ll N)$$

⇒ I/O cost with Index = $\lceil \log_2 M \rceil + 1 \ll \lceil \log_2 N \rceil$
or
N

Categories of Index!-

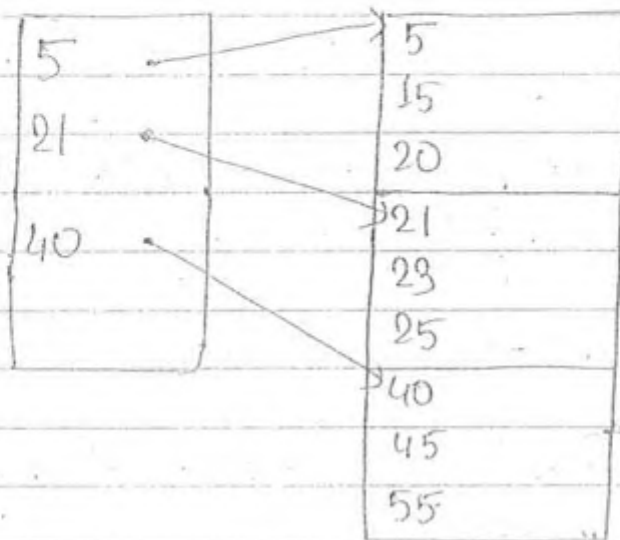
[i] Dense Index!-

For every database record there should be corresponding entry in the index file. (i.e. one to one mapping between index 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 $\ll$ No. of ^{DB} ~~Index~~ Records
~~File Entries~~

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 12 bytes takes keys & pointer take 8 bytes.

10000 records exist in the database.

How many no. of Dense & sparse Index block?

$$\text{Block Factor of DB File} = \frac{1000}{100} = 10 \text{ Record/block}$$

$$\text{No. of DB Blocks} = \frac{\text{No. of records}}{\text{Block Factor}} = \frac{10000}{10} = 1000 \text{ Block}$$

I/O cost without index $\Rightarrow \lceil \log_2 1000 \rceil = 10 \text{ Blocks}$

$$\frac{O/N}{N} = 1000 \text{ Blos}$$

By using Indexing:-

Entry Size of Index = $12 + 8 = 20 \text{ Bytes}$

Block factor for index file = $\frac{1000}{\text{entry size}} \cdot \frac{1000}{50} = 50$

Dense Index

No. of Index File Entries -
No. of D.B record
= 10,000

No. of Block = $\frac{200}{50}$
= 200 Blocks

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

No. of Block = $\frac{20}{50}$
= 20 Blocks

I/O cost = $\lceil \log_2 20 \rceil + 1$
= $5 + 1 = 6$

Types of Indexes:-

[1] Primary Index:-

Index is said to be primary index if it satisfied following condition:-

(1) Conditions:-

(1) Search key used in the index file should be

used to physically 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.

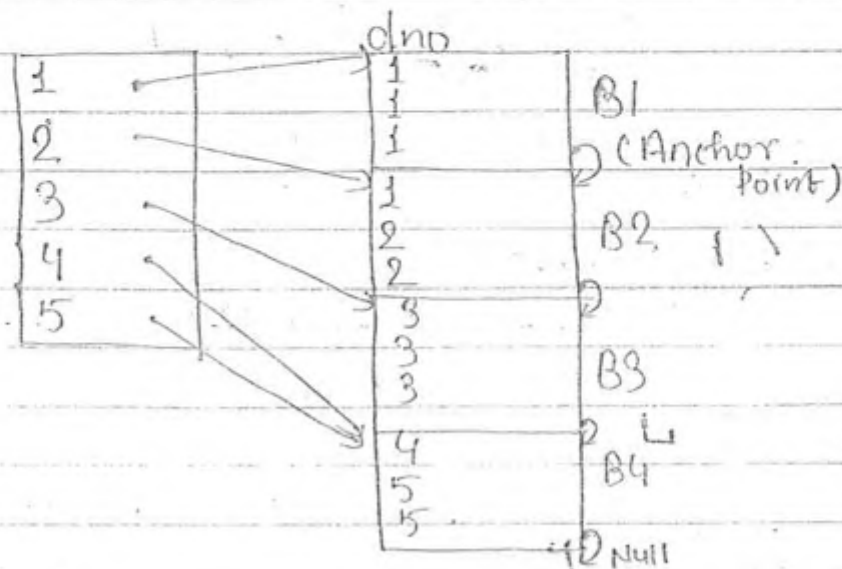
③ At most one primary Index is possible.

[2] Clustering Index:-

① 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.

classmate

Date _____

Page _____

* Block Anchors are required in clustering index.

* If every employee comes under the same department then I/O cost = $1 + N = N$

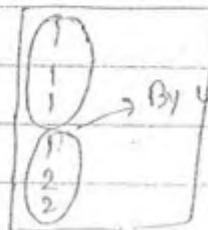
D_1
 D_2

B_N

Select *

From Emp

where dno = 1



By using Anchor point

(b) Always sparse

(c) At most one clustering index is possible

if $\left\{ \begin{array}{l} \text{Select *} \\ \text{From Emp} \\ \text{where dno = x} \end{array} \right\}$ 50%

OR

$N \left\{ \begin{array}{l} \text{Select *} \\ \text{From Emp} \\ \text{where pno = x} \end{array} \right\}$ 50%

