07/10 Friday

Databases

Jagmeet

Data: Known fact abt any ~~object~~ entity
eg: Rno, name

Record: Collection of interrelated data.

Rno	Name	G-no	mark
101	xyz	1	100

— ~~object~~ ~~entity~~ object — ~~passive~~ active
data + fn.
characters + behaviour

— entity - passive
only characteristics
no behaviour.

Database: collection of records

DBMS: s/w to ~~collect~~ create, ~~modify~~ manipulate & delete dB.

DS: DB + DBMS
(dB s/s)
Eg: Oracle, SQL server

DB

OLAP	OLTP
— online analytical processing	— online transaction processing
— historical data	— current data.
— Data warehousing	— D bases.
— decision making	— Date to date optns.
— Terabytes	— MB / GB.
— CIO, MD, CEO, GM	— used by clerks, leads, mgrs.
— R	— R/W

Data Mining —

— DBase
— commercial (inventory, material) (char, number.)
multimedia (data stored as objects)(audio, video)
Deductive (stores rules)
Temporal (time aspect also involved)
Geological Info System DB (Google maps — continous images)
Distributed DB (eg. network dBs.).

13/9/10

Database design

100% Reqments
↓
70% conceptual design - using ER diagram.
↓
20% logical design - Func. depend, SQL, Relation algebra, Relational calculus, Normalization.
↓
5-10% Physical files, Indexes (B, B+)
log ; DB files, storage structure

transaction & concurrency control

→ If users are more than 100, we use indexes for easy access

E-R Diagrams:

- It gives graphical represen of reqmts in terms of entities, relationships and attributes (or)

It is a domain knowledge representation in terms of entities, relationships & attributes.

ER diagram components

a) Entities
b) relationships
c) attributes.

a) Entities:

A real world object or thing with independent existence is known as entities. There are 2 types of

a) physical entity (tangible)
b) conceptual entity. (nontangible)

Physical entity: Person, vehicle, furniture
Conceptual : Sale, course, brand image

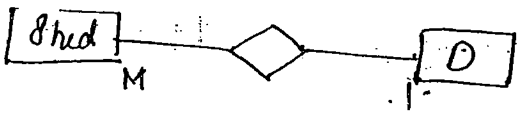
b) Relationships:

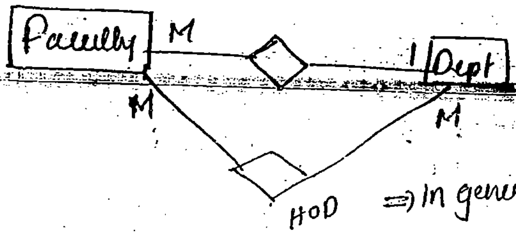
It gives association among entities.

B/w entities, one/more relations are possible

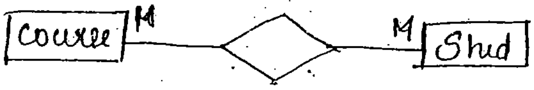
Types of relations

1) one - one (10%)
2) one - many (20%)
3) many - many. (70%)

eg:- 

2) 

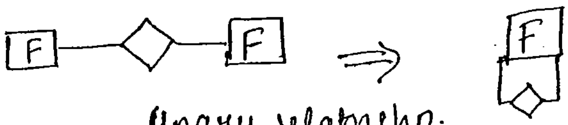
⇒ In general, it is one-one.

3) 

4) 

Degree of relationship:

It specifies no. of entities participating in a relationship.

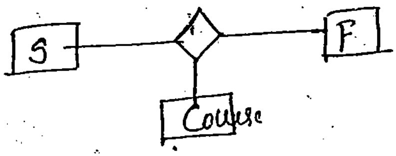
① 

Unary relationship.

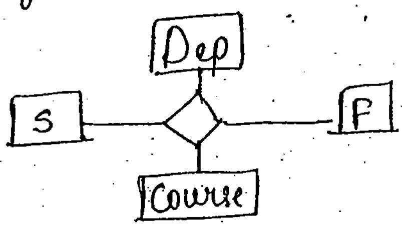
② Binary relationship

Only 2 entities participate. 

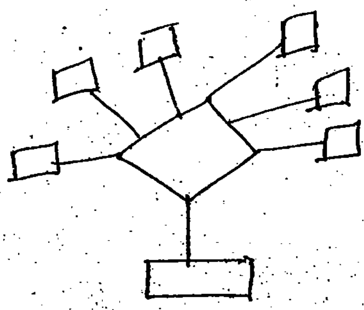
③ Ternary relationship.



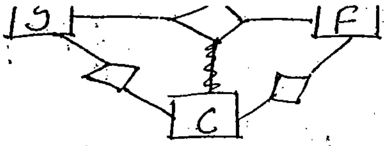
④ Quarternary relation.



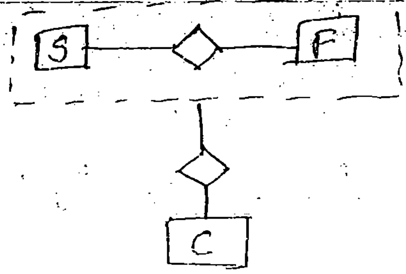
⑤ n-ary



ER diagms are designed to take care of unary & binary relationships, not other type of relationships.

Ternary ⇒ ① 

(OR)



Constraints on ER digm:

① Structural constraints

a) participation constraints:

b) cardinality ratio.

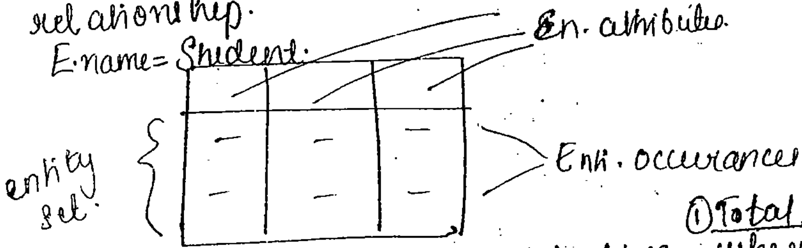
② Covering constraints

③ Overlapped constraints

② and ③ is used in EER (ER + OO)

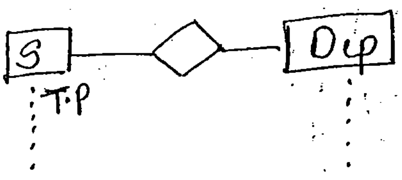
Participation constraints:

It defns participation of entity occurances thru a relationship.



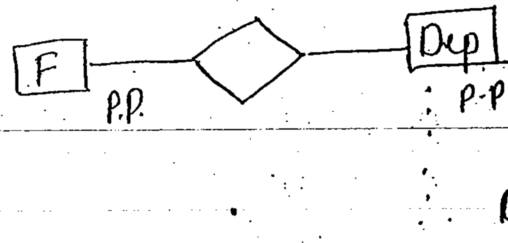
There are 2 types of participation ① Total: where all entity occurances participating through a relationship.

② Partial: Some of entities are not participating thru a relationship

① 

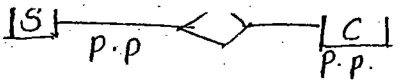
All students must have dept

But all dept need not contain student

② 

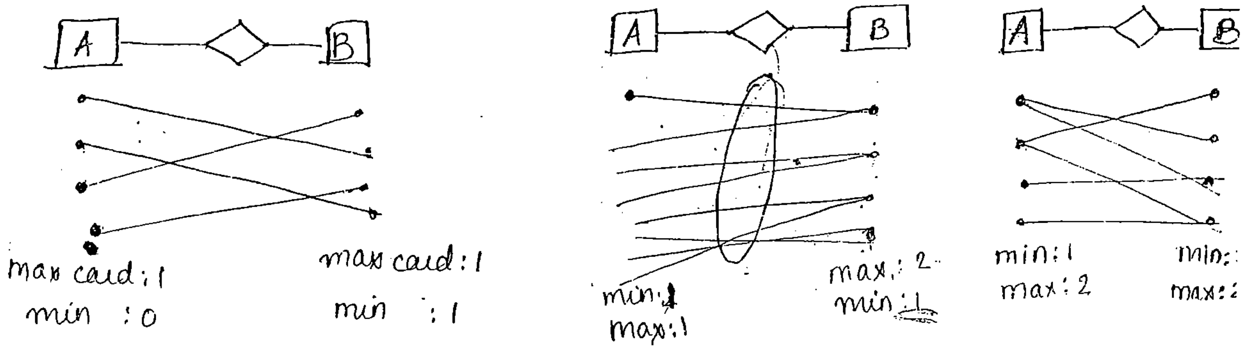
all students participate but depts donot.

Dept may not contain a faculty. (P.P)



Cardinality ratio:

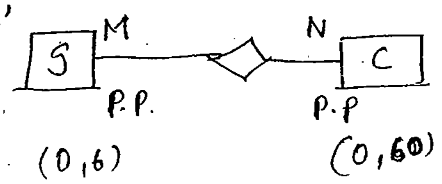
~~defn. maximum no. of times an entity occurrence~~ participating in a relationship.



min cardinality : If 0 ⇒ partial participation.
If 1 ⇒ Total participation

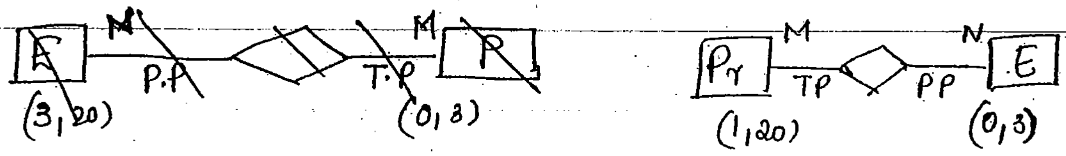
max cardinality : If 1 ⇒ entity occurance is participating thru a relation only once.
If N ⇒ then N no. of times an entity occuranc. participate in a relation.

(1) Consider,



Description: All the students need not register courses but they can go max upto 6.
A course can allow max of 60 students to register, but all the courses need not have registration.

(2) Consider: A project suppose to have min 3 employees and maxm of 20 employees. All the employees need not be in proj', but they can participate upto 3 project at a time.



but (max 3 proj') cannot be represented.

III Attributes:

3 categories:

a) simple and composite attributes

~~b) single valued & multivalued attr.~~

c) Stored and derived attr.

⇒ multivalued causes sm probs

a) Simple – atomic values and cant be divided further.

Roll no, age

Composite – can be divided further into simple attributes

Addr — HNo, Streetname, City, State

Name — firstname, Lastname

b) Single valued : attr that holds single value is called single valued. Eg: PANcard no, blood g, voterID.

Multivalued : holds multiple values.

eg: address, teleph no, emailIDs

c) Stored : supplies values to derived attribute.

age

Derived : get value from stored attributes.

① DOB ← age.

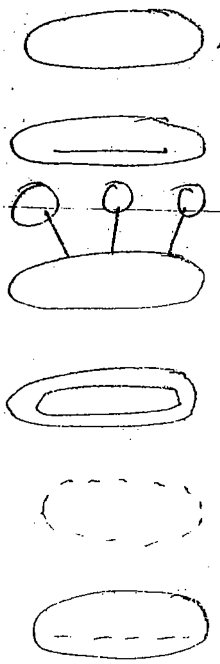
② RNo. BNo I/date R/date fine defnd from

③

RNo.	Name	BNo
101	a	1
102	b	1
102	c	2

Bino	Bname	#
1	CSE	2
2	IT	1

instead of many entering, # can be derived from 1st table.



Attr.

1° key attr

CA (composite)

Multi var. attr.

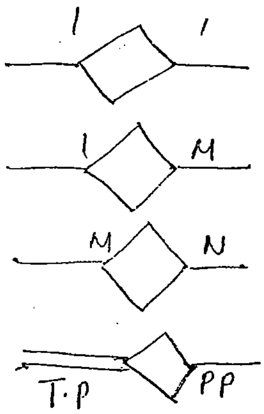
DA

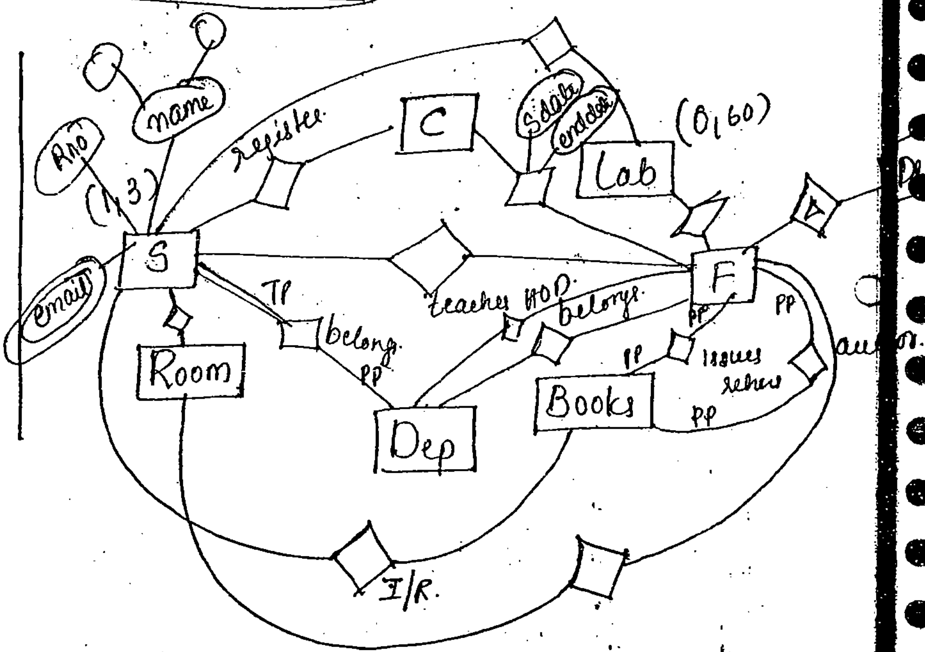
discriminated ~~designated~~ attr.

An attr/attrs in weak entity that are combined with primary key of S.E to declare the combination as a 1° key for the W.E is known as (- - - -)

1° key ← F.key

Pno Oname Dage





Participation: ═ TP
— PP.

Cardinality ratio — (0,60) (1,3) ....

- Another classification of entities
  - Strong entity
  - Weak entity
  - Associative entity.

1° key

FNo	FName
101	a
102	b
103	c

Dependent table.

FNo	Depname	age
101	X	5
101	X	60
102	Y	10

when 102 is removed, its dependent table entry also removd.

1° key

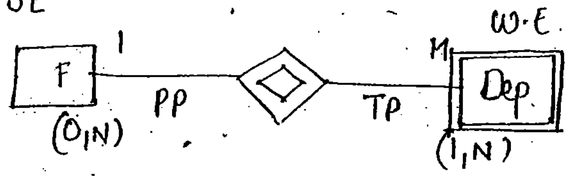
PNo	Pname
101	a
102	b
103	c

Strong entity

1° key

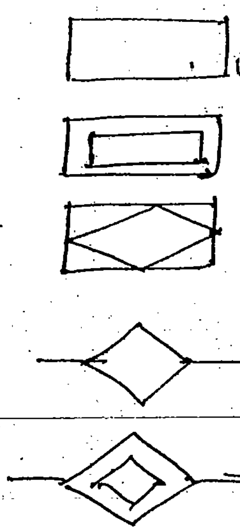
PNo	LNo	Ltype
101	1A1	LMV
101	1B1	IDL
102	1L1	HMV

strong. entity.



E-R Diagram notations:

- Chen
- Crow's feet → UML
- Idefix ✗
- Rein5 ✗



SE

WE

AE looks like relationship but actually its entity.