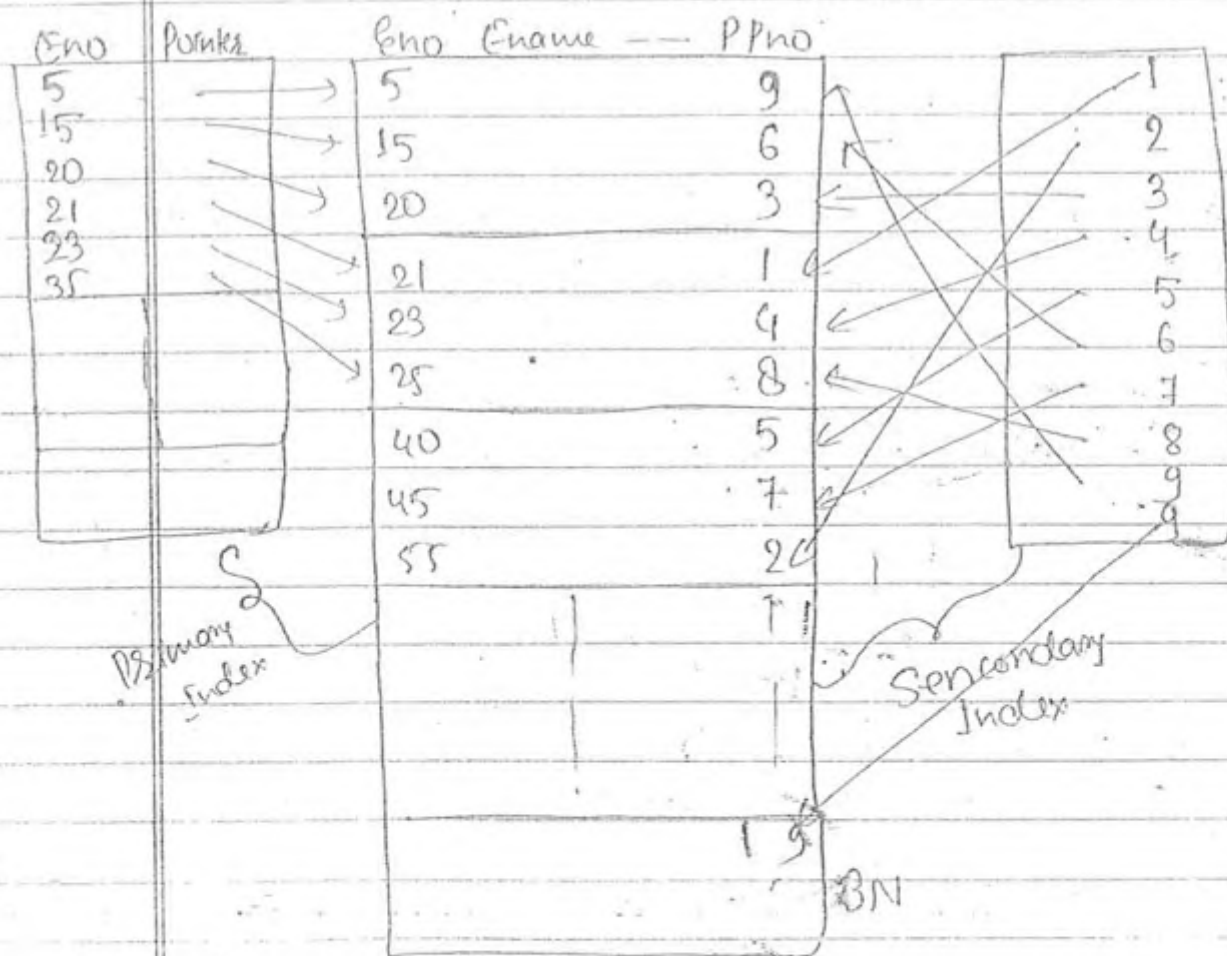
[8] Secondary Index:-

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.



$\Rightarrow$ 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 or — — Always Sparse.

$\Rightarrow$ ~~IO~~ Cost = No. of levels + 1

Page 55
Ex-11)

16384 Records

Record = 32 byte

File size = 6 byte

Ordered on 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 B-tree
= 16384

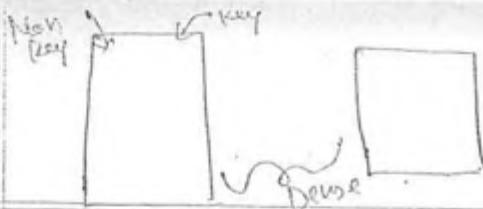
Block Size = 1024 bytes

Entry size = ~~1024~~ 6 + 10
= 16

Block factor = $\frac{1024}{16}$

$\frac{1024}{16} = 64$ Entries per block

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



classmate

Date _____

Page _____

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

No. Blocks = $\{256 + 4 + 1\}$: Blocks

Ques) If Ordered on key &
SPARSE Index build on key in the above Ques)
then 1) No. of levels = 22
2) No. of Index blocks = 22

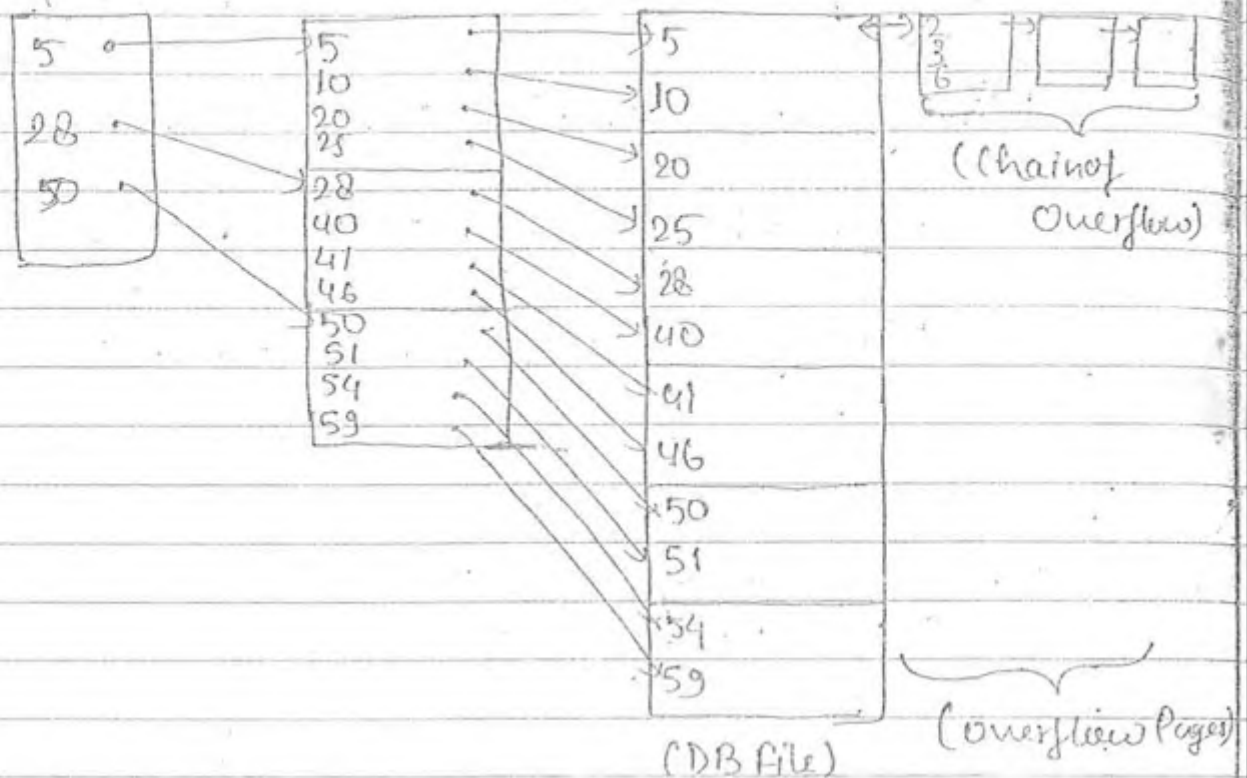
1/09/2011

classmate

Date _____

Page _____

STATIC MULTILEVEL INDEX:-



Limitation of Static Multilevel Index:-

→ Insertion:-

[1] Shifting DB:-

Entire DB file and Index file required to shift becoz of New insertion. i.e. (Entire Index file changed)

[2] Overflow Page:-

Result, more I/O cost.
Worst case: $O(N)$ blocks I/O cost

→ Deletion:-

[1] Shifting DB:-

Which results Entire Index file change.

[2] Leave Space:-

Some Index block may result 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:-

[1] Insertion/Deletion:-

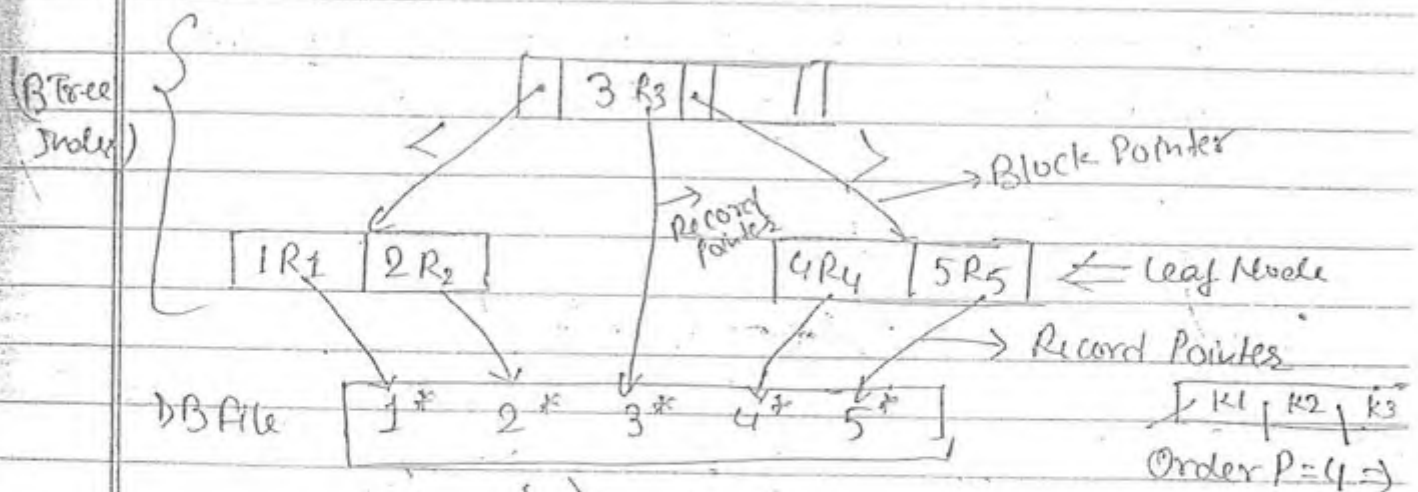
i.e. Insertion/deletion should not result entire index file changes.

[2] No overflow pages allowed.

[3] Because of deletion No Index Node should be less than 50% occupied.

* B Tree & B+ Tree comes under Dynamic Multilevel Index.

B Tree:-



Record Pointers
(04)

Data Pointers
(04)

Value Pointers

Block Pointers
(04)

Node Pointers
(03)

Child Pointers
(03)

Tree Pointers

4 B P

3 Key

3 R P

POI

I/O Cost To access any Record =

$$O(\log_p N)$$

N : No. of keys. = 5

P : maximum no. of child pointers in
B-Tree Node = 3

To access all records:-

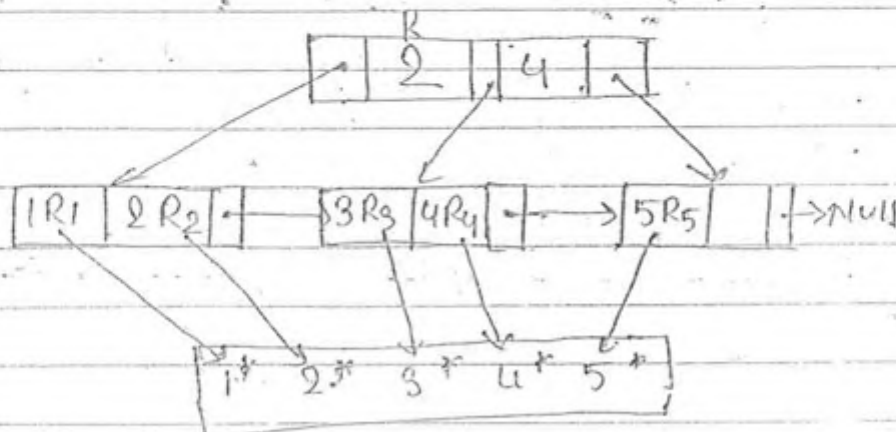
$$\text{I/O cost} = O(2^M)$$

M = No. of index Nodes.

⇒ Not best Suitable for to access all records of the DB.

To avoid the problem in accessing all records in B tree we go through B⁺ Tree.

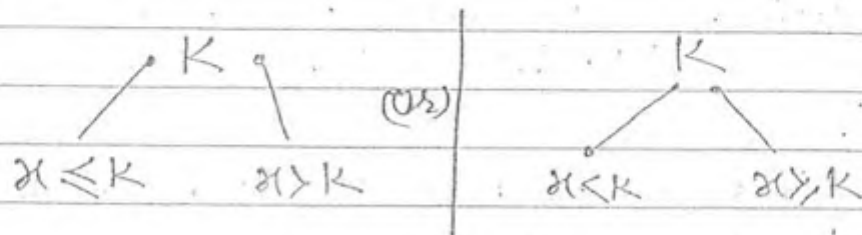
B⁺ Tree:-



⇒ Every key is there at leaf Node.

⇒ Every leaf Node pointed to next leaf node.

if K is a key then Block pointers point the value whose keys value x follow below pattern i.e.



⇒ I/O cost To access any record = $O(\log_p N)$

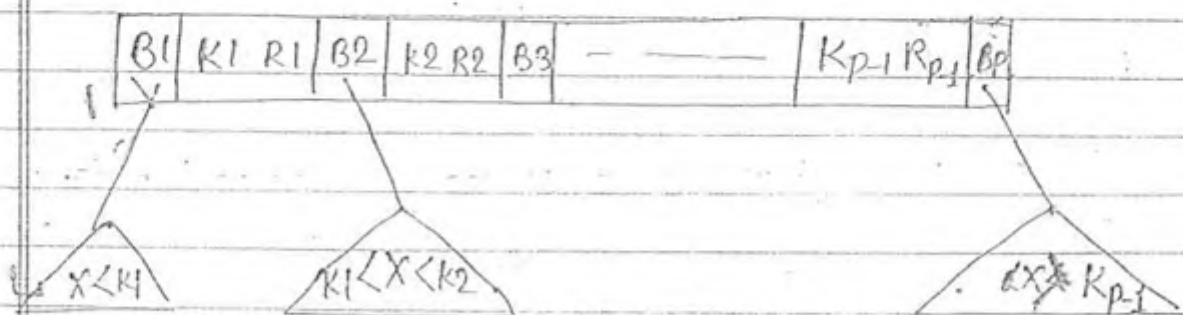
⇒ All records only required leaf Nodes.

B-Tree Definition:-

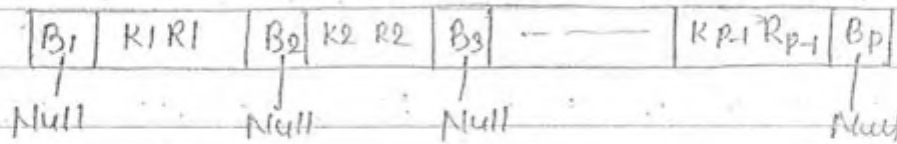
ORDER P = maximum no. of block pointers in B-Tree node

① Structure of Internal Node:-

$$PB_p + (P-1) \text{ keys} + (P-1) \text{ Record Pointers} \leq \text{Index Block}$$



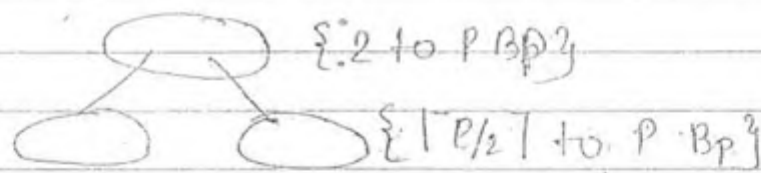
② Structure of Leaf Node:-



③ Keys within the node should be in Ascending order $\{K1 < K2 < \dots < K_{p-1}\}$

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.