Relation

Identifyg relation

* Conversion of ER diagrams to tables:

Step 1: Conversion of strong entities

a) for each strong entity create a separate table with same name

b) Include all attrs, if there is any composite attr. split it into simple attr and include them. Ignore multivalued attr at this stage

c) Select 1° key for the table.

Step 2: Conversion of weak entity.

a) For each weak entity create a separate table with same name.

b) Same as step (1)

c) ~~Create~~ Include 1° key of strong entity as foreign key in the weak entity.

d) Declare the combination of foreign key & discriminator attributes as 1° key for the weak entity.

Step 3: Conversion of one to one relationship.

a) For each one to one relation, say A and B, modify either A side or B side to include 1° key of other side as a foreign key.

b) If A or B is having total participation, then that shd be the modified table.

if both PP, either A/B

c) If relationship consists attributes include them also in the modified table.

Step 4: Conversion of one to many relationship

a) For each one to many relationship, modify M side to include 1° key of one side as a foreign key.

b) If relationship consists attr, include them also

Step 5: Conversion of many-many relationship

(a) For each many-many, create a separate table & to include 1° keys of M side and N side as foreign keys in the new table.

(b) Declare the combination of foreign keys as 1° key for new table.

(c) If relationship consists attr, include them also in the new table

Step 6: Conversion of multivalued attr:

a) For each multivalued attr, create a separate table and include 1° key of parent table as foreign key.

b) Declare the combination of foreign key & Multi valued attr as 1° key.

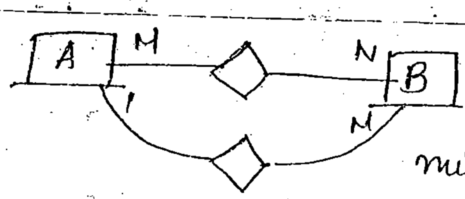
Step 7: Conversion of n-ary relationships.

a) For each n-ary relationship, create a separate table & include 1° keys of all entities as foreign keys.

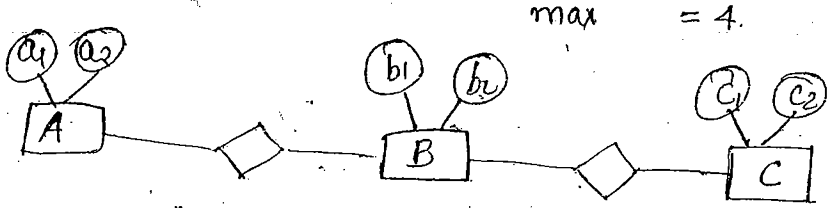
b) Declare the combination of foreign keys as 1° key.

• Note:

1—1 and 1—M relations can also be separated as tables but its not adviseable due to performance reasons.



min table = 3
max = 4.

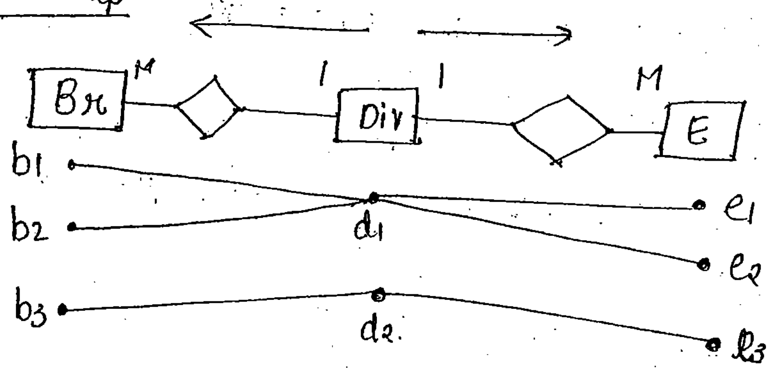


a) a_1 a_2 b_1
b) b_1 b_2 c_1
c) a_1 a_2 c_1 ✓
d) a_1 a_2 b_2

Trap

1) FAN trap
2) CHASM trap.

FAN trap



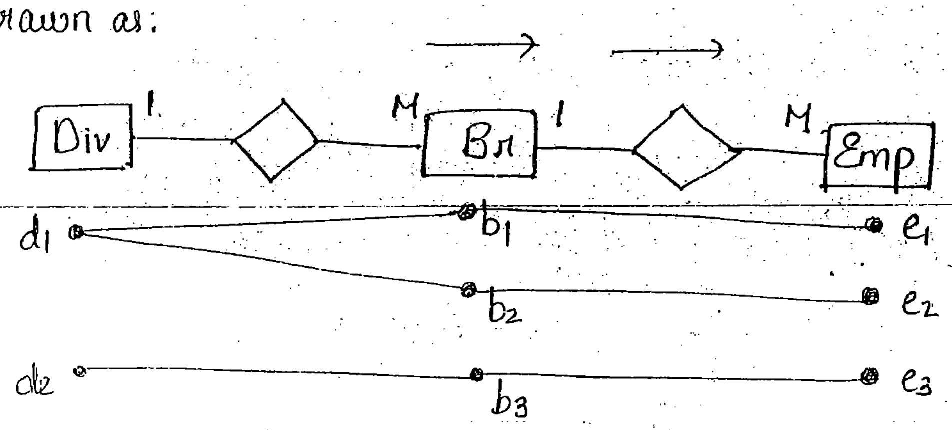
we cant say whether e_1 belong to b_1 or b_2. Since d_1 has both.

If this E-R diagm is converted into tables, we are able to retrieve all info except e_1 belongs to which branch(b_1/b_2). This is due to FAN trap.

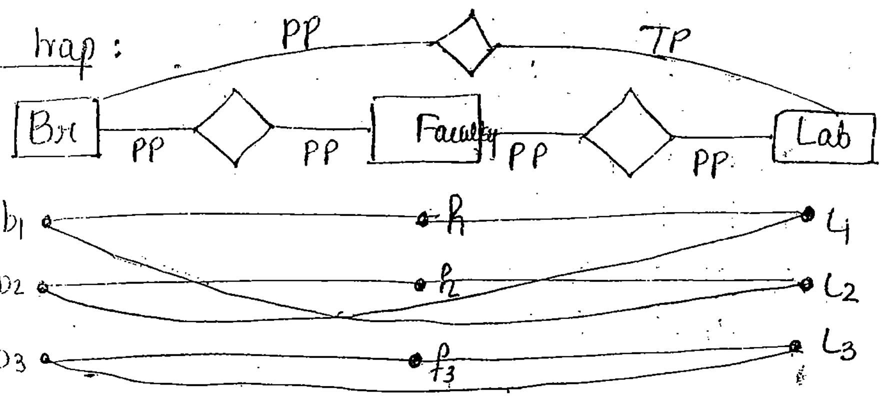
How to identify FAN trap?

If two 1—M relationships are emerging out from single entity, then there will be a FAN trap.

Redrawn as:



CHASM trap:



If we convert this ER diagm into tables we face the 2 probs:

1) we are unable to identify branch details of ~~branch~~ lab 2.
2) it shows lab1 belongs to branch1 since f1 is operating this lab — may/may nt be correct.

This is due to CHASM trap.

How to identify?

If two directly related entities are conntd thru another entity with partial participatn then there is a CHASM trap.

How to eliminate?

Create a direct relationship b/w these 2 entities

Advantages of ER diagrams

1) It is an effective communication tool amg database designer, domain expert & stakeholders.

2) It is ~~stio~~ tightly integrated with relational dB model.

3) It is easy to understand.

7

Disadvantages:

1) loss of information content.
2) limited constraint representation.
3) it is overly complex for small projects.

Various dB design concepts:

1) Top down design
2) Bottom up design
3) Mixed design
4) Inside-Outside design

top down	Bottom up	Mixed	in/out
Entity	all attr	E	1:1
Relation	relation b/w attr.	R	1:M
Attr	Entities	A	M:N
ER	FD, normalization	ER	attributes
↓	↓	↓	
tables.	tables.	tables	
— big prjt	—small projt	↓	
— X	— purful	FD, norm	
— easy	—difficult.	↓	
		tables.	

Defn: Functional Dependency:

It defns association amg entities.

RNo, name, course no, vehi no .... Course credit (cc)

RNo → name
name ↛ Rno.

RNo → CNo.

39 **Q.2.**

A → B
B ↛ C
A ↛ C
AC → B
AB ↛ C

RNo → BrNo
~~R,no → BrNo.~~

	A	B
t_1	101	A
t_2	102 (101)	B

If t_1 & t_2 agree then t_1 & t_2 must also agree.

if t_1 & t_2 disagree t_1 & t_2 may/not agree.

39 3) $A \to B$
$B \to C$
$C \not\to A$

~~$AC \to B$~~
$AB \to C$
$BC \to A$

NM
DS
TOC

39 4) $B \to C$ ✓ solve 5th HW.

Characteristics of func. dependancies:

FDs must hold always, ∴ they shd be defnd on schema not on ~~depen~~ instances.

RNo	Name	age

$A \to B$.

A	B	C
1	2	3
4	5	6
7	8	5
3	4	6

Rno → name
age → Rno.

3 4 6
3 5 6 X not possible

2) deals with 1–1 relationship.

3) It shd be non ~~tri~~ trivial or completely non trivial.

$AB \to CD$ — completely nontrivial
$AB \to BC$ — non trivial
some attr are subset of left.
$ABC \to BC$ — trivial
all attr of rt are subset of left.

Req
$\downarrow$
F_1 $\quad A \to B$
$\quad B \to C$
$\downarrow$
F_2 (additional FDs) $A \to C$

$F = F_1 + F_2$

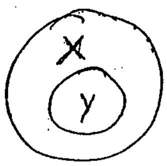
From the reqments, once FDs are identified ~~& calls~~ call it as F_1; From F_1 addtnl fn depen. can be identified

Total func. dependancies $= F_1 + F_2$.

This shd be i/p for normalisation process.

To identify F_2 we use following two methods, ~~inference~~

1) inference rules
2) closure set of attr

① If $y \subset x$  $\Rightarrow$ $x \to y$

② $x \to y$
$y \to z$
$\underline{x \to z}$.

③ $x \to yz$
thn, $x \to y$
$x \to z$ decomposition.

④ $x \to y$
$xz \to yz$

⑤ $x \to y$
$x \to z$ augmentation.
$x \to yz$

Eg:- Req
$\downarrow$
$F_1 =$ $A \to B$
$B \to C$
$C \to DE$ — $C \to D$, $C \to E$
$E \to F$

F_2 : $A \to C$
$B \to DE$
$C \to F$
⋮
X

we don use inference rules to identify addn. FDs but it has 2 limitations.

① time consuming.

② error prone.

② ~~Closure set method~~

A^+

$X = A$
$= AB$
$= ABC$
$= ABCDE$
$= ABCDEF$

C^+

$X = C$
$= CDE$
$= CDEF$

Algorithm to identify closure set of attributes:

① Equate an attribute or attributes to X for which closure needs to be identified.

② Repeatedly take func. dependencies one by one and check whether left hand side attr is available or not. If available, add rt. hand side attr of func dependency to X.

③ Repeat step 2 as many times as possible to cover all ~~possble~~ FDs.

④ Stop the process if no more attributes can be added to X.

Pg. AO

6. $A \to B$ ✓
$BC \to DE$ ✓
9. $AEG \to G$ ✗
Compute AC^+

$X = AC$
$= ABC$
$= ABCDE$
$AC^+ = ABCDE$

i.e. $AC \to BDE$

(repeatedly go for 1 or 2 cycle)

→ determine

7. $A \to BC$ × | × $X = B$
$CD \to E$ × | × $= BD$
$B \to D$ ✓ | ×
$E \to A$ × | ×
$B^+ = ?$

9

8 HW.

Applns of closure set of attr

1. To identify addnl func dependancies.
2. To identify equivalences.
3. To identify candidate keys.
4. To identify irreducible set of FDs or canonical form of FDs. (std form of FDs)

PgA]

9. $A = ABCDEFGHI$

$A \to BC$ ✓
$CD \to E$
$E \to C$ ✓
$D \to AEH$ ✓
$ABH \to BD$ ✓
$DH \to BC$

$BCD \to H \Rightarrow ?$

BCD^+

$X = BCD$
$= BCDE$
$= ABCDEH$
$= ABCDEH$ ⟶ hence posble.

$BCD^+ \to A\cancel{B}\cancel{C}\cancel{D}EH$
$\to AEH$

ie,
$BCD \to A$
$BCD \to E$
$BCD \to H$ ✓ hence posble

$AED \rightarrow H$? True.

$BH \rightarrow AC$ X

BH^+ X $= BH$

$BH^+ = BH$

$BH \rightarrow \underline{BH}$

Reqm

DB1	DB2
$A \rightarrow B$	$A \rightarrow B$
$C \rightarrow HD$	$D \rightarrow BC$
$F \rightarrow E$	$E \rightarrow AF$

From the same reqm, database designers might come with diff sets of F.Ds. B4 evaluating which is wrong/right, it is necessary to chk whether both are equivalent or not. For this purpose, we use closure set of attributes.

3A/10) F = $A \rightarrow C$ G = $A \rightarrow CD$

$AC \rightarrow D$ $E \rightarrow AH$

$E \rightarrow AD$

$E \rightarrow H$

Take F set and verify all its FDs can be ~~deto~~ derivd from G or not

$A \rightarrow C$ compute A^+ from G.

$AC \rightarrow D$ compute AC^+ from G.

$E \rightarrow AD$ compute E^+ from G.

Take 'G' set and verify all its FD can be derived from F/no

$A \to CD$ compute A^+ from F ✓ A^+ —
$E \to AH$ E^+ from F ✓

11. $F = $ $B \to CD$
$AD \to E$ ✓
$B \to A$ ✓

$G = $ $B \to CDE$
$B \to ABC$
$AD \to E$

F: $X = B$
$= BCD$
$= ABCD$
$= ABCDE$

G: $X = B$
$= BCDE$
$= ABCDE$

both are equivalent.

Appln 3. Types of keys

1) Primary key
2) Composite primary key.
3) Candidate key
4) Super key.
5) Surrogate key
6) Foreign key.

Primary key: - unique value column
- not null column
- only 1 1° key per table
- enforces entity integrity.

Composite 1° key: - 1° key with 2 or more attr.
- mainly in transaction table.

to uniquely identify

FNo	CNo (course)	S/date	E/date
101	10		
101	11		
102	12		

Candidate key : If we have onemore unique value ~~cols~~ columns in a table, thn 1 of them can be elected as 1° key for a table. and these alternative keys are called candidate keys.

(or)

VNo	Engine No	DOP

R = ABCDEF

A → BCD

E → CD

F → ABC

A
E
F

AE
AF
EF
AEF

Superkey – union of all lefthand side attr in FD.

Surrogate key

If we are unable to identify 1° key from existing cols of table, we look for surrogate key.

App Id	RNo	BNo	S/date	E/date
1	101	1	1/1/2010	15/1/2010
2	101	2		
3	102	1	16/1/10	20/1/2010
4	101	1	21/1/2010	21/1/10
5	101	1	21/1/2010	21/1/2010.

Q/10 Foreign key: – used to implement referential integrity.

– can be a null column

– we can have more than 1 foreign key in a table.

– Foreign key shd refer always primary key either in its own table or in some other table.

F key → Prim key

RNo	Name	Brno
1	a	101
2	b	101
3	c	102.
4	d	~~105~~ 103

Brno	Bname
101	CSE
102	IT
103	ECE

↳ Facultyno Fname HOD F.key.

P.key

Facultyno	Fname	HOD
1	a	2
2	b	—
3	c	—
4	d	3
5	e	3

since they are HODs

G.Q. ⟹ how many addnl rows have to be removed if 3,1 is removed.

A	B
1	—
2	1
~~3~~	~~1~~
4	1
×5	3
6	5 ×
× 7	3
8	6 ×

Totally 5 rows have to be removed.