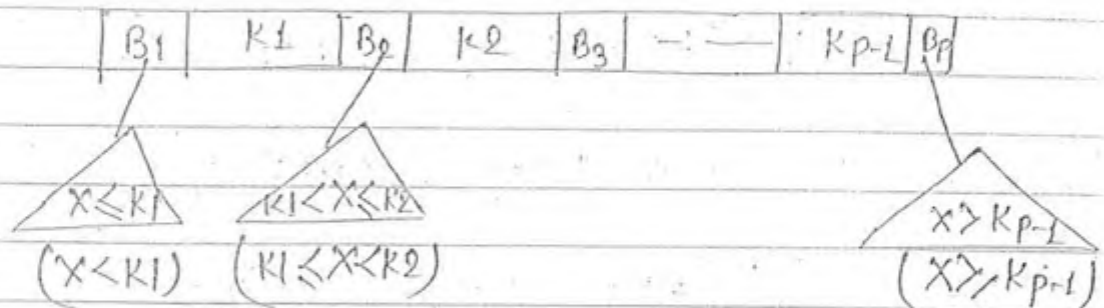
B+ Tree Definition:-

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: $\{K1 < K2 < \dots < Kp-1\}$

④ Every Node Except Root should be at least $\lceil P/2 \rceil$ Block pointers at most P block pointers.

⑤ Root Node can be at least 2BP at most P Block pointers.

⑥ Every leaf node should be same level.

(Ques) Internal node B⁺ Tree order P = maximum no. of keys

$$(P+1)BP + P \text{ keys} \leq \text{Block Size}$$

Order P : min P keys, max $2P$ keys in B⁺ Tree internal node.

$$(2P+1)BP + 2P \text{ keys} \leq \text{Block Size}$$

Order P : min P keys max $2P$ keys in B-Tree node

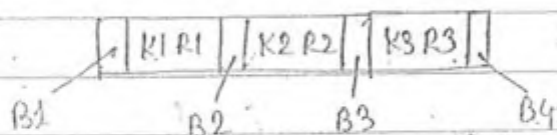
$$(2P+1)BP + 2P \text{ keys} + 2P \text{ RP} \leq \text{Block Size}$$

Order P : min P BP max $2P$ Block pointers in B Tree Node

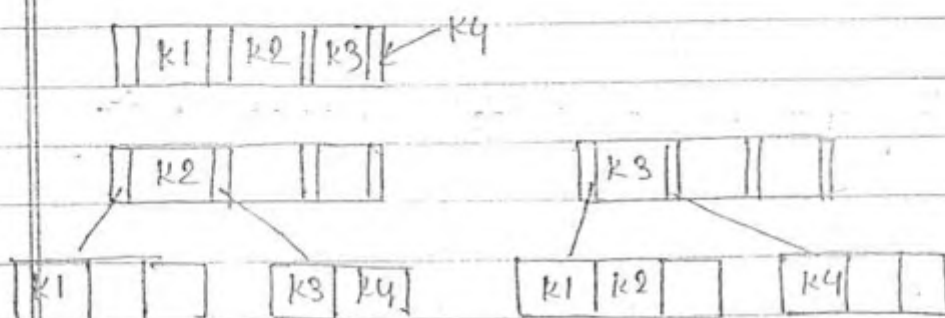
$$(2P)BP + (2P-1) \text{ keys} + (2P-1) \text{ record pointer} \leq \text{Block size}$$

(Ques) Construct B-Tree with order $P=4$ and following sequence of keys.

20, 5, 10, 3, 12, 18, 15, 45, 38, 69, 93



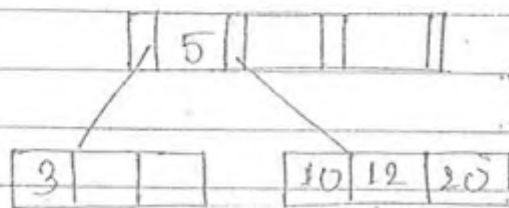
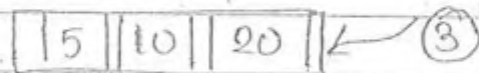
Node Splitting



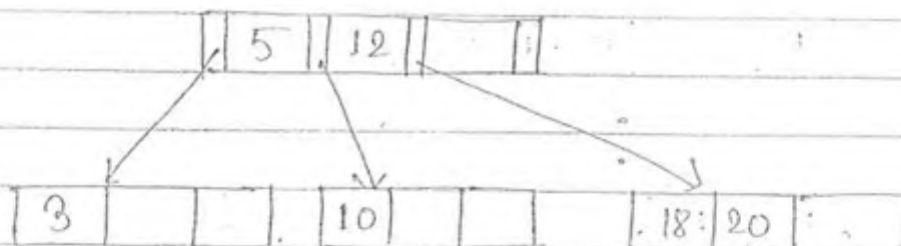


Balanced = Height Restricted = $O(\log_p N)$

i.e. height must not go higher than $\log_p N$ for balancing the tree.

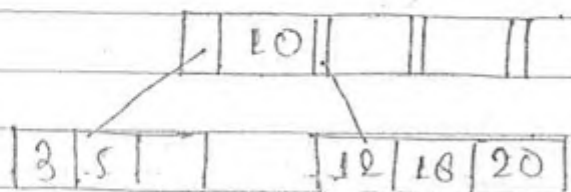


a) Node Splitting:-



b) Redistribution of the keys:-

If any leaf Node contains some free position we can distribute key to accommodate New key in only Existing Nodes.



If redistribution is not possible then go for Node splitting

By using redistribution

[5] [10] [20] ← 3

[5] [] [] []

[3] [] [] [10] [12] [20] ← 18

[10] [] [] []

[3] [5] [] [12] [18] [20] ← 15

[12] [] [] []

[3] [5] [10] [15] [18] [20] ← 45

[12] [18] [] []

[3] [5] [10] [15] [] [] [20] [45] ← 38

[12] [18] [] []

[3] [5] [10] [15] [] [] [20] [38] [45] ← 69

[12] [20] [] []

[3] [5] [10] [15] [18] [] [38] [45] [69] ← 33

[12] [33] [] []

[3] [5] [10] [15] [18] [20] [38] [45] [69]

B-Tree

ORDER P: maximum no. of Bp in B Tree Node.

Height/Lvl	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	$2 \lceil P/2 \rceil^2$	$2 \lceil P/2 \rceil^3$	$2 \lceil P/2 \rceil^2 (\lceil P/2 \rceil - 1)$
h-1/h	$2 \lceil P/2 \rceil^{h-1}$	$2 \lceil P/2 \rceil^h$	$2 \lceil P/2 \rceil^{h-1} (\lceil P/2 \rceil - 1)$
h/h+1	$2 \lceil P/2 \rceil^{h-1}$	$2 \lceil P/2 \rceil^h$	$2 \lceil P/2 \rceil^{h-1} (\lceil P/2 \rceil - 1)$

(Minimum null pointers)

Total min. no. of keys in B-Trees with order P & height

$$1 + 2(\lceil P/2 \rceil - 1) + 2 \lceil P/2 \rceil (\lceil P/2 \rceil - 1) + \dots + 2 \lceil P/2 \rceil^{h-1} (\lceil P/2 \rceil - 1)$$

$$1 + 2(\lceil P/2 \rceil - 1) \left\{ 1 + \lceil P/2 \rceil + \lceil P/2 \rceil^2 + \dots + \lceil P/2 \rceil^{h-1} \right\}$$

$$1 + 2(\lceil P/2 \rceil - 1) \left\{ \frac{\lceil P/2 \rceil^h - 1}{\lceil P/2 \rceil - 1} \right\} = 1 + 2 \lceil P/2 \rceil^{h-1}$$

Total min. no. of keys in B-Tree with order P & level l .

$$= 1 + 2 \left\{ \left\lceil \frac{P}{2} \right\rceil^{l-1} - 1 \right\}$$

(Ques) Total min. no. of keys in B-Trees with order P & level l .

a) $1 + 2 \left\{ \left\lceil \frac{P}{2} \right\rceil^l - 1 \right\}$ { Here ^{levels} starts from 0 }

b) $P^{l+1} - 1$

c) $P^{2l+1} - 1$

d) $\left\{ \left\lceil \frac{P}{2} \right\rceil^{2l+1} - 1 \right\}$

Height/Level	max. no. of nodes	Maximum no. of Block pointer	Maximum no. of keys
0/1	1	P	$P-1$
1/2	P	P^2	$P(P-1)$
2/3	P^2	P^3	$P^2(P-1)$
3/4	P^3	P^4	$P^3(P-1)$
h/h	P^h	P^{h+1} (Maximum Null pointer)	$P^h(P-1)$

Total maximum no. of keys in B Tree with Order P & height h .

$$(P-1) + P(P-1) + P^2(P-1) + \dots + P^h(P-1)$$

$$(P-1) \{ 1 + P + P^2 + \dots + P^h \}$$

$$(P-1) \left\{ \frac{P^{h+1} - 1}{P - 1} \right\} = \boxed{P^{h+1} - 1}$$

$\Rightarrow$ Total maximum no. of keys in B Tree with order P & level h .

$$\boxed{= P^{h+1} - 1}$$

Height of the B-Tree with N keys:-

$$N = 1 + 2 \left\{ \left\lceil \frac{P}{2} \right\rceil^h - 1 \right\}$$

$$\Rightarrow \boxed{h = \log_{\frac{P}{2}} \left\{ \frac{(N-1)+1}{2} \right\}} \Rightarrow \text{maximum possible height with } N \text{ keys}$$

$$N = P^{h+1} - 1$$

$$\Rightarrow \boxed{h = \log_P (N+1) - 1} \Rightarrow \text{minimum possible height with } N \text{ key}$$

* In B Tree:- Minimum Null pointers = $2 \lceil \frac{P}{2} \rceil^h$

Maximum Null pointers = P^{h+1}

* Height of B Tree:- $O(\log_p N)$

where:-

N = No. of keys

P = Order

Ques) Order P is maximum no. of keys in B Tree node.
How many no. of maximum nodes & keys in B-Tree
with order $P=3$ and level $L=4$

Order $P=3$ $L=4$

Key

One block can accommodate:-

$\frac{P \text{ keys} + (P+1) \text{ Block pointers} + P \text{ record pointer}}{3} \leq \text{Block}$

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	$4^3 = 64$	64×4	64×3
	85	BP pointed Null	255

(Que) 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:-

$$(P)BP + (P-1)Keys + (P-1)R \cdot P \leq \text{Block}$$

Level	minimum no. of nodes	minimum no. of Block pointers	minimum no. of keys
1	1	2	1
2	2	$2 * \lceil \frac{P}{2} \rceil = 2 * 2$	$2 * 1$
3	4	$4 * 2$	$4 * 1$
4	8	$8 * 2$ ↓ Pointed Null	$8 * 1$
	18		15

Page-55
(Que 12)

P : maximum no. of Tree pointers in B-Tree Node

Keys = 10 bytes

Block = 512 bytes

data pointers = 8 bytes

Block pointers = 5 byte

maximum value of $P = ?$

$$P + BP + (P-1)Key + (P-1)RP \leq \text{Block size}$$

$$P * 5 + (P-1)10 + (P-1)8 \leq 512$$

$$23P \leq 512 + 18$$

$$23P \leq 530$$

$$P = \left\lfloor \frac{530}{23} \right\rfloor = 23$$

2/09/11

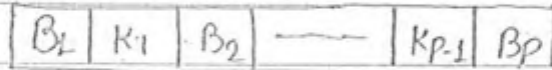
classmate

Date _____

Page _____

B⁺ Tree:-

Internal Node:-



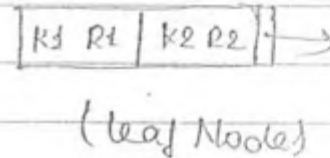
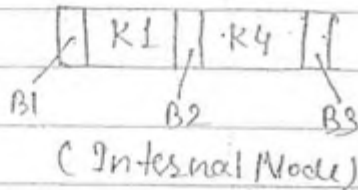
P Block Pointers + (P-1) keys

Page 56

Que 16

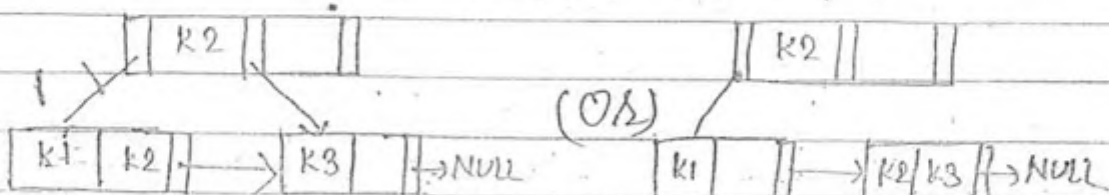
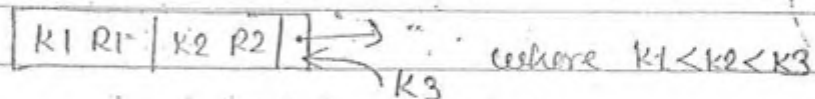
Order of internal Node = 3 (maximum no. of Tree pointers)

Leaf Node = 2 (maximum no. of data item)



Keys Inserted = 10, 3, 6, 8, 4, 2, 1

NODE SPLITTING IN B⁺ Tree:-



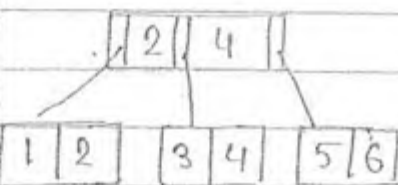
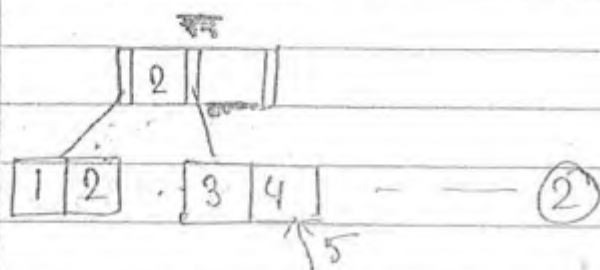
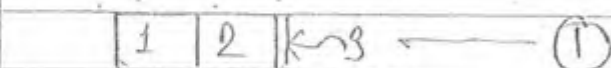
(Left biasing)

(Right biasing)