⟹ Eg:-

$A \to B$

$C \to D$

$F \to CH$

If any closure can identify all attr then it is key.

$A^+ = ABCDH$

$A \to \underline{\underline{BCDH}}$

12. $R = ABCDEH$

$A \to BC$

$CD \to E$

$E \to C$

$D \to AEH$

$ABH \to BD$

$DH \to BC$

Single LHS first, find closure.

$A^+ = ABC$

$E^+ = EC$

$D^+ \to DAEHBC$

∴ D is a key.

AD - Superkey.

CD -

BD -

(13) R = ABCDE

$A \to B$ | $A^+ = AB$
$BC \to E$ | $BC^+ = BCE$
$ED \to A$ | $ED^+ = EDAB$

?

$AB^+ = AB$

$BD^+ = BD$

$AD^+ = ADB$

$BCD^+ = BCDEA$ (key)

$ACD^+ = ACDBE$ (key)

ACD is the key.

Note:

ABCDE

$A \to BDE$
$C \to E$
$D \to BC$

A ✓ C ✗ D ✗ AC ✓ CD ✗ AD ✗ ACD ✓

3 keys

(19) $ABD \to E$

$AB \to G$

$B \to F$

$C \to J$

$CJ \to I$

$G \to H$

$B^+ = BF$

$C^+ = CJI$

now 2 attr;

$AB^+ = ABGFH$ includes majority of attr..

$CJ = CJI$

? ?

$ABC^+ = ABGFHCJI$

$ABD^+ = ABGFHDE$

$ABCD^+ = ABCGHFEDI$

$\downarrow$
f_1
$\downarrow$
f_2

$F = F_1 + F_2$.

(10)

$A \to B$
$B\cancel{X} \to D$
$E \to F\cancel{X}$
$G \to H$ ×

$\Rightarrow$

$A \to B$
$B \to D$
$E \to F$

Total func dependency $F = F_1 + F_2$. where F_1 is derived from reqm. and F_2 is derived from F_1. F is i/p for normalization process.

Before making a move to normalization, we have to evaluate the following; ~~ie~~.

1) redundant func. dependency.
2) redundant left hand attr.
3) redundant right hand attr.

20) R = ABCD

$A \to B$
$C \to B$
$D \to ABC$
$AC \to D$

Step 1 make RHS single

1. $A \to B$ ✓
2. $C \to B$ ✓
3. $D \to A$ ✓
4. $D \to B$ ×
5. $D \to C$ ✓
6. $AC \to D$ ✓

Step 2

remove (1), compute A^+ from 2, 3, 4, 5, 6
$A^+ = A$

remove (2), compute C^+ from 1, 3, 4, 5, 6
$C^+ = C$

remov (3), compute D^+ from 1, 2, 4, 5, 6
$D^+ = DBC$

remov. (4), compute D^+ from 1, 2, 3, 5, 6
$D^+ = DCBA$

remove (5), compute D^+ from 1, 2, 3, 4, 6.
$D^+ = DBA$

~~remove (6), compute AC^+ from 1, 2, 3, 4, 5~~
$AC^+ = ABC$

irreducible set

$\Rightarrow$
$A \to B$
$C \to B$
$D \to A$
$D \to C$
$AC \to D$

Step 3. remove A

$A \to B$	$A \to B$
$C \to B$	$C \to B$
~~$D \to A$~~	~~$D \to A$~~
$D \to C$	$D \to C$
$AC \to D$	$\cancel{A}C \to D$

redundant left

$C^+ = CB$ $\quad$ $C^+ = CDAB$

not equivalent - so cant remove A.

remove 'C'

$A \to B$	$A \to B$
$C \to B$	$C \to B$
$D \to A$	$D \to A$
$D \to C$	$D \to C$
$AC \to D$	$A\cancel{C} \to D$

$A^+ = AB \neq A^+ = ABCD$

cant remove C

Step 4 apply union rules.

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

2. Q.21

$AB \to C$
$C \to B$
$A \to B$

Step 1
single attr on rt
(1) $AB = C$
(2) $C \to B$
(3) $A \to B$

Step 2:
Remove (1), compute AB^+ from 2, 3
$AB^+ = AB$
Remove (2), compute C^+ from 1, 3.
$C^+ = C$
Remove (3), compute A^+ from 1, 2
$A^+ = A$.

1Q8.

Step 3

	Remove A
$\cancel{A}B \to C$	$B \to C$
$C \to B$	$C \to B$
$A \to B$	$A \to B$

$B^+ = B \neq B^+ = BC$
cant remove A

remove B

$AB \to C$	$A \to C$
$C \to B$	$C \to B$
$A \to B$	$A \to B$

$A^+ = CAB = A^+ = ABC$

$A \to C$
$C \to B$
$A \to B$ ✗

Remove ① & compute A^+ from 2, 3
$A^+ = AB$

~~remove ② compute C^+ from 1, 3~~
$C^+ = C$

remove ③ compute A^+
$A^+ = ACB$

Step 4

$$\boxed{\begin{matrix} A \to C \\ C \to B \end{matrix}} \not\equiv \boxed{\begin{matrix} AB \to C \\ C \to B \\ A \to B \end{matrix}}$$

Question ① { To chk ans. correct → equivalence chk.
left hand side removal – step 3.
rt hand side " – step 2.

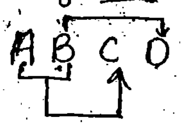
Classification of FDs

Partial FDs
Transitive FDs
Full func. dependency

1) Partial

All non key attributes must depend totally on primary key attributes.

$R = ABCD$
$AB \to C$
$B \to D$ (partialy dependant on AB.)
key: **AB**



Note: Under the following circumstances a table cannot have partial dependencies:

~~a) If 1° key consists single attribute.~~
b) If table has only 2 attr.
c) If all attr in a table are part of primary key.

Transitive ~~rel~~

If there is a relation amg nonkey attr, then it is transitive dependency.

R = ABCD

AB → C

C → D

Key : AB

AB C D

here AB shd have identified D
but C → D. not AB → D.

Under the followg circumstances, no transitive

a) If table consists only 2 attr.

b) If all attr in a table is part of prim key.

ABCD

AB → C

C → D, (Trans)

D → B ⟶ non key attr identifies B.

BC → D

AB C D

Full functional

If there is a dependency of the form X → Y then the removal of attr / attrs from X makes X → Y invalid.

Normalization Tool

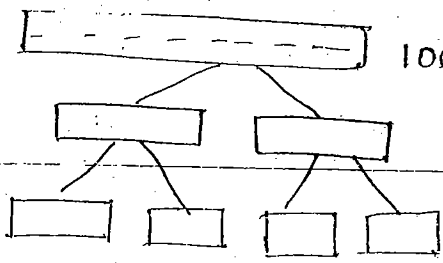
It is an evaluation tool ~~to~~ evaluate logic database design while insertion, deletion & modification problems.

It is a process of reducing redundancy by eliminating ins, del & mod problems.

1NF 100%

2NF 50%

3NF 25%



BCNF

4NF

5NF

DK/NF - - - - - - - - 1%

Classification of dependencies:

FD	MVD	JD	ID	TD
functional	multivalued	Join	Inclusion	Template.
2NF	4NF	5NF		
3NF				
BCNF				

Read: insertion, deletion, updation anomalies

Pg.17 Refer Table

F-no Fname Q Dname HOD Strength

FNo (p. key)	Fname	Q	Dn
-	-	-	ECE
2	-	-	ECE
-	-	-	ECE

Dnam (p. key.)	HOD	Strength
EIE	-	60
ECE	~~Ram~~ Sultan	60

7.24 Normal Forms

2NF: A table is in 2NF, if its already in 1NF and shd be free from partial func. dependencies.

Pg. A3 25. F: $AB \to C$ $R = ABCDEFGHIJ$

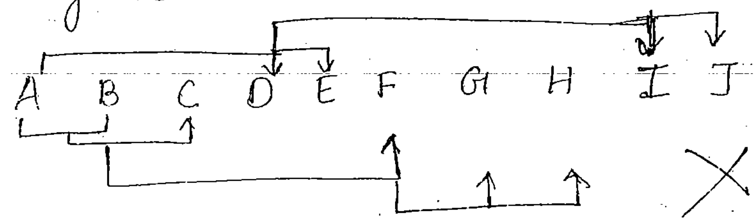
$A \to DE$ (PD)

$B \to F$ (PD)

~~$F \to GH$~~

$D \to IJ$

(a) Key: AB.



use closure method

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

$B^+ = BF\underline{GH}$

2NF

$$\boxed{\begin{array}{l} R_1 = \underline{A}\ DEIJ \\ R_2 = \underline{B}\ FGH \\ R_3 = \underline{AB}\ C \end{array}}$$

Note:

If there is a partial dependency remove partially dependent attr from original table along with copy of its determinant.

40) $A \to B$

$B \to C$

$C \to D$

key: A

a) 2NF

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

b) 3NF

$$\boxed{\begin{array}{l} \underline{C}D \\ \underline{B}C \\ \underline{A}B \end{array}}$$

c) 3NF = BCNF

Pg.42 FD Normal Form

15

22. ABD → AC
C → BE
AD → BF
B → E

Step 1: i.
ABD → A ✗
ABD → C ✓
C → B, C → E ✗
AD → B ✓
AD → F ✓
B → E ✓

Step 2:
① $ABD^+ = ABDCEF$
② $ABD^+ = ABDEF$ (C missing)
③ $C^+ = CE$ (no B)
④ $C^+ = CBE$ (E present)
⑤ $AD^+ = ADF$ (no B)
⑥ $AD^+ = ADBC$ (no F)
⑦ $B^+ = B$ (no E)

ii. ABD → C
C → B
AD → B
AD → F
B → E

Remove A

ABD → C C → B AD → B AD → F B → E $BD^+ = BDE$ $AD^+ = ABCDFE$ $D^+ = D$.	BD → C C → B AD → B AD → F B → E $BD^+ = BDCE$ not equal	remove B AD → C C → B AD → B AD → F B → E $AD^+ = ADCBEF$ equal hence B can be removed.	ABD → C C → B D → B D → F B → E $D^+ = DBF$ cant removeA cant removeD

iii. AD → C
C → B
AD → F
B → E
AD → B

Union. AD → CFB
C → B
B → E

23) $A \to BC$

$ABE \to CDGH$

$C \to GD$

$D \to G$

$E \to F$

Step 1:

$A \to B$
$A \to C$
$ABE \to C$ ✗
$ABE \to D$ ✗
$ABE \to G$ ✗
$ABE \to H$ ✓
$C \to G$ ✗
$C \to D$ ✓
$D \to G$ ✓
$E \to F$ ✓

Step 2:

① $A^+ = ACGDG$ (no B)
② $A^+ = AB$ no C
③ $ABE^+ = ABEDGHFC$ (all)
④ $ABE^+ = ABECGHDF$ (all)
⑤ $ABE^+ = ABECDHGF$ (all)
⑥ $ABE^+ = ABECDGF$ (no H)
⑦ $C^+ = CDG$
⑧ $C^+ = CG$ no D
⑨ $D^+ = D$ no G
⑩ $E^+ = E$ no F

ie. $A \to B$
$A \to C$
$ABE \to H$
$C \to D$
$D \to G$
$E \to F$

$BE^+ = BEF$

$AE^+ = AEBCF$

$AB^+ = ABCDG$

Step 3:

remove A

$BE^+ = BEHF$

cnt remove A

remove B

$AE^+ = AEBCF$

equal.

remove E

$AB^+ = ABCDGH$

not equal

ie. $A \to B$
$A \to C$
$AE \to H$
$C \to D$
$D \to G$
$E \to F$

Union

$A \to BC$

$C \to D$

$D \to G$

$E \to F$

$AE \to H$

(2)(16)

24. $BCD \to A$ ✓
$BC \to E$ ✓
$A \to F$ ✓
~~$F \to G$~~ ✓
$C \to D$ ✓
$A \to G$ ✗

Step I satisfied.

step 2.
$BCD^+ = BCDE$ no A
$BC^+ = BCDAGF$ no E
~~$A^+ = AG$ no F~~
$F^+ = F$ no G.
$C^+ = C$ no D
$A^+ = AFG$ (G present)

iii. $BCD \to A$
$BC \to E$
$A \to F$
$F \to G$
$C \to D$

$BC^+ = BCEDAFG$
$BD^+ = BD$
$CD^+ = CD$

remove D:
$BC \to BCAEFGD$ not equal.
remv C:
$BD^+ = BDAFG$
remv B:
$CD = C$

cnt remove:

$\boxed{\begin{array}{c} \cancel{BCD \to A} \\ BC \to E \\ A \to F \\ F \to G \\ C \to D \end{array}}$

$BC \to A$
$BC \to E$
$A \to F$
$F \to G$
$C \to D$

$\equiv$

$\boxed{\begin{array}{c} BC \to AE \\ A \to F \\ F \to G \\ C \to D \end{array}}$

30) $A \to B$
$BC \to E$
$ED \to A$

Key : ACD
$A^+ = AB$
$C^+ = C$
$D^+ = D$

$\Rightarrow$ 2NF $\boxed{\begin{array}{c} \underline{ACD}\ E \\ \underline{A}\ B \end{array}}$ no transitive dependency so in 3NF also.

BCNF
A ✓ ACD ✓
BC ✗ A ✓
ED ✗ $BC \to E$
$\quad ED \to A$

$\boxed{\begin{array}{c} ACD \\ AB \\ AEC \\ BCE \end{array}}$

34) $A \to BC$
$ABE \to CDGH$
$C \to GD$
$D \to G$
$E \to F$

Key: AE

$AE^+ = ABCDEFGH$

b) 2NF
$A^+ = ABCGD$
$E^+ = EF$

ABCGD
EF
AEH.

2NF

c) 2 trans relations

ABC
DG
CGD
EF
AEH

BCNF

A	A ✓
Superkey ABE	D ✓
C	C ✓
D	E ✓
E	AE ✓

BCNF

ABC
DG
CGD
EF
AEH

36) R = ABCDE
$AB \to CDE$
$A \to C$
$D \to E$

: AB

b) 2NF
$A^+ = CA$
$B^+ = B$

AC
ABDE

2NF

c)

AC
ABD
DE

3NF

BCNF

AB	A
A	AB
D	D

∴ also BCNF.

37) $AB \to CDE$
$A \to C$
$C \to D$

ABCDE

key: AB

b) $A^+ = ACD$
$B^+ = B$

ACD
ABE

2NF.

c)

AC
CD
ABE

3NF

3NF = BCNF

d). BCNF:

AB	A
A	C
C	AB

26. $R = ABCDEF$

$A \to FC$

$C \to D$

$B \to E$

(a) Key: AB

(b) $A^+ = AFCD$

$B^+ = BE$

$AB\cancel{C}\cancel{D}E\cancel{F}$

$$\boxed{\begin{array}{l} R_1 = ACDF \\ R_2 = BE \\ R_3 = AB \end{array}}$$

27. $R = ABCDE$

$B \to E$

$C \to D$

$A \to B$

a) Key: AC

b) $A^+ = ABE$

$C^+ = CD$

$A\cancel{B}\cancel{C}\cancel{D}\cancel{E}$

$$\boxed{\begin{array}{l} R_1 = ABE \\ R_2 = CD \\ R_3 = AC \end{array}}$$

28) $R = ABCDEFGHIJ$

$AB \to C$

$BD \to EF$

$AD \to GH$

$A \to I$

$H \to J$

Key: ABD

$AB^+ = \underline{AB}CI$

$BD^+ = \underline{BD}EF$

$AD^+ = \underline{AD}GHIJ$

$R = ABCDEFGHIJ$

$R_1 = \underline{AB}CI$ — $A^+ = AI$ — $R_1 = \underline{A}I$ → redundant

$B^+ = B$ — $R_2 = \underline{AB}C$

$R_2 = \underline{BD}\ EF$ — $B^+ = B$, $D^+ = D$ — $R_3 = \underline{BD}\ EF$

$R_3 = \underline{AD}\ GHIJ$ — $A^+ = AI$, $D^+ = D$

$R_4 = \underline{A}I$ → redundant

$R_5 = \underline{AD}\ GHJ$

$R_4 : ABD$

$R_6 = ABD$.