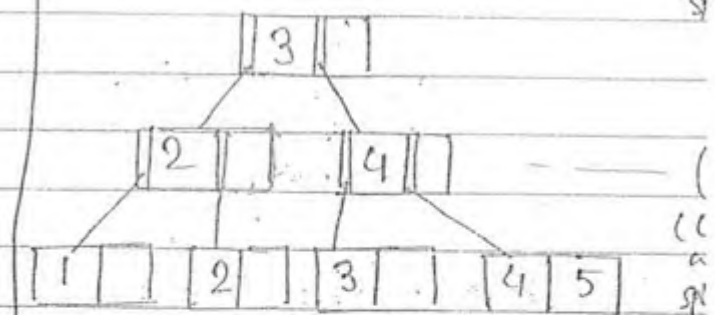
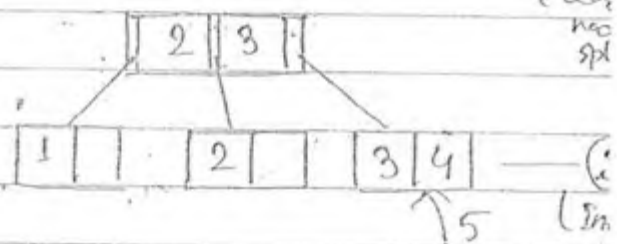
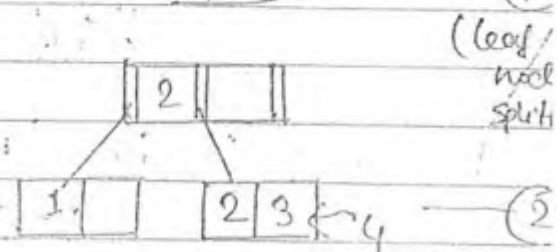
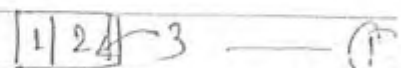
If keys are in Ascending Order

1, 2, 3, 4, 5, 6, ———, N

By Left Biasing. How many node splitting



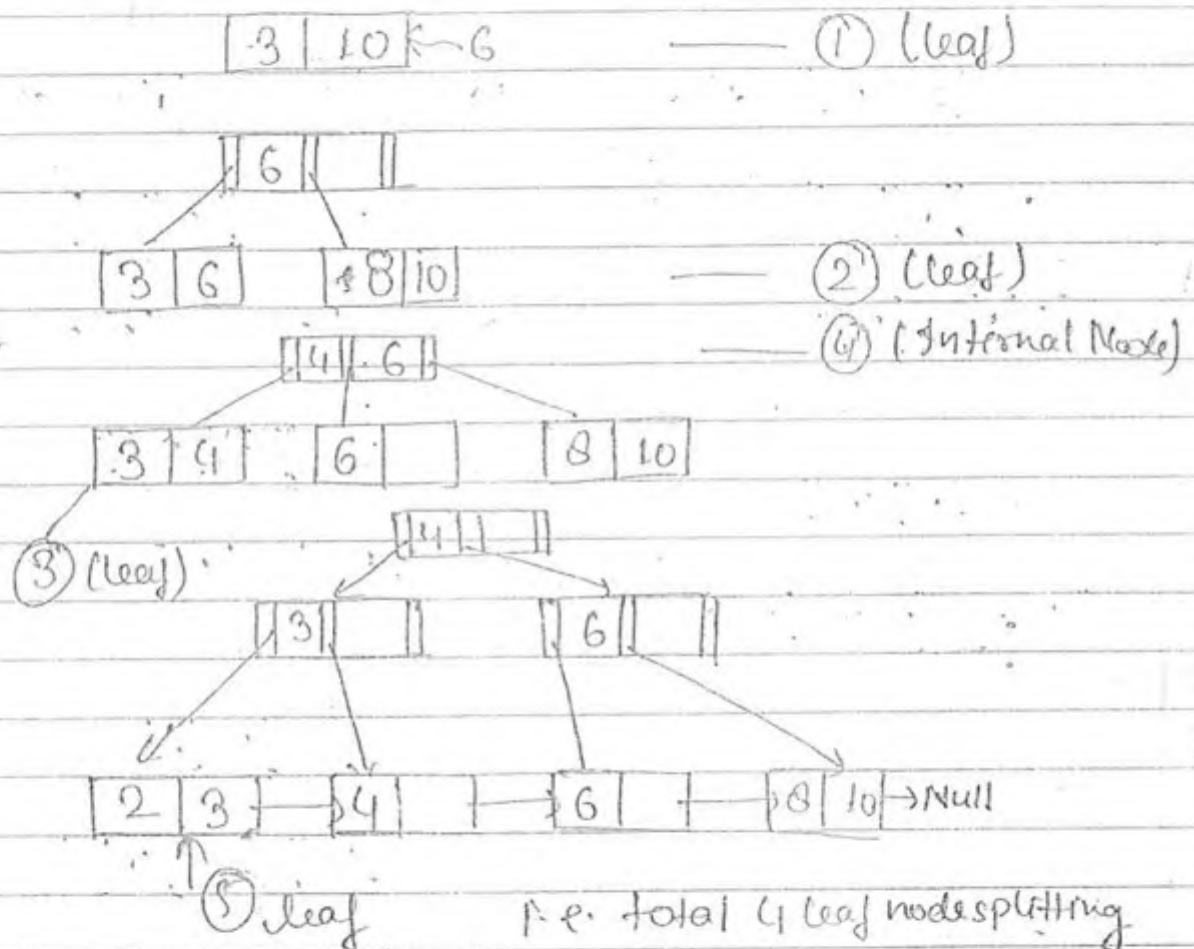
By Right biasing. No. of node splitting.



⇒ If keys are in Ascending Order then Right bias gives the more Node splitting than 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.

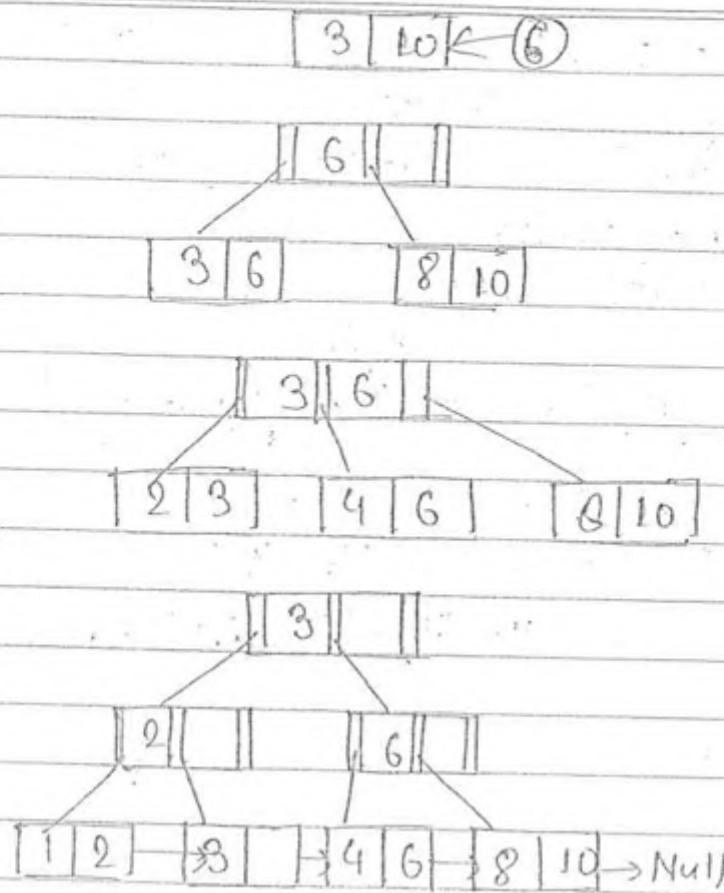
In the above question 10, 3, 6, 8, 4, 2, 1. more than half elements are in descending order. So, we go through left biasing for obtaining maximum no. of node splitting.



⇒ Redistribution always gives less node splitting. So, whenever question asked like minimum no. of node splitting we always go through Redistribution method.

By apply redistribution method —

We get 3 leaf node splitting & 1 internal node splitting.



(Que 5)

Keys = 8 bytes

Block Size = 512 bytes

Index Pointers = 4 bytes

$$P \times 4 + (P-1) \times 8 \leq \text{Block size}$$

$$P \times 4 + (P-1) \times 8 \leq 512$$

$$4P + 8P - 8 \leq 512$$

$$12P \leq 512 + 8$$

$$12P \leq 520$$

$$P \leq \frac{520}{12}$$

$$P = \left\lfloor \frac{520}{12} \right\rfloor$$

Ques 6)

Records = 5,00,000

Record Size = 200 bytes

Search key Size = 15 bytes

 $B_p = R_p = 5$ bytes

Block = 1024 bytes

ORDER of B⁺ Tree! → No. of pointers

Internal Node

Leaf Node

$$P B_p + (P-1) \text{ keys}$$

$$(P-1) \text{ Keys} + (P-1) R_p + 1 B_p$$

$$P(5) + (P-1)15 \leq 1024$$

$$5P + 15P - 15 \leq 1024$$

$$20P \leq 1024 + 15$$

$$20P \leq 1039$$

$$P \leq \frac{1039}{20}$$

Ques 7)

$$P B_p + (P-1) \text{ keys} \leq \text{Block Size}$$

Ques 9)

Leaf Node Stores P!

Maximum no. of (value, data R_p)
pairs it can hold

$$P * \text{Keys} + P * R_p + 1 B_p \leq \text{Block Size}$$

Ques 13) Largest possible order of Nonleaf Node (Internal Node)

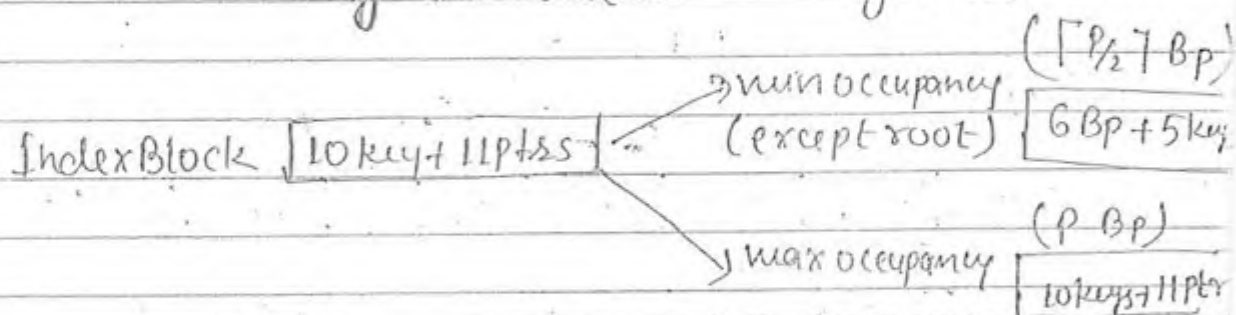
$$P B_p + (P-1) \text{ keys} \leq \text{Block size}$$

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 10000 Records what is the smallest no. of blocks that could be used to store R and a sparse B⁺ Tree Index on the key of R.