Finally, AI → 3NF AI

ABC

BDEF

ADGHJ

ABD

3NF

A table is said to be in 3NF, if its already in 2NF and shd be free from transitive dependencies.

Note:

If there is a transitive dependency, remove transitively dependent attr from 2NF table & place in separate table.

Q: 25.

$R_1 = \underline{ADE}IJ$

$R_2 = \underline{B}FGH$

$R_3 = ABC$

⇒

DIJ

ADE

FGH

BF

ABC.

Q: 26. $R_1 = ACDF$ CD $A \rightarrow FC$ (18)

$R_2 = BE$ ACF $C \rightarrow D$ AB

$R_3 = AB$ BE $B \rightarrow E$

AB

$A^+ = AFCD$

$B^+ = BE$

11/10

While decomposing universal table, we are eliminating insertion, deletion and modification problem.

Beyond 3NF if we make a move, these problems are further reduced but at the same time, we will invite few additional problems. Therefore they must be verified in BCNF and beyond that. They are:-

1. Lossless join property. (mandatory)
2. Dependency preserving property (optional)

Lossless join property:

A decomposition is said to be lossless if natural join of all decompositions = universal relations. (original table)

$$R = \Pi_{R_1}(R) \bowtie \Pi_{R_2}(R) \dots \Pi_{R_n}(R) \text{ - lossless}$$

$$R \subset \Pi_{R_1}(R) \bowtie \Pi_{R_2}(R) \dots \Pi_{R_n}(R) \text{ - lossy}$$

where $R_1, R_2 \dots R_n$ are fragmentations of universal relation R.

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_1	c_3

R_1

A	B
a_1	b_1
a_2	b_2
a_3	b_1

R_2

B	C
b_1	c_1
b_2	c_2
b_1	c_3

Check: $R = \Pi_{R_1}(R) \bowtie \Pi_{R_2}(R)$

A	B	C
a_1	b_1	c_1
a_1	b_1	c_3
~~a_2~~	~~b_2~~	~~c_2~~
a_3	b_1	c_1
a_3	b_1	c_3

Hence not equal.

R

R_1 R_2 R_3

R

R_1 R_2

9.30-1 calculus 807

~~2-5 digital~~ 312.

$R_1 \cap R_2 \to R_1$
$R_1 \cap R_2 \to R_2$
} If common column is key

$(R_1 \cup R_2) \cap R_3 \to R_3.$

$(R_1 \cup R_2) \cap R_3 \to (R_1 \cup R_2)$

$R_1 \cap R_2 \to R_1$ key.

$R_1 \cap R_2 \to R_2$ key.

if $R_1 \cap R_2$ is a key either in R_1 or R_2 then it is lossless otherwise lossy.

Dependancy preserving properties

① a decomposition is said to be dependancy preserving decomposition if $(F_1 \cup F_2 \ldots F_n)^+ = F^+$

$F_1 \Rightarrow$ set of func. dependancies in R_1

$F_2 \Rightarrow$ " " " " in R_2 & so on.

Eg:- A table $R = ABCD$

$$\boxed{\begin{array}{c} AB \to C \\ C \to D \end{array}}$$

decomposed into

R_1: $\underline{AB}C$ R_2: $\underline{C}D$

$F_1: AB \to C$ $F_2: C \to D$.

If ABCD

ABC BD

$F_1: AB \to C$ $F_2: \phi$

The following cases are nt properly handled by 3NF: (57)

① ~~If table consists composite primary keys~~ (AB, CD, EF)

② If composite prim. keys consists overlapping attributes

Differences b/w 3NF and BCNF:

3NF	BCNF
1) Its focus is on 1° key	1) its focus is on candidate key.
2) In 3NF, possibility for high degree of insertion, deletion & mod. problems due to candidate keys.	2) they are very much reduced as BCNF is taking care of candidate keys.
3) If there is a dependancy, $X \rightarrow Y$, it is allowed in 3NF, if X is a superkey or Y is part of some key	3) If there is a dependancy of the form $X \rightarrow Y$, X should be a superkey.

more ← redundancy problems less Comp. to 3NF

BCNF

A table is said to be in BCNF if it is already in 3rd normal form and all determinants are keys.

NOTE:

① If any dependancy violates BCNF rule, then place RHS attributes of that dependancy in a separate table along with copy of LHS., then remove those RHS attr from 3NF tables.

1° → Primary

44/32. R(ABCDEFGH)

AB → CEFGH
A → D
F → G
FB → H
HBC → ADEFG
FBC → ADE

a) key : AB

b) 2NF

~~$A^+ = AD$~~

$B^+ = B$

; partial

$R_1 = \underline{A}D$

$R_2 = \underline{AB}\ CEFGH$

c) 3NF

transiti

$R_1 = \underline{A}D$

$R_2 = \underline{F}G$

$R_3 = \underline{AB}\ CEFH$

AB CEF

FB H

d) BCNF

keys	determ
A	~~ABH~~
F	A ✓
AB	F ✓
	FB ✗
	HBC ✓
	FBC ✓

?

$\underline{A}D$

$\underline{F}G$

$\underline{FB}\ H$

$\underline{AB}\ CEF$

33. R = ABCDEFG

BCD → A
BC → E
A → F (TD)
F → G (TD)
C → D
A → G (TD)

a) key : BC

b) 2F

$B^+ = B$

$C^+ = CD$

$R_1 = \underline{C}D$

$R_2 = A\underline{BC}\ EFG$

transitive.

(c) $R_1 = \underline{C}D$

$R_2 = \underline{A}F$

$R_3 = \underline{F}G$] cant be combined since they have trans depen.

$R_4 = \underline{A}G$

$R_5 = A\ \underline{BC}\ E$

(d) BCNF (all other's key)

C ✓	BCD (superkey)
A ✓	BC ✓
F ✓	A ✓
A ✓	F ✓
BC ✓	C ✓
	A ✓

3NF = BCNF.

39. R = ABCDE

$AB \rightarrow CDE$

$C \rightarrow A$

$D \rightarrow E$

(a) key = AB

(b) 2NF = $\underline{AB}\ CDE$

(c) 3NF

$R_1 = \underline{D}\ E$

$R_2 = \underline{AB}\ CD$

d) BCNF

key	deter.	
D ✓	AB ✓	
AB ✓	C ✗	chk whether C is superkey
	D ✓	

$R_1 = \underline{C}\ A$

$R_2 = \underline{D}\ E$

$R_3 = \underline{AB}\ CD$

One table is subset of another table

41. $R \to ABCD$

$AB \to CD$

$C \to A$

$A \to C$ (PD)

(a) key = AB

(b) $A^+ = AC$

$B^+ = B$

$R_1 = \underline{A}C$

$R_2 = \underline{AB}D$

(c) 3NF

3NF = 2NF

$R_1 = \underline{A}C$

$R_2 = \underline{AB}D$

(d) BCNF

keys.	determinants
A ✓	AB ✓
AB ✓	C ?
	A ✓

$R_1 = \underline{A}C$

$R_2 = \underline{AB}D$

same.

38) $A \to BCDEF$

$B \to CD$

$C \to D$

$E \to F$

$E \to B$

key: A

b) $A^+ = \underline{A}BCDEF$

∴ in 2NF

P.D.

T.D.

c)

$\underline{B}CD$

$\underline{C}D$

$\underline{E}B$

$\underline{E}F$

$\underline{A}CE$

3NF

d)

A	B
B	C
C	E
E	E
E	A

∴ BCNF bad

16/10/10

35) (1) $R = ABCD$

(1) $B \to C$

$D \to A$

It is preserving dependencies but lossless join property is not preserved. hence decomposition is bad.

(2) $AB \to C$

$C \to A$

$C \to D$

$\bar{C}$ can det A & D

so Lossless join.

A	<u>C</u>	D

B	C

It satisfies lossless join property, BCNF but not dependency preserving property.

5) R = ABCD

A → B

B → C

C → D

A	B

R_1

A	D

R_2

C	D

R_3

It is not preserving lossless join property.

∴ decomposition is bad.

Q.43 31.) R = ABCD

I) C → D

C → A

B → C

a) key: B

b) transitive dependency present, ∴ it is in 2NF but not 3NF.

A B C D

c) $\boxed{\begin{matrix} \underline{C}D \\ \underline{C}A \\ \underline{B}C \end{matrix}}$ C C✓ C C✓ B B✓

⇓

BCNF

II) B → C

D → A

Key: BD

1NF but not 2NF

BD AC

$\boxed{\begin{matrix} \underline{B}C \\ \underline{D}A \\ \underline{BD} \end{matrix}}$ — 2NF = 3NF = BCNF

B | B

D | D

5) $AB \to C$

$AB \to D$

$C \to A$

$D \to B$

a) Key : AB

b) $\underline{AB}\,CD$ = 2NF = 3NF

$C \to A$
$D \to B$ are not allowed in BCNF

$\underline{C}A$ | $\underline{A}BCD$

$\underline{D}B$

$\underline{AB}CD$

↓

- 2 tables are subsets of $\underline{AB}CD$.

Rakhuraa

Advantages of normalization:

① it improves query retrieval performance.

② it eliminates insertion, deletion and mod. problems to great extend.

Disadvantages:

① It degrades query retrieval performance.

② Normalized tables will lose real world meaning.

Fundamental SQL queries: **SQL**

<u>Queries based on logical optrs</u> , AND, OR, <=, >=,

1) SELECT * from student where age >17 or branch = IT.
2) SELECT * from student where (city = Hyd and branch = CSE) or age ~~is greater th~~ >20

<u>Queries based on range optrs</u> (between / not between)

① Select * from student where marks b/w 300 and 400.;
② " " " " name b/w 'A' and 'K';
③ " " " " marks b/w 17 and 15;

b/w is inclusive.

③ 17 – 15 ✗
15 – 17

Queries based on set membership IN / not IN.

Find details of the students from 1st, 2nd and 3rd yr.

1) Select * from student where year = 1 or year = 2 or year = 3.
2) Select * " " " year in (1,2,3) .
3) " " " " " year b/w 1 and 3.
4) " " " " " year ~~not in~~ ≠ 4 (✗ not work always)
5) " " " " " year <= 3.
6) " " " " " not in (4) (✗ ")
7) " " " " " year < 4

Queries based on pattern matching: (like / unlike)

A% start with A

%A

%A%

'___' 3 letter word.

%/% % escape %
looking for %

Q: Find details of the students whose name starts with S and third letter is A and last letter is U.

Ans: select * from student where name like 'S_a%u'

Q: Find details of the students who are having letter e any where.

Ans where name like '%e%'

Q: Find details of students who are having exactly 3 letter name.

Ans where name like '___'

Queries based on null value:

is null is !null.

Eg: Select * from student where address is null.

" " " " where email is not null.

1. Select max(marks) from student.
2. Select max(name) from student.
3. Select avg(marks) from student.
4. Select avg (name) from student.
5. Select sum(mark1, mark2) from student.
6. Select ~~avg~~ aggr (mark1 + marks2) from student.
7. Select max(distinct marks) from student.
8. Select ~~avg~~ aggr $\left(\frac{mark_1^2 * \sqrt{mark_2}}{\log marks}\right)$ from student.
9. Select rollno, max(marks) from student.
10. Select * from student where marks = max(marks) — syntax error
11. Select sum(marks) / count(marks) from student.
12. Select count(*) from marks.
13. Select count (~~mark~~ name) from marks.

1/10

Max and Min funcs will work both with numeric and non-numeric columns.

6.30-8.30 - Digi
9 — 11 — DBMS

Sum & Avg only with numeric columns.

eg: 4)

Aggr func works on single column & produces single col. o/p.

eg: 5 ✗
6 ✓

4) Aggr func will work with simple mathematical func.

5) aggr. cant be used in where class, group by, order by clause.

It is freq used in select, having clause.

If thr is a non aggr column along with aggr col. ~~tot~~ in select stmt It shd be associated with group by clause. and all non aggr columns in select list shd appear in gp by class.

non aggr. aggr.
Eg: select rollno, count(marks) from student
group by roll no.
↳ then correct.

101	5
102	
103	
104	

this violates 1NF principle.

There is no influence of distinct keyword in max and min func. But it will have an impact on rest of the functions.

6) count * includes all values.
↳ null values and dupplicates

count excludes null values.

in some dbms, it excludes duplicates.

Eg: 101, –, 102, 101, 103, –, 105

count *	count
7	5/4

Group by

It is a very useful clause to get group aggr.

eg:- Find the total no. of stud. in each branch.

select brname, count (roll no) from student
group by name;

Eg:-

101	CSE
102	IT
103	CSE
104	ECE
105	CSE

o/p.

CSE	3
IT	1
ECE	1

Eg:- Total

CSE	200
IT	150
ECE	200

CSE	I	50	IT	50
CSE	II	70	IT	25
CSE	III	30	IT	25
CSE	IV	50	IT	50

Eg 3: Find details of stud in each branch in each yr.

select brname, year, count (roll no) from student group by
brname, year;

4: Select brname, year, gender, count (roll no) from student
group by brname, year, gender;

CSE	I	M	30	50
CSE	I	F	20	
CSE	II	M	40	70
CSE	II	F	30	

gp by ⇒ for analysis.

Having clause: used to filter group aggregates.

eg: find total female students in each branch under each yr & display results if count is more than 20 in any yr in any branch.

select brname, year, gender, count(roll no) from student where gender = 'F' group by brname, year ~~gender~~ having count(roll no) > 20 ;

Diff. b/w having and where clauses:

1) Where is used to filter rows.
having is used to filter groups

2) There is no alternative for 'where'.
To filter groups we have options other than 'having'

3) aggr cant be used in 'where' clause
can be used in 'having' clause.

Order by

1) select * from student order by roll no.

2) select roll no, name from student order by 1,2;

3) select roll no, name from student order by rollno ASC, name DESC ;

4) select brname, count (roll no) from student group by br name order by count (roll no);
aggr cant be used here

5) select roll no, name from student order by roll no, where br = "CSE" ;
shd be at last

→ Aggr. funcs are not allowed in order by clause
∴ we use alias for aggr func and it is used in order by clause if we want to order based on aggr. values.

branch	A
CSE	13
IT	12

after

br	A
IT	12
CSE	3

4th: select brname, count(roll no) ~~as~~ 'A' from student group by brname order by 'A';

select brname, count(roll no) as 'A' from student group by brname having A > 25 order by A;

not possible (alias not allowed)

→ having ~~count (roll no) > 25~~

Sub Queries:

1) It is one of the alternatives to get data from multiple tables.

2) It consists of

   outer query

   inner query

3) There is no restriction on the no. of levels

4) The followg optrs cant be used b/w inner & outer queries

   a) b/w and not b/w.

   b) like and not like.

   c) is null and is not null.

5) The followg optrs alone must be used " "

   a) in and not in

   b) any, all, some, greater than any/all,

   c) exist and not exist

Classification of subqueries:

a) uncorrelated subqueries:

In these, inner query is independent of outer query and it runs only once. Then result is substituted in the outer query.

~~Eg:- select * from students where marks <~~

(select avg(marks) from student)

— select * from student where marks = (select max(marks)
from student)

(displays details of stud who got max)

2) <u>Correlated subqueries</u>

In this querys inner query uses outer query variable and inner query runs as many times as the no. of values in outer query.