5 record / block = 10 keys & 11 pointers

if data block

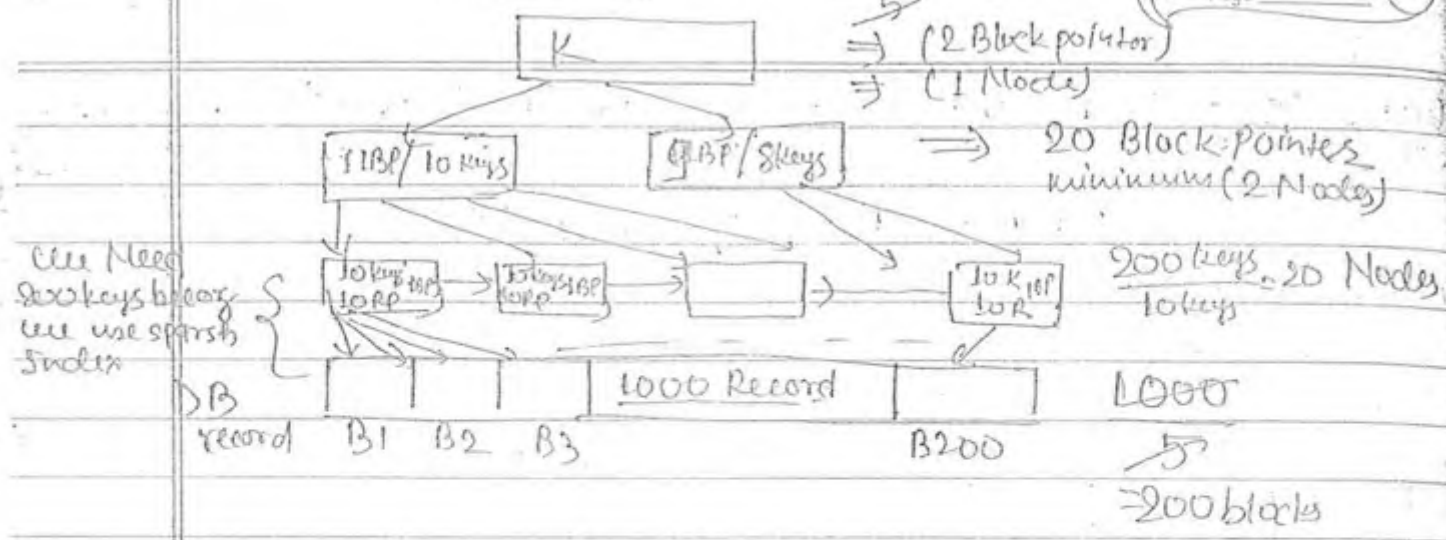
if Index Block



⇒ Smallest No. of Index blocks are possible with only Maximum occupancy per Node

⇒ Largest No. of Index blocks are possible with only minimum occupancy per Node

1000 record
 Total no. of block = $\frac{1000}{5} = 200$ blocks



Total No. of Nodes

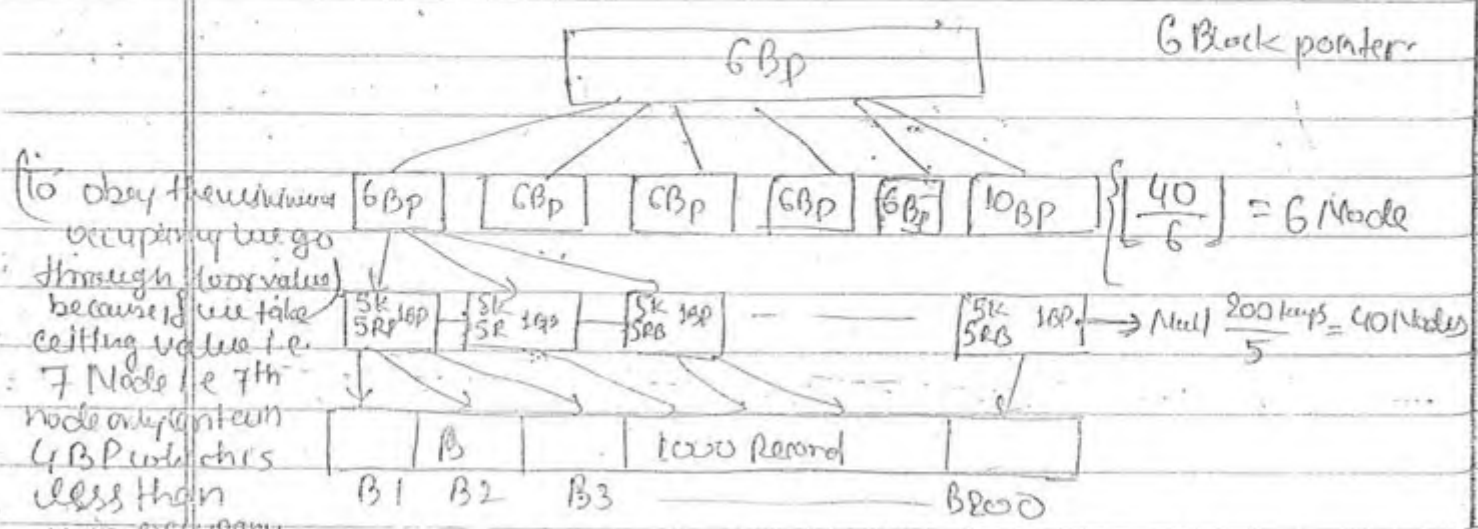
200 + 20 + 2 + 1 = 223

↓ ↓ ↓ ↓

Database 1st Index 2nd Index root level

Largest No. of Blocks + Sparse Index:-

Note:- Minimum B+ Tree Node capacity:



Total No. of Node - 200 + 40 + 6 + 1 = 247

DB file 1st Index 2nd Index root level

Identify Smallest No. of Blocks for Dense Index! -

10 Block Point
= 1 Node.

$$\left\lceil \frac{1000}{10} \right\rceil = 100$$

$$\frac{\text{Loop keys}}{\text{to keys}} = \frac{100}{100}$$

$$\frac{1000}{5} = 200 \text{ Bloce}$$

Identify Largest No. of Blocks for Dense Index! -

TUPLE RELATIONAL CALCULUS:-

⇒ It is one of the Non Procedural Query language.

Nonprocedural Query language:-

Here what to retrieve from Database, we least bothered as how to retrieve from DB.

⇒ First Order Logic formula.

⇒ Formate of the Query $\Rightarrow \{ T / P(T) \}$
(It results Tuples T such that P(T) is true)

Example:- Retrieve the suppliers whose rating greater than 10.

$\{ S / \underbrace{S \in \text{Suppliers}}_{\text{Formula}} (S.\text{rating} > 10) \}$
 ↑
 Free Tuple Variable || (Suppliers)
 rating > 10

Query:- Retrieve the Sid's of suppliers whose rating greater than 10.

$\pi_{\text{sid}} (\sigma_{\text{rating} > 10} (\text{Suppliers}))$
 ||

TR $\{ T / \exists S \in \text{Suppliers} (S.\text{rating} > 10 \wedge T.\text{sid} = S.\text{sid}) \}$

* We can't project bounded type variable only free type variable can be projected.

classmate

Date _____

Page _____

⇒ Bounded Tuple Variable

Tuple variable set to be bounded only if $\exists, \forall$ used before Tuple Variable

$S \in \text{Supplies}$

↑
(Free Tuple Variable)

$\exists S \in \text{Supplies}$

↑
(Bounded Variable)

Query Retrieve Sid's of suppliers who supply some red part.

$\pi_{sid}(\text{Catalog} \bowtie \sigma_{\text{Color} = \text{Red}}(\text{Parts}))$

$\{ T / \exists C \in \text{Catalog} (\exists P \in \text{Parts} (P.\text{Col} = \text{Red} \wedge P.\text{Sid} = C.\text{Sid}) \wedge T.\text{Sid} = C.\text{Sid}) \}$

Query Retrieve Sname of the Suppliers who supplied Red or green part.

$\{ T / \exists S \in \text{Supplies} (\exists C \in \text{Catalog} (C.\text{Sid} = S.\text{Sid} \wedge \exists P \in \text{Parts} ((P.\text{Col} = \text{Red} \vee P.\text{Col} = \text{Green}) \wedge P.\text{Pid} = C.\text{Pid})) \wedge T.\text{Sname} = S.\text{Sname}) \}$

Query Retrieve Sid's of suppliers who supplied Red & green part.

[03]

$\{ T / \exists C \in \text{Catalog} (\exists P \in \text{Parts} ((P.\text{Col} = \text{Red} \wedge P.\text{Col} = \text{Green}) \wedge P.\text{Pid} = S.\text{Pid}) \wedge T.\text{Sid} = C.\text{Sid}) \}$

Above query is wrong because we can't take

$\{ T / \exists C1 \in \text{Catalog} (\exists P1 \in \text{Parts} (P1.\text{color} = \text{RED}) \wedge P1.\text{Pid} = C1.\text{Pid}) \wedge$

$\exists C2 \in \text{Catalog} (\exists P2 \in \text{Parts} (P2.\text{color} = \text{green} \wedge P2.\text{Pid} = C2.\text{Pid}) \wedge (C1.\text{Sid} = C2.\text{Sid}) \wedge$
 $T.\text{Sid} = C1.\text{Sid}) \}$

Query Retrieve Sid of the Suppliers who supplied Some Red part but not green part.

$U : \vee$

$\wedge : \wedge$

$\neg : \neg$

$\rightarrow : \rightarrow$

$\{ T / \exists C1 \in \text{Catalog} (\exists P1 \in \text{Parts} (P1.\text{color} = \text{RED}) \wedge P1.\text{Pid} = C1.\text{Pid}) \wedge$

$\neg \exists C2 \in \text{Catalog} (\exists P2 \in \text{Parts} (P2.\text{color} = \text{GREEN} \wedge P2.\text{Pid} = C2.\text{Pid})) \wedge$
 $T.\text{Sid} = C1.\text{Sid}) \}$

Query1 Retrieve Sid's of the Suppliers who supplied Every Red part.

$\pi_{\text{Sid, Pid}} (\text{Catalog}) / \pi_{\text{Pid}} (\sigma_{\text{color} = \text{Red}} (\text{Parts}))$

$\Rightarrow \forall t (P(t)) \Rightarrow$ Tuples that are for all tuples -
 if t such that $P(t)$ is true.

$\{ T / \exists C2 \in \text{Catalog} (\underset{1}{\forall P \in \text{Parts}} (\underset{2}{\exists C1 \in \text{Catalog}} (\underset{3}{P.Cat = RGD4C1.Pid = P.Pid} \wedge C1.Sid = C2.Sid)) \wedge T.Sid = C2.Sid) \}$

{ For all which is Equal
 to division }

Query)

$\{ S / T (S \in \text{suppliers}) \}$

Free variable.

$\Rightarrow$ O/P! Infinite ! retrieve tuples those are not belongs to suppliers table.

$\Rightarrow$ Unsafe Query. (means results of Query Infinite)

$\{ T / \exists S \in \text{Suppliers} \wedge T.Sid = S.Sid \}$

$\Rightarrow$ O/P! Empty

$\Rightarrow$ Safe Query

* Basic Relation Algebra Expressive Power =
 Safe TRC Expressive Power

$$10^9 \Rightarrow 7.$$

$$(10)^9 \times \log_{10} 10^9 \text{ d} = 10^9 \times \log_{10} 10^9 \times \frac{10}{3600} \text{ ms.}$$

$$= 10^9 \times 9 \times \frac{1}{3600} \Rightarrow 3 \times 10^7$$

$$\textcircled{C} = 3000,000,0$$

~~Data S.T. 10~~

Data STRUCTURE

170