Eg: Find details of students who got n^{th} max.

101	~~550~~	101	550
102	600	102	600
103	500	103	500
104	400	104	400

select * from student S_1 where 3 = (select count (S_2.marks)
from student S_2
where S_1.marks <= S_2.marks)

— In another classification of subqueries we classify them as

1) <u>scalar subqueries</u> where one col & 1 row will be displayed.

eg:- select max(marks) from (select * from students where branch = CSE.

2) <u>Row subquery</u>

it retrieves multiple cols but single row.

3) <u>Table subquery</u>

retrieves multiple cols & multiple rows

in and not in optrs

Rno.	Name	Brno
101	a	1
102	b	1
103	c	2
104	d	3

Brno	Brname
1	CSE
2	IT
3	ECE

select * from student where ~~roll no~~ br.no in (select brno from branch where brname = 'CSE')

Eg: Pg 51
list out details of all studs who have performed transactions in the library.

select * from student where rollno in (select distinct roll no from library)

Pg 53: Find details of agents from hyderabad and delhi.

select * from agents where city in ('Hyd', 'Delhi')

a03
a06.

Q: Find customer ids who have done transactions with agents from Delhi or Hyd.

From Table:
Query identify: O/P:
Condtn:

c001
002
003
004
006

select cids from order where cid in (select aid from agents where city in ('Hyd', 'Delhi'));

Q: Find details of customers who performd transactions with agents from 'hyd' or 'delhi'.

select * from customer where cid in [ ]

— (_ _ _ _ _) ~~Any~~ (_ _ _ _ _)
IQ OQ

If thr is 'any' optr b/w inner and outer query then condtn is considered as true if atleast one of the rows satisfies this condtn. from the inner query.

If 'all' optr is b/w IQ & OQ, thn condtn is considered true if it is satisfied by all rows from inner query.

Q: Find details of agent who is offering min percent commission

Select * from agent where percent = 5; (not always)

Select * from agent where percent = (select min (percent) from agent)

Q: Find details of agents who offer more than min percent commission.

Select * from agent where percent > any (select percent from agents)

			>Any	≥any	>all
6 ✓	6		6	6	0
6 –	6		6	6	
7 =	7		7	7	
6	6		6	6	
5 ✗ not greater than any.	5			5	
5 ✗	5			5	

Note:
If inner query retrieves 0 rows: any optr treats this as false & all optr true condtn

B/w inner and outer query if thr is (or) an 'exists' operator, this condn is considered as true, if inner ~~query returns non empty set.~~

If thr is 'not exist' operator b/w inner and outer query, thn condition is considered as true, if inner query retns empty set.

Note:

Retrieve student details who perform transaction with library

1). Select * from student where rollno in (select roll no from library)

2) Select S.* from student S, library L where S.Rollno = L.Rollno.

3) Select S.* from student S where rollno exist (Select * from library L where S.roll no = L.rollno

Q. Find details of customers who purchased both the pdts P01, P07

Select cid from order where pid = 'p01';

∩

Select cid from order where pid = 'p07';

(or)

Select O_1.cid from Order O_1 where O_1.pid = p01 and exists (Select * from order O_2 where O_1.cid = O_2.cid and O_2.pid = 'p07')

any & all	exist
1) all queries retrd by exist optr maynot be possible with any and all.	1) all queries retrd by any & all can be obtained by exist optr
2) It is indepen. of correlated subqueries.	2) all queries are correlated subqueries.
3) Performance is good	3) always performance problems.

Join operations

Various ways to specify join condtns.

① Select * from student S, library L $\boxed{\text{where}}$ S.roll no = L.rollno.

② " " " $\boxed{\text{on}}$ S.roll no = L.roll no.

③ " " " using rollno.

④ Select * from student S $\boxed{\text{natural join}}$ library L

Types of joins:

① Natural join
② Self join
③ Equijoin
④ Inner join
⑤ Outer join
- left " "
- rt " "
- full outer join

Select * from ~~employee~~ 1, emp2 where emp.id = mgr.id; (28)

emp ID	name	age	Q	mgr. ID
1				2
2				2
3				2
5				5

Inner join:

In inner join if two tables are joined, only matching rows are displayed as o/p.

Outer join

In outer join, apart from matching rows, non-matching rows will also be displayed, but with null values

In left outer join, everything from LHS is displayed. If they have a matching row in RHS, that will also be displayed, else rt hand side table values are displayed with nulls.

In rt. outer join — opposite of left outer join.

Full outer join.

Full outer join = (left outer) U (rt outer)

City

S No	C.name
1	Hyd
2	Delhi
3	Chennai

flight

FNo	Starts from
A01	Blr
B02	Chennai
C03	Bombay

(1) select c.*, f.* from city c, flight f where
c.cname = f.start_from

3	Chennai	B02	chennai

(2) select c.*, f.* from city c left join flight f
where c.cname = f.starts_from

1	Hyd	X	X
2	Delhi	X	X
3	Chennai	B02	chennai

(3) Select c.*, f.* from city c ~~Right join~~ full join flight f where
c.cname = f.starts_from

X	X	A01	Blr
B	chennai	B02	chennai
X	X	C03	Bombay

1	Hyd	X	X
2	Delhi	X	X
3	Chennai	B02	Chennai
X	X	A01	Blr
X	X	C03	Bombay

Note: Select p.* from p, q, r where p.a = q.a or p.a = r.a; (29)

a) retns 0 ~~to~~ rows if p is empty

~~b) retns 0 rows if q or r is empty.~~

c) retns 0 rows if q and r are empty.

d) all of the above.

1) Select rollno, ~~age~~ count (marks) from student group by roll no. where roll no is a primary key.

What is not true abt this query?

a) length of o/p table is same as original table

b) o/p table consists duplicates.

c) o/p table never contains duplicate.

d) No syntax error here.

101	500	101	1
102	100	102	1
103	200	103	1
104	300	104	1

3) select * from student where name like 'A%' equivalent?

a) select * from student where name >= 'A' or <= 'B';

b) " " >= 'A' and <= 'B';

c) " " >= A and < B;

d) " " > A and < B;

4) Select O_1.pid from order O_1 where 2 <= (select count (O_2.cid) from order O_2 where O_1.cid = O_2.cid)

a) It retrieves pid purchased by atleast 2 customers. ~~1~~, 2, 3, 4

b) It retrieves pid " atmost 2 customers.

c) " " ~~pid~~ cid who purchased atleast 2 pdts.

d) " " ~~pid~~ cid who purchased atmost

Classification of relational algebra optrs.

Orafaq for quer

1) Native relational algebra optrs

σ select

Π project

ρ rename

← assign

÷ division

⋈ join

2) Set theory relational operators

∪

∩

−

×

3) Extended relational algebra optrs

max, min, sum, avg, count, count(*)

Select

- It is unary optr.
- Degree of o/p relation is same as original relation.
- It eliminates only rows but not columns.
- Commutative in nature.

$$\sigma_{C_1}(\sigma_{C_2}(R)) = \sigma_{C_2}(\sigma_{C_1}(R))$$

$$\sigma_{<C_1>}(\sigma_{<C_2>}(\sigma_{<C_3>}(R))) = \sigma_{<C_1> \text{ and } <C_2> \text{ and } <C_3>}(R)$$

$\Pi_{attr\text{-}list}(R)$

- degree of resultant R is equivalent to attr-list.
- It is not commutative.

$$\Pi_{name}(\Pi_{name, rollno}(student)) \neq \Pi_{name, rollno}(\Pi_{name}(student))$$

- Eliminates column, but it will also eliminate duplicate rows.

RA: $\Pi_{rollno, name}(\sigma_{age>15}(student))$

SQL: select rollno, name from student where age > 15;

SQL: select * from student where age > 15

RA: $\sigma_{age>15}(student)$

SQL: select rollno, name from student

RA: $\Pi_{rollno, name}(student)$

SQL: select * from student;

RA: student;

$\rho(\text{stu-CSE}) \leftarrow (\sigma_{br=CSE}(\text{student}))$

after this we can use this new name

$\pi_{name,age}(\text{stu-CSE})$

$\sigma_{age>15}(\text{stu-CSE})$

If there is no rename then

$\sigma_{age>15 \text{ and } br=CSE}(\text{student})$

Division

R

CNo	PNo
1	101
2	102
3	101
4	101
1	102

S

PNo
101
102

$R \div S = \dfrac{CNo, \cancel{PNo}}{\cancel{PNo}} =$

CNo
1

Eg ①:

A	B	C
a_1	b_1	c_1
a_2	b_1	c_2
a_1	b_2	c_1 ✓
a_1	b_2	c_2 ✓
a_2	b_1	c_2
a_1	b_2	c_3 ✓
a_1	b_2	c_3 ✓
a_1	b_1	c_5

$S_1 =$

C
c_1

$S_2 =$

C
c_1
c_2
c_3
c_4

$S_3 =$

B	C
b_1	c_1
b_2	c_1

$R \div S_1 = \dfrac{AB\cancel{C}}{\cancel{C}} =$

A	B
a_1	b_1
a_1	b_2

$R \div S_2 = \dfrac{AB\cancel{C}}{\cancel{C}} =$

A	B
a_1	b_2

combination having all c_1

c_1, c_2, c_3, c_4

$R \div S_3 = \frac{A\cancel{B}\cancel{C}}{\cancel{B}\cancel{C}} = \boxed{\begin{array}{c} A \\ \hline a_1 \end{array}}$

R

A	B	C
a_1	b_1	c_1
a_2	b_1	c_2
a_3	b_2	c_3

S

A	B	C
a_2	b_2	c_2
a_2	b_1	c_2
a_3	b_3	c_3

$R \cup S$ =

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

$R \cap S$ =

A	B	C
a_2	b_1	c_2

$R - S$ =

A	B	C
a_1	b_1	c_1
a_3	b_2	c_3

$S - R$ =

A	B	C
a_2	b_2	c_2
a_3	b_3	c_3

Q: Consider 2 table R_1 and R_2 with n_1 and n_2 rows. where n_2 is $>> n_1$

Find minimum and maxm rows for each of the following relational algebra expression.

Mention assumptions if any.

expression	assumption	Min	Max
$\sigma_{age>15}(R_1)$	age	0	n_1
$\Pi_{name,age}(R_2)$	name, age no duplicates	n_2	n_2
$R_1 \cup R_2$	✓	n_2	n_1+n_2
$R_1 \cap R_2$	✓	0	n_1
$R_1 - R_2$	✓	0	n_1
$R_1 \times R_2$	✓	$n_1 * n_2$	$n_1 * n_2$

rva drops my rows so n_2, n_2.

Complete set of relational algebra optrs:

$\sigma, \pi, \cup, -, \times, \cap, \div, \bowtie$

(1) $R \cap S = R \cup S - ((S-R) \cup (R-S))$

(2) $R \bowtie_C S = \sigma_{<C>}(R \times S)$

(3) $R \div S = T$

$T_1 \leftarrow \Pi_{attrlist}(R)$

$T_2 \leftarrow \Pi_{attrlist}(S \times T_1) - R$

$T = T_1 - T_2$

Precedence of relational operators:

π High

σ

$\times$

$\bowtie \div$

$\cap$

$\cup -$ Low

Relational calculus:

RA — How to get

RC — what to get

relational algebra is procedural lang and concerns abt how to get the result.

relational calculus is non procedural & looks for what to get.

SQL will have more flavours of relational calculus.

RC → tuple RC (no boundary)

RC → Domain RC (boundary is defnd) gives finite result.

name	age	addr

Eg:

TRC : $\{ t \mid S(t) \cap t.br = CSE \}$

SQL : select * from student where br=CSE

RA = $\sigma_{br=CSE}$ (student)

DRC = $\{ \underbrace{(Rno, name, br)}_{o/p} \mid S \underbrace{(Rno, name, br)}_{atts} \cap br = CSE \}$

Eg②: SQL: Select Rno from student where br=CSE;

RA : $\pi_{Rno}(\sigma_{br=CSE}$ (student))

TRC: $\{ t.rno \mid S(t) \wedge t.br = CSE \}$

DRC: $\{ (Rno) \mid S(Rno, name, br)\ (name, br) \wedge br = CSE \}$

o/p ①

total var ②

not o/p ③

Existennal quantifier ($\exists$)

$(\exists d)(c)$ is considered as true if condtn is true ~~for atleast one of the tuples else it is considered as false~~

Universal

$(\forall d)(c)$ is considered as true if condtn is true for all the tuples.

SQL: select * from student S, ~~[illegible]~~ library L where S.Rno = L.Rno

RA: $\sigma_{student.rno = library.rno}$ (student $\times$ library)

TRC: { t, L (all cols from t, L) | student(t) and ($\exists$L) library(L) and t.Rno = L.Rno}

DRC: { (Rno, name, age), (Rno, S/date, E/date) | ~~student~~ student (Rno, name, age) and $\exists$ (library) (Rno, S/date, E/date and Rno = Rno1 }

Transaction:

Trans.
↓ schedu
conc
↓
Seriab
↓

Need for transaction:

A_1 A_2

$A_1 = A_1 - 100$ $A_2 = A_2 + 100$

9.00 → C_1 → 9.10

C_1 9.05 C_2 9.05

$A = A_1 + A_2$

here C_2 gets wrong ans. since immd after that there is an updation.

① Inconsistent results.

② interference from various users in multi user environment.

③ ambiguity in deciding when to make changes permanent.

To Solve all the probs we have a sol called transactions with 4 properties:

④ ACID properties.

Atomicity
Consistency
Isolation
Durability

Atomicity:

- Either all or none of the transactions must be executed.
- no partial transactions.
- transaction mgr in dbms ensure this problem.

Consistency:

- Transaction optns on db shd bring db from one consistent state to another consistent state.

Eg: withdrawal of money.
maintain min. balance.

- None of the dbms component will ensure this property. Pgmr must implement this using pgmg logic.

Isolation:

- Optn of 1 transaction shdn't be interfered by another transaction.

- concurrency control protocol will ensure this

Durability

- changes made by the transaction shd be permanent.

Log mechanism will ensure this using transaction & recovery mngr.

Name	dep	sal
J	P	500
K	S	600
M	Q	~~100~~ 600

Problems:

① Dirty reading

reading uncommitted data

before committg "600" we perform sm calculatn on it but instead of commiting '600' can be aborted.

timing is imp. here.

read write conflict

② ~~Non~~ repeatable values

10·10 set M = 600

10·06 Read sal ⇒ 500, 600, 100

10·25 Read sal ⇒ 500, 600, 600

③ not repeatable reads (Rows)

10 St Trans 1

10·10 Insert ~~new~~ value (Jeff, D, 800)

10·20 Commit!

10·05 St T2

10·25 read salary — 500, 600, 100, 800

④ Incorrect summary problems

10. St Tx1

10·10 set M sal = 600

10·20 C

10·05 St Tx2

10·06 sum(sal) 1600

10·25 sum(sal) 1700

(5) lost update or ww conflict

10.00 St TX1
10.10 set m sal =600 ~~100~~ ~~600~~ 700
10.20 C1

10.15 Set TX 2
10.16 set m = sal = 700
10.17 C_2.

(6) Unstable control

10 St TX1
10.10 Drop col sal
10.15 C_1

10.05 St TX2
10.06 Read sal ✓
10.20 Read sal X

To solve transaction problems mentioned above we use schedule.

Schedule
Order of execution of stmts from different transactions is known as schedule.

Classification of schedule:

- Serial schedule
- Non serial schedule
  - recoverable
    - cascadg
    - cascadeless
  - non-recoverable.

T_1

T_2

S_1	S_2	
T_1	T_2	S1 everything from T_1 then T_2
T_2	T_1	S2 " " T_2 then T_1

ie, we can have $n!$ schedules posble

T_1 — if err is a blk, it stay at halt

T_2

so we look for non serial ~~serializability~~

T_1
T_2
T_1
T_2

$$\text{no. of nonserial schedule} = \frac{(n_1 + n_2 + \dots n_m)!}{n_1! \cdot n_2! \dots n_m!}$$

where n_1 = no. of stmts in transaction 1
n_2 = " " " 2
m = no. of transactions.

Eg:- $n_1 = 5$, $n_2 = 3$ no. of nonserial $= \frac{(5+3)!}{5! \times 3!} = \underline{56}$.

Recoverability (in non serial)

S1:

T_1	T_2
R(X) 10	
	R(X)
W(X) 20	
	W(X)
	R(X)
	C_2
a_1 ~~C_1~~	

T_2 not rolled bk.

If there is a necessity to rollback committed transaction, then schedule is called non recoverable schedule.

S_1 is non recoverable bcz instead of committg if T_1 aborts the opn, then T_1 is rolled bk but T_2 is not rolled bk. but there is a necessity to roll bk.

S2:

T_1	T_2
R(X)	
	R(Y)
W(X)	
	W(Y)
	R(X)
a_1 ~~C_1~~	
	C_2..

S_2 is recoverable but it is cascading

if T_1 is rolled bk we are able to roll back T_2.

roll bk of one transaction leadg to roll bk of another.

S3:

T_1	T_2
R(X)	
W(X)	
C_1	
	R(Y)
	W(Y)
	R(X)
	C_2.

S3 is recoverable cascadeless schedule but it is serial schedule. but we are interested in non serial.

S4:

T_1	T_2
R(X)	
	R(Y)
	R(X)
W(X)	
	W(Y)
	C_2
C_1	

S4 is recoverable cascadeless and non serial schedule.

1   1 × ×   3

2   2× ✓ ×   56

3 ✓   2

× ×   48

48

56 ✓ ×

A schedule S with n transactions is said to be serializable, if it is equivalent to a schedule S' with same n transactions.

where S → non serial schedule

S' → serial schedule.

3 kinds of serializability

1. result seria. or result equivalence
2. conflict " or conflict "
3. view " or view "

Rule

(1) O/p result of any transaction heavily depends on initial values of data items. For some initial values o/p may be same, result equivalence is not valid always.

Eg:-

(2)

10 $R_1(X)$	$W_2(X)$ 20
20 $W_2(X)$	$R_1(X)$ 20

✗

$R_1(X)$	$R_2(X)$
$R_2(X)$	$R_1(X)$

✓

$W_1(X)$	$W_2(X)$
$W_2(X)$	$W_1(X)$

✗

$R_1(X)$	$W_2(Y)$
$W_2(Y)$	$R_1(X)$

✓

$R_1(X)$	$W_1(X)$
$W_1(X)$	$R_1(X)$

✓

same trans, cant swap opn. So cant consider it as conflict opn.

$X_i(A)$, $Y_i(B)$

(1) $A = B$
(2) $i \neq j$
(3) X or Y must be write opn.

S1:

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

S2:

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

S2:

T1	T2
	R(X)
	W(X)
R(X)	
W(X)	
	R(Y)
	W(Y)
R(Y)	
W(Y)	

S2 is conflict equi to S1
(non serial) (serial)

Rules for view serializability

a) A non serial schedule S and serial S' are view equivalent if they meet all the following 3 condns:

① if T_1 reads initial value of X in S' then T_1 shd also read initial value of X in S

② If T_2 performs final write opn on X in S', then T_2 shd also perform final write opn on X in S.

③ If T_2 reads a value produced by T_j in S', then T_2 in S shd also Read value produced by T_j.

6-8 Sets. 404.
6-8:30 CO 311

Desirable properties of decomposition:

(1) lossless join

(2) dependency preserving.

Note:

If a relation R is given, then the decomposition of relation into R_1 and R_2 shd be done such that common attribute in R_1 and R_2 is a candidate key of anyone of the relation (either R_1 or R_2).

eg: $R = (A, B, C)$

Decomposition can be done as:

$$R_1 = (A, C)$$
$$R_2 = (B, C)$$

Dependency preservation:

If a relation 'R' is gvn, then it shd be decomposed into relations R_1, R_2 such that the FDs of relation R can be obtained from FD's of R_1 and R_2.

$$(F_1 \cup F_2)^+ = F^+$$

eg:- $R = (A, B, C, D)$

$A \to B$
$A \to C$
$C \to D$

$R_1 = (A, B, C)$
$R_2 = (C, D)$

$R_1 \cap R_2 = C$.

$R_1 \cap R_2 = R_2$

lossless join property is preserved.

FDs: $F_1 = A \to B$, $A \to C$ $\quad$ $F_2: C \to D$

$F_1 \cup F_2 = A \to B$
$A \to C$
$C \to D$

Eg 2: $A \to B$, $A \to C$, $C \to D$

$R = (A, B, C, D)$

$R_1 = (A, B, D)$
$R_2 = (B, C)$

$R_1 \cap R_2 = B$

lossless join not satisfied

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

$A \to B$ ✓

S1

T_1	T_2
R(A)	
A = A+10	
W(A)	
R(B)	
B = B+A	
W(B)	
	R(A)
	A = A+20
	W(A)
	R(B)
	B = B*1.1
	W(B)

S2

T_1	T_2
R(A)	
A = A+10	
W(A)	
	R(A)
	A = A+20
	W(A)
R(B)	
B = B+A	
W(B)	
	R(B)
	B = B*1.1
	W(B)

S3

T_1	T_2
R(A)	
A = A+10	
	R(A)
	A = A+20
W(A)	
	W(A)
R(B)	
B = B+A	
	R(B)
	B = B*1.1
W(B)	
	W(B)

S_2 is view equivalent to S_1 but S_3 is not view equivalent.

Differences b/w conflict serializability and view serializability

1) All conflict ~~sth~~ serializable schedules are view ser. schedules but converse not true.

2) It is easy to test and achieve conflict serializ. but its difficult to test and achieve view serializ.

3) Majority of concurrency control protocols are based on conflict serials. except Thomas ~~each~~ Wright rule.

S

T_1	T_2
$W_1(X)$	
	$W_2(X)$
$W_1(X)$	

S_1	S_3
T_1	T_2
T_2	T_1
$W_1(X)$	$W_2(X)$
$W_1(X)$	$W_1(X)$
$W_2(X)$	$W_1(X)$

S_3 is view equivalent to S.

but S_3 is not conflict equiv- to S.

S1

T1	T2
R(A)	
A=A+10	
W(A)	
R(B)	
B=B+A	
W(B)	
	R(A)
	A=A+20
	W(A)
	R(B)
	B=B*1.1
	W(B)

S2

T1	T2
R(A)	
A=A+10	
W(A)	
	R(A)
	A=A+20
	W(A)
R(B)	
B=B+A	
W(B)	
	R(B)
	B=B*1.1
	W(B)

S3

T1	T2
R(A)	
A=A+10	
	R(A)
	A=A+20
W(A)	
	W(A)
R(B)	
B=B+A	
	R(B)
	B=B*1.1
W(B)	
	W(B)

S_2 is view equivalent to S_1 but S_3 is not view equivalent.

Differences b/w conflict serializability and view serializability

1) All conflict ~~sch~~ serializable schedules are view ser. schedules but converse not true.

2) It is easy to test and achieve conflict serializ. but its difficult to test and achieve view serializ.

3) Majority of concurrency control protocols are based on conflict serials. except Thomas Write rule.

S

T1	T2
$W_1(X)$	
	$W_2(X)$
$W_1(X)$	

S1	S3
T1	T2
T2	T1
$W_1(X)$	$W_2(X)$
$W_1(X)$	$W_1(X)$
$W_2(X)$	$W_1(X)$

S_3 is view equivalent to S.

but S_3 is not conflict equiv. to S.

strict schedules — view, recoverable and cascadeless
serial schedule is strict.

99/3. a) first test conflict ser.

~~$T_1 : R(X)$ $T_2 : R(X)$ $T_1 : W(X)$ $T_2 : W(X)$~~

$T_2 < T_1$ $T_1 < T_2$

not conflict serializable.

T_1	T_2
$R(X)$	
	$R(X)$
$W(X)$	
	$W(X)$

~~X~~ X

T_1	T_2
T_2	T_1
$R_1(X)$	$R_2(X)$
$W_1(X)$	$W_2(X)$
$R_2(X)$	$R_1(X)$
$W_2(X)$	$W_1(X)$

R_2 reads value written by R_1

not view serializable.

— there is no dirty read, therefore it is cascadeless
cascadeless schedules are recoverable as there is no dirty read.

3(b) $T_1 : W(X)$, $T_2 : R(y)$, $T_1 : R(y)$, $T_2 : R(X)$

$T_1 < T_2$

it is conflict equivalent & conflict equivalent to ~~T_1 and T_2~~ serial schedule T_1 and T_2.
Since conflict equivalent, its also view equivalent.

T_1	T_2
W(X)	
	R(Y)
R(Y)	
	R(X) *

since it has dirty read, it is cascading schedule.

$C_2 < C_1$ X recoverable
$C_1 < C_2$ recoverable.

Depending on commit recoverability is decided.

3(c) $T_1 : R(X)$, $T_2 : R(Y)$, $T_3 : W(X)$, $T_2 : R(X)$

$T_1 < T_3$ $T_3 < T_2$

T_1
T_3
T_2

to conflict equivalent

to ∴ view equivalent.

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

dirty read (∴ cascading schedule)

$C_3 < C_2 \longrightarrow$ recoverable.

$C_2 < C_3 \longrightarrow$ not recoverable.

3(B) $T_1 : R(X)$ $T_2 : W(X)$, $T_1 : W(X)$, $T_2 : A$; $T_1 : C$

$T_1 < \cancel{T_2}$ $\cancel{T_2} < T_1$

based on rules its not conflict equivalent

but T_2 is aborted : ∴ s/m is left with only 1 transaction

T_1	T_2
R(X)	
	W(X)
W(X)	
	A_2
C_2	

Cascadeless and recoverable.

all cascadeless are recoverable.

d) $T_1: R(X)$, $T_1: R(y)$, $T_1: w(x)$, $T_2: R(y)$, $T_3: W(y)$, $T_1: w(x)$, $T_2: R(y)$

$T_2 < T_3$

$T_3 < T_2$

T_1	T_2	T_3
R(X)		
R(y)		
w(X)		
	R(y)	
		w(y)
w(X)		
	R(y)	

it is not conflict equivalent.

T_2	T_3
T_3	T_2
$R_2(y)$	$W_3(y)$
$R_2(y)$	$R_2(y)$
$W_3(y)$	$R_2(y)$

j) $T_2: R(X)$, $T_3: W(X)$, $T_3: C$, $T_1: W(Y)$, $T_1: C$, $T_2: R(Y)$, $T_2: W(Z)$, $T_2: C$

$T_2 < T_3$

since T_1 already C

T_1	T_2
T_2	T_1
T_3	T_3

conflict equivalent hence view equivalent too.

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

no dirty read so cascadeless & recoverable.

1.) $T_1 : R(X)$ $T_2 : W(X)$ $T_1 : W(X)$ $T_3 : R(X)$ $T_1 : C_1$ $T_2 : C$ $T_3 : C$

$\underbrace{T_1 : R(X) \;\; T_2 : W(X)}_{T_1 < T_2}$ $\underbrace{T_2 : W(X) \;\; T_1 : W(X)}_{T_2 < T_1}$

not conflict equivalent.

T_1	T_2	T_3
R(X)		
	W(X)	
W(X)		
		R(X)
C		
	C	
		C

R(X) cascading schedule

×	×	×	×	×	×
T_1	T_1	T_2	T_1	T_3	T_3
T_2	T_3	T_1	T_3	T_1	T_2
T_3	T_2	T_3	T_2	T_2	T_1

$T_1 < T_2$ not view equivalent

$T_2 < T_1$ "

not view equivalent. too.

cascading schedule but its recoverable since we commit T_3 after T_1.

Shortcut to identify conflict serializability

by using directed graph.

No. of nodes in the graph is exactly equivalent to no. of transactions in the schedule.

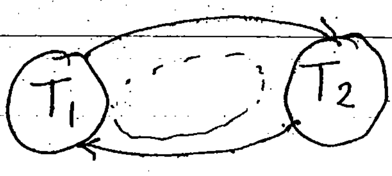
No. of edges in gph = no. of conflict opns in the schedule.

Once gph is drawn, verify for cycles, if graph consists cycles, then it is not conflict serializable.

19/4. $R_1(X) : R_2(X) : W_1(X) ; W_2(X)$, C_1, C_2

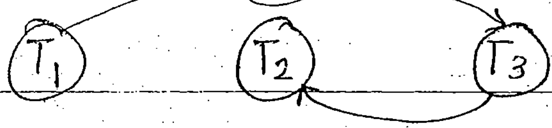
① ~~conflict~~ already ① is added so taken care

$\underbrace{R_2(X) : W_1(X)}_{T_2 < T_1}$ $\underbrace{W_1(X) ; W_2(X)}_{T_1 < T_2}$ not conflict equivalent.



not conflict equivalent.

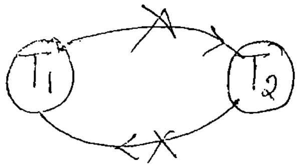
b) $R_1(X)$, $R_2(Y)$, $W_3(X)$, $R_2(X)$, $R_2(Y)$, C_1 , C_2



not conflict equivalent.

conflict equivalent ~~to~~ $T_1 T_3 T_2$

d) $W_1(X)$; $R_2(X)$, $W_1(X)$, C_2 , A_1



but T_1 is aborted.
so all edges cancelled
∴ conflict serializable.

(i) no cycles ∴ conflict equivalent.

79/2 . $R_1(X)$ $R_1(Y)$ $W_1(X)$ $R_2(Y)$ $W_2(Y)$ $R_1(Y)$ ~~W_1~~(Y)

WW conflict. suppose then

WR dirty read

conflict

T_1 is not allowed to write Y

(10) 10 RW 20 (20)

~~if it~~

R_1

WR conflict present. ~~[illegible]~~ non repeatable reads

RW conflict.

If there is W b/w R–R
then R-W conflict.

99/1 (i) $r_1(X)$ $r_3(X)$ $w_1(X)$ $r_2(X)$ $w_2(X)$

$T_3 < T_1$ $T_1 < T_2$

T_3
T_1
T_2

✓ conflict equivalent

Classification of concurrency control protocol

1) log-based protocol.

a) 2 phase locking

(i) B2PL

(ii) C2PL conservative

(iii) S2PL

(iv) R2PL rigorous

b) Graph based

2) Time stamp based protocol

a) Timestamp ordering protocol

a) ~~Fitho~~ Thomas Weight.

3) Multiple granularity protocol.

4) multiversional protocol.

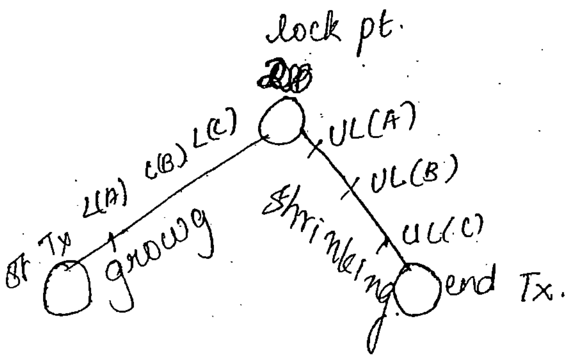
a) MV two phase proto.

b) MV timestamp ordering protocol.

Locks

Shared – R.

Xclusive – W.



all locks in one phase

after unlock no locking.

ie all unlocks together.

growg – lockg

shrinkg – unlockg.

RL(A)

WL(A)

RL(B)

RL(C)

UL(B)

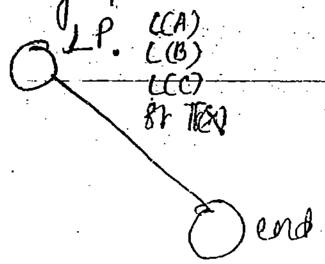
UL(A)

UL(C)

2) C2PL

L(A)
L(B)
L(C)
St. Tx
⋮
UL(A)
⋮
UL(B)
⋮
UL(C)
end Tx

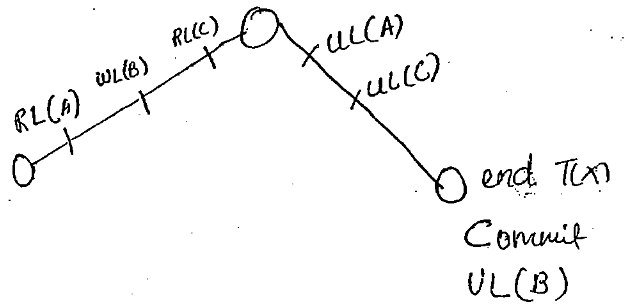
here no growing phase.



If all items are locked / waiting then we can't start.
If items more go for B2PL.

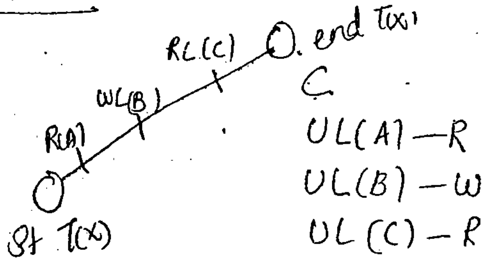
3) Strict 2PL

GATE'10



unlock write locks only after committing.

4) Rigorous 2PL:



③ & ④ are used nowadays.

---

eg: St Tx_1
$R_1L(A)$

St Tx_2
⋮
$W_2L(B)$ — it is strict 2PL.

UL(A) C_2
C1 UL(B) after committing
end Tx_2

disadv: of protocol.
① deadlocks
② starvation

St TX1 — 9.00
R(A) — 9.30
W(A) — 10.00
End Tx — 10.30

[A] RTS [9.00]
WTS [9.00]
TS.

Read and write timestamp values for a data item is not equal to actual read and write of a transaction. But its transaction timestamp. Starting time of transaction.

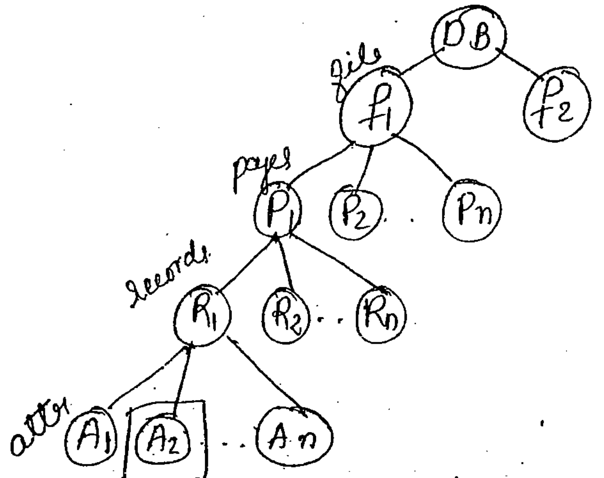
Pg. 88.

8.50 T_1	9.00 T_2
$W_1(X)$	
	$W_2(X)$
$W_1(X)$	

is also allowed.

TS → conflict serializ
Thomas wright — view serializ.

Multiple Granularity protocol
deals with size of db.



lock on roll no. only. for particular roll no
[select * fr student then lock on file. f1.

SIX condition that T_2 shd release when T_1 makes it X

SIX
9.00 S 9.05 R
R R
9.30 X 9.30
W
100
T_n

X|S is meangless

Consider a db with 2 files F_1 and F_2 and F_1 consists pgs P_1 to P_{1000} and F_2 consists P_{1001} to P_{2000}. Each pg consists 100 records $R_1 - R_{100}$. Each rec is read as (P_i, R_j)

$P \rightarrow$ page no. $R \rightarrow$ record no. For each of the following optns specify the sequence of blk request

① Read records from $P_1 \cdot 98$ to $P_2 \cdot 2$ ~~101~~.

IS on DB
IS on f_1
S on P_2
S on P_1

But if $P_1 \cdot 50$ is to accessed it cant be accessed since it is also locked. 100 rec are locked. reduces concurrency.

IS on DB
IS on f_1
IS on P_2
S on $P_2 \cdot 2$
S on $P_2 \cdot 1$
IS on P_1
S on $P_1 \cdot 100$
S on $P_1 \cdot 99$
S on $P_1 \cdot 98$

good method.

② Read rec from $P_{50} \cdot 1$ to $P_{100} \cdot 1$

IS on DB
IS on f_1
S on P_{100}
⋮
S on P_{51}
S on P_{50}

lengthy. we need almost 50 locks.

IS on DB
S on f_1

all thousand under control

∴ concurrency is low.

3) Read record $R_{50}.1$ to $R_{500}.1$

IS on DB
IS on f_1

S on P_{500}
⋮
S on P_2
S on P_1

lock

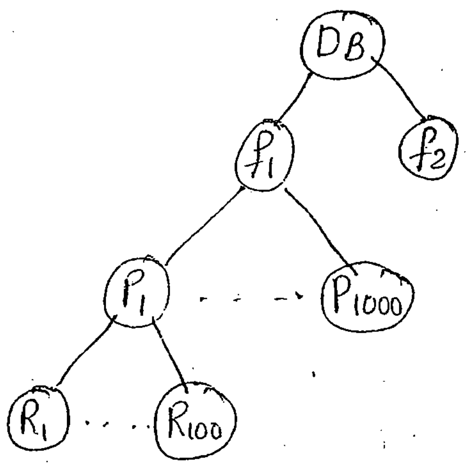
IS on DB IS on f_1 S on P_{500} ⋮ S on P_2 S on P_1	IS on DB S on ~~f_1~~

4) Delete first record in each page.

to delete - exclusive lock needed.

①	②	③
IX on DB IX on f_2 X on $P_{2000}.1$ X on $P_{1001}.1$ IX on f_1 X on $P_{1000}.1$ ⋮ X on $P_2.1$ X on $P_1.1$	IX on DB X on f_2 X on f_1	X on DB

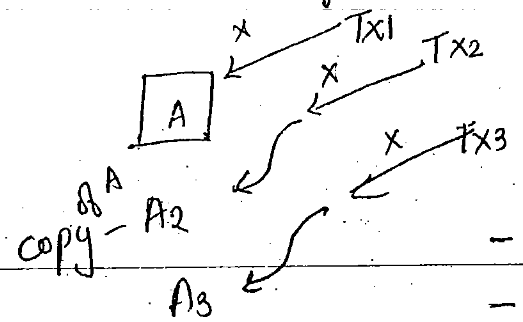
Q: Select * from student. Assume all student recs are in f_1.



no need to hold f_2.

IS on DB S on f_1 can be read.	S on DB. since f_2 will also be affected

Multiversion Locking protocol



we create separate version of dataitem A. for each TX

- improves concurrency. but granularity be small as poss.
- migrate A_3 to A_2 to A.
- If A_2 has not completed A_3 can be migrated directly to A.

if we ask lock a file and make copy takes longer time than getting actual lock on that file, then its disadvantageous.

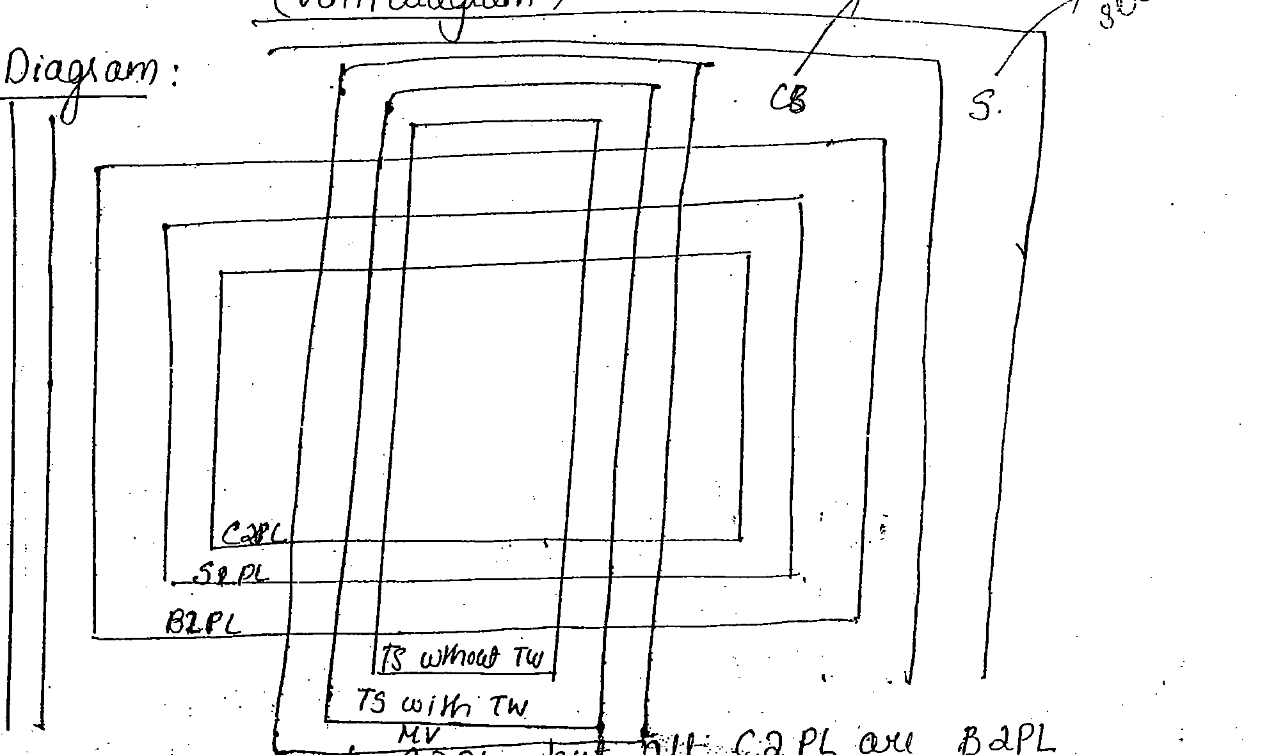
~~From files onwards to DB,~~ multiversion is not advan. because it is time consuming.

In multiversion, we created new Q_1 and give to old timestamp. Before, $TS(T_i) < WTS(Q)$ means it has to be rolled back.

Ques asked

1) strict 2PL (which represents ?)
   identify 2PL, BPL.
2) which repres. timestamp ordering?
3) relationship betwn protocols (Venn diagram)

Venn Diagram:



all B2PL are not C2PL but all C2PL are B2PL

Types of Indexes

- 1° index
- clustered index
- 2° index

another classification:

- Sparse index
- Dense index

1° index

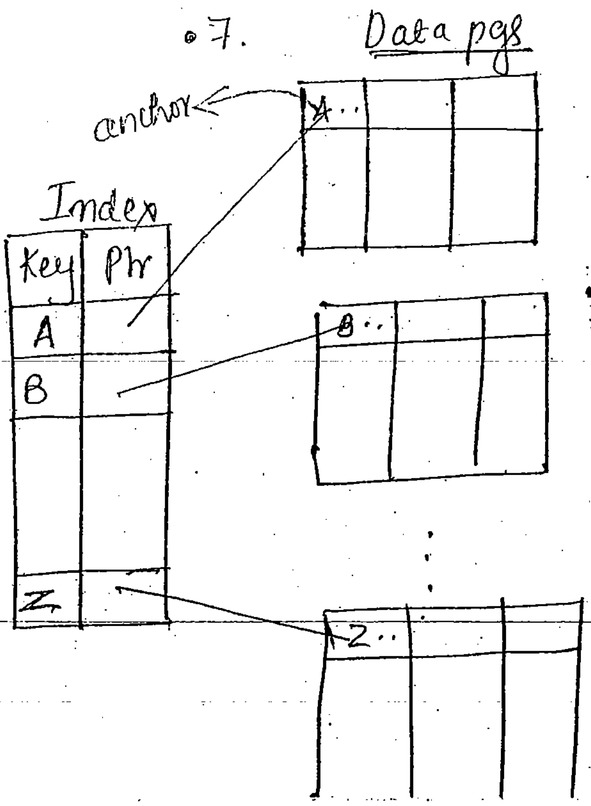
① It is created on primary key of a table hence there shd be only 1 primary index per table.

② It is an eg of sparse index.

③ Both data pgs & indx pgs are ordered.

④ It consists 2 attr — ① ~~set~~ search key value
② blk pointer.

⑤ Fill factor of data and indx pgs shd be from 0.5 to 1 and mostly it is around .7.



P. key : name.

1st rec in each pg is called anchor rec

Searching
- first in index then to data pgs.

(b) Index pages are developed using anchor rec.

Assume that no. of total records = 30,000,

~~Length of each record = 100 B, OS~~

OS page size = 1024 B

no. of rec / pg = $\frac{1024}{100}$ = 10

no. of pgs reqd = $\frac{30,000}{10}$ = 3000

no. of index records = 3000

Key = 9 bytes

Ptr = 6 bytes.

Length of each record = ~~9×~~ 9+6 = 15 B

no. of index rec / pg = $\frac{1024 \text{ (OS pg. size)}}{15}$ = 68

no. of index records (indx pgs) = $\frac{3000}{68}$ = 45.

Case (a) Search operation reqd without index

No. of searches reqd (Binary) = $\log_2 3000$

= 12

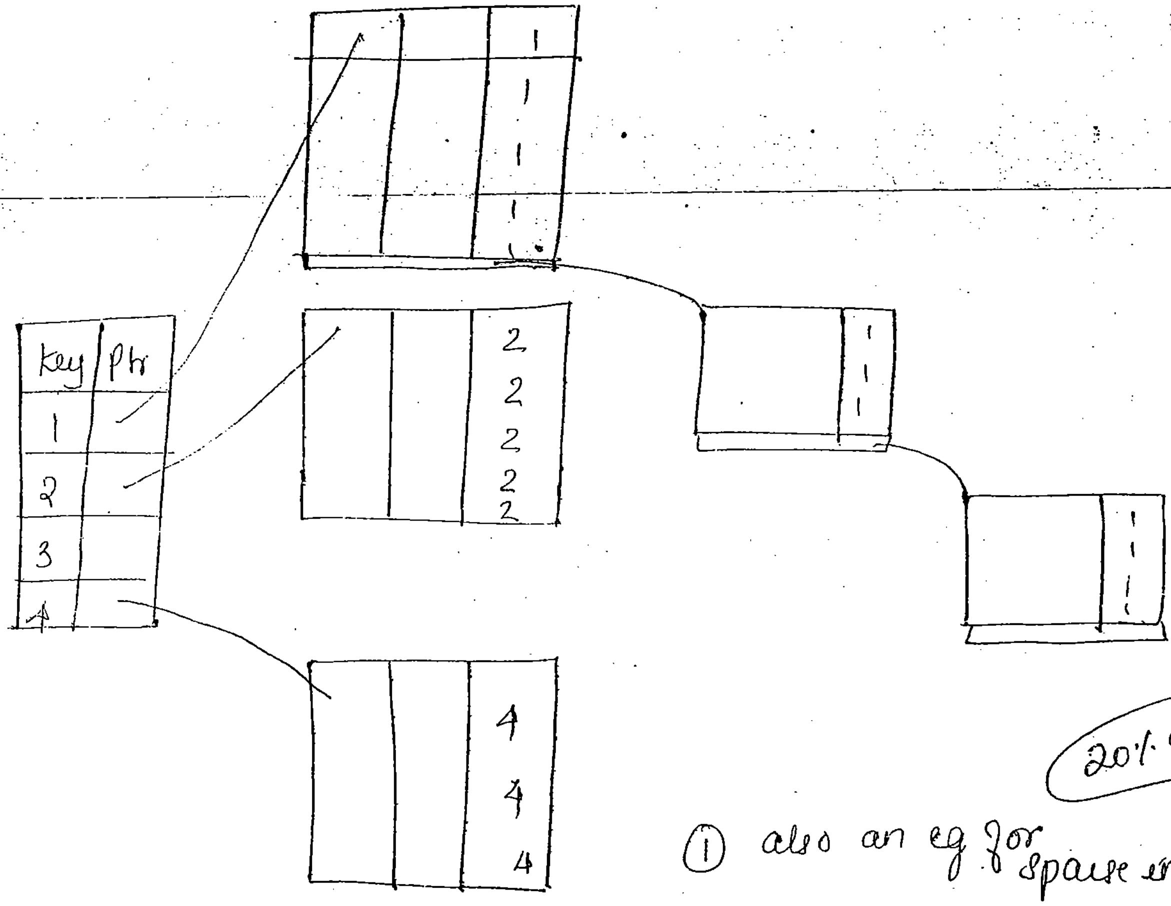
No. of searches with indx pgs = $\log_2 45$ = 6.

Search from index to data pg = 1

Total = 7 search.

60% queries run on 1° keys

Note: Primary Index is created on a column with unique values.
But clustered index is created on a gp of values.



20% queries

① also an eg for sparse index

② Only one cluster index per table

③ Both datapgs and index pgs are ordered.

Secondary index

① It is created on other than 1° key & clustered cols.

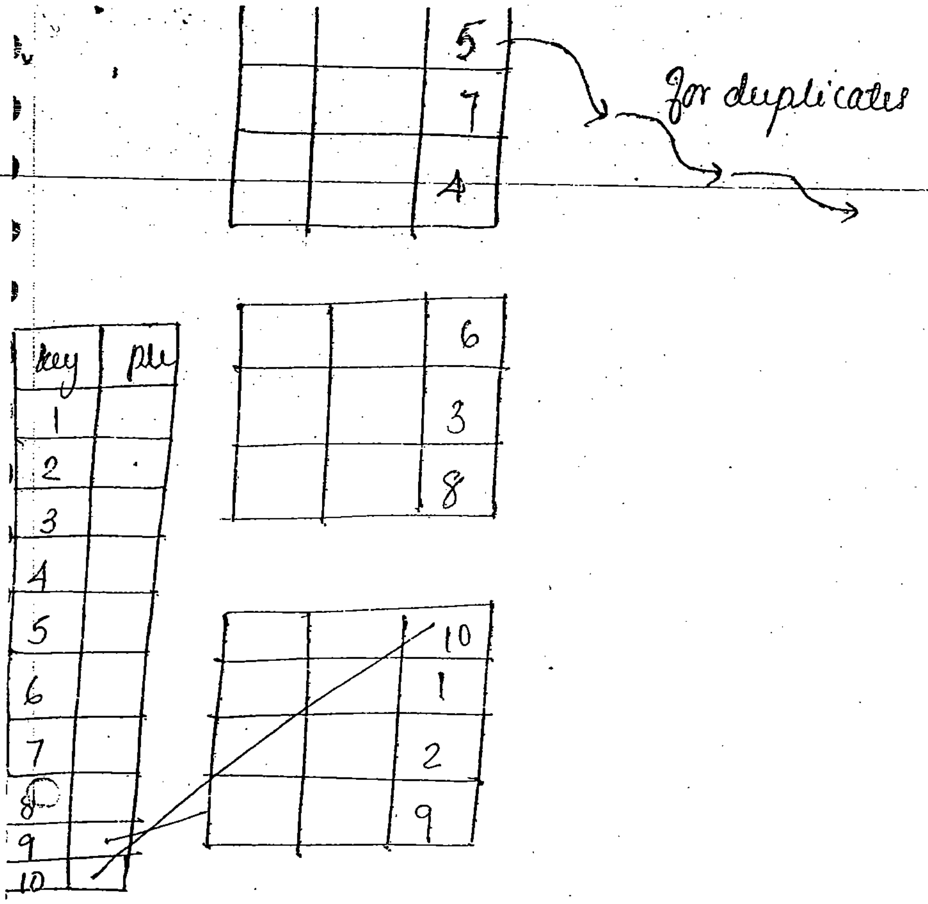
② It is an example for dense indx.

③ Each index consists 2 attr.

  ① Search key value
  ② Record ptr

GQ ④ Here indx pgs are ordered but not datapgs.

⑤ We can have more than 1 secondary index per table.



no. of rec = 30,000

length of each rec = 100 B

OS pg size = 1024 B

$$\text{no. of rec/pg} = \frac{1024}{100} = 10$$

$$\text{no of pgs reqd} = \frac{30000}{10} = 3000$$

10 $\therefore$ No. of indx rec = 30,000

key = 9 B

ptr = 6 B

len of each record = 9+6 = 15 B

$$\text{no. of index records/pg} = \frac{1024}{15} = 68.$$

$$\text{no. of indx rec} = \frac{30000}{68} = 450$$

Here linear search,

$$\text{we need} = \frac{3000}{2} = 1500$$

b) with indxg $= \log_2 450 = \underline{9}$

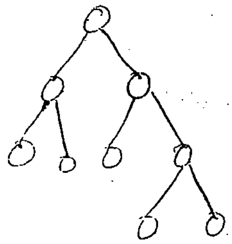
B and B^+ trees are multilevel indexes.

most effective indxg
$- 2^\circ$ indx

most useful indxg
$- 1^\circ$ indx

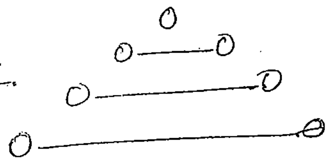
Q: Why are use B & B^+ in db?

Binary

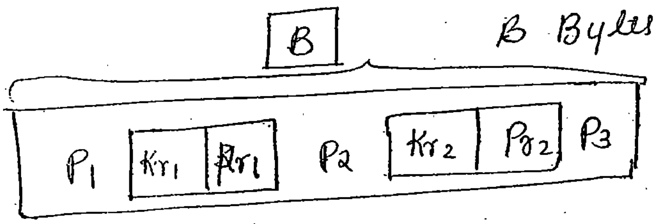


levels are more
grows vertically

B tree



grows horizontally.
with less no of levels.
data items is placed at equidistance
from root.



B Bytes.

$P \rightarrow$ blk ptr
$K_v \rightarrow$ key value
$P_r \rightarrow$ data ptr.
$B \rightarrow$ size of pg
$n \rightarrow$ order of pg

$$\boxed{nP + (n-1)K_v + (n-1)P_r <= B}$$

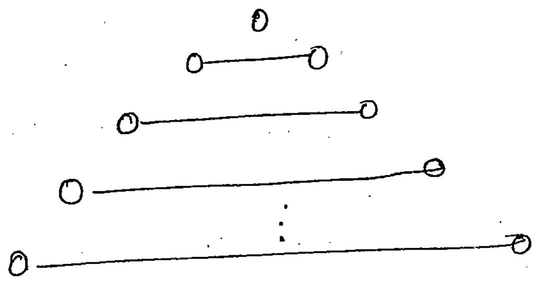
eg:- $P - 6$ B
$K_v = 9$ B
$P_r = 7$ B
$B = 512$ B

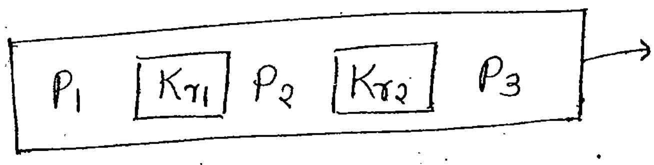
$n \times 6 + (n-1)9 + (n-1)7 <= 512$

$n = 23$
fill factor = 0.65
$n = 16$

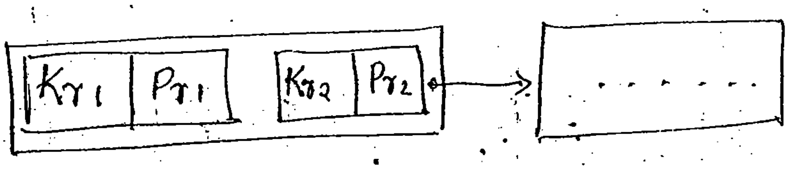
	Node	ptr	Data
R	1	16	15
		16×16	16×15
L_1	16	256	240
L_2	256	4096	3840
L_3	4096	65,536	61,440

c In B tree, each node has K value & ptr, to minimize this we have <u>B+ trees.</u>





10



$$nP + (n-1)K_r <= B$$

$$P = 6, \quad K = 9$$

$$n \times 6 + (n-1)\,9 <= 512$$

$$n = 32$$

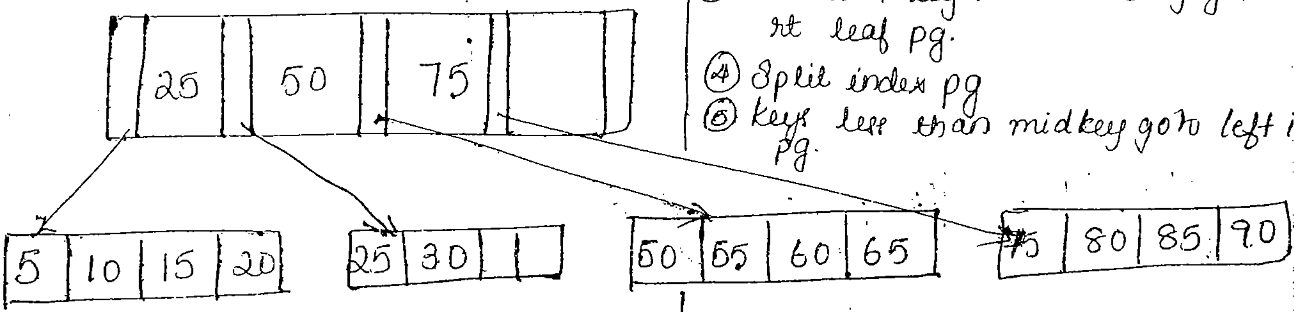
fill factor = 0.65

$$n = 23.$$

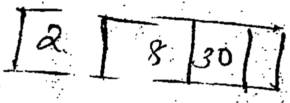
	nodes	ptr	Data
R	1	23	22
L_1	23	529	506
L_2	529	12,167	11638
L_3	12,167	2,79,841	2,67,674

Inserting data item:

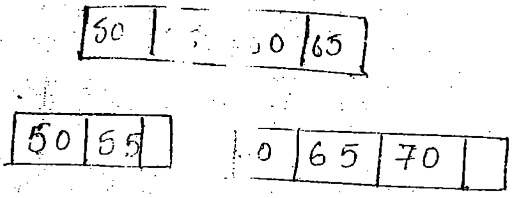
Leaf page full	Indx Page full	Action
NO	NO	Place the record in the sorted position in the appr. leaf pg.
YES	NO	① Split the leaf pg. ② Place the middle key in index pg in sorted order ③ Left leaf pg contains records with keys below the middle key. ④ rt leaf pg contains rec with equal or greater than middle.
~~YES~~	~~YES~~	
YES	YES	① Split the leaf pg. ② Rec with keys less than middle key go to left leaf pg. ③ Rec with key >= middle key go to rt leaf pg. ④ Split index pg ⑤ keys less than midkey go to left indx pg. ⑥ keys greater than midkey go to rt index pg. ⑦ Midkey goes to next higher level of index.



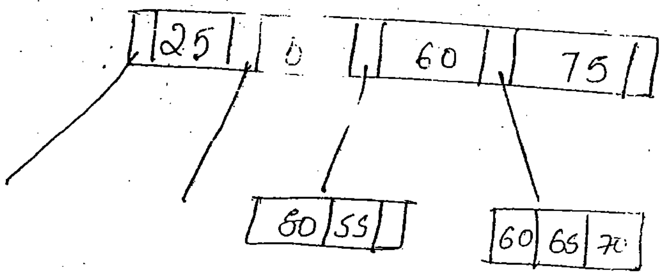
Case 1: add ...



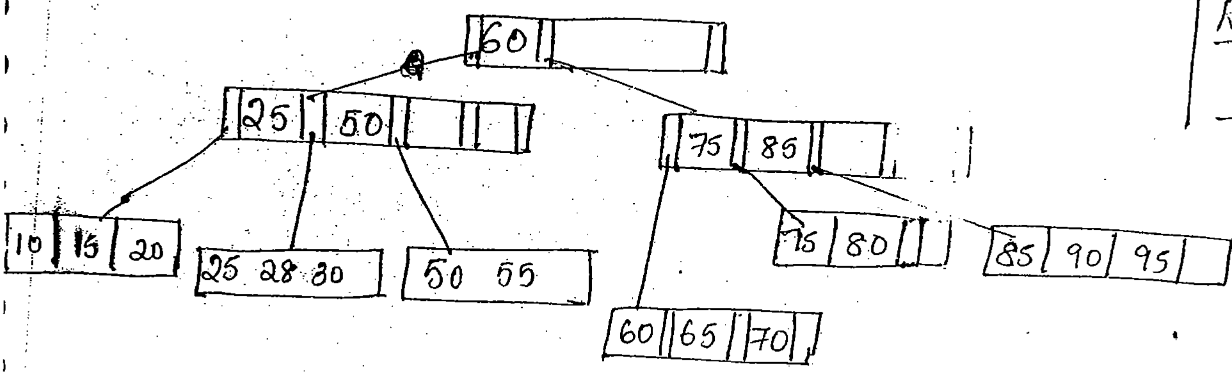
2: add record with key value 70.



push 60 to root



3: add 95.



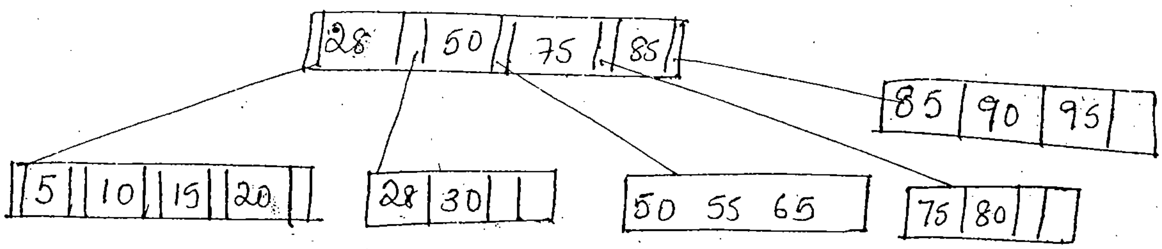
Root > only

>= only in leaf

Delete B+

Leaf Pg below fill factor	Index Page below fill factor	Action
NO	NO	① Delete rec from leaf page and arrange keys in ascending order to fill the gap. ② If the key of the deleted rec appears in index pg, use the next key to replace it
YES	NO	① Combine leaf pg and its sibling ② Change indx pg to reflect this change.
YES	YES	① Combine leaf pg and its sibling. ② adjust the indx pg to reflect the change. ③ combine the indx pg and its sibling. ④ Continue combining indx pgs until u reach a pg with correct fill factor (or u reach root pg).

ⓐ Delete 70
ⓑ " 25
ⓒ 60



50 — SQL
30 — Transac
10-15 — Norma
— Files & ER diag