Databases

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

Record: Collection of interrelated data.

Rno	Name	S.no	mark
101	xyz	1	100

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
    - online analytical processing
    - historical data
    - Data warehousing
    - decision making
    - Terabyte
    - CIO, MD, CEO, GM
    - R
  - OLTP
    - online transaction processing
    - current data.
    - Dbase.
    - Date to date oprns.
    - MB / GB.
    - used by clerks, leads, mgrs.
    - 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 - continuous images)
  - Distributed DB (eg: network dbs.).

Jagruti

- object - active / passive
  entity data + fn
  characters + behaviour
- entity - passive
  only characteristics
  no behaviour.

13/9/10

Database design

100% Reqments.

10% 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 structures

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, relatnshps & 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 entities

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

Physical entity: Person, vehicle, furniture

Conceptual : Sale, course, brand image

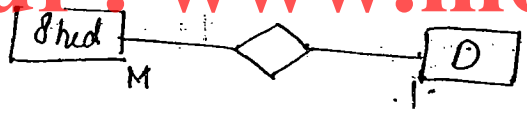
b) Relationships.

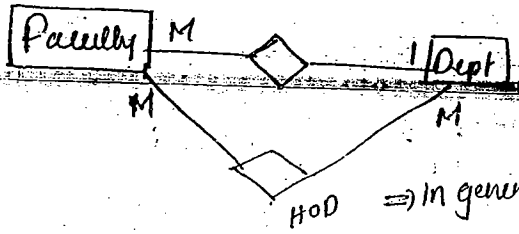
It gives association amg entities.

Btw entities, one/more relations are posble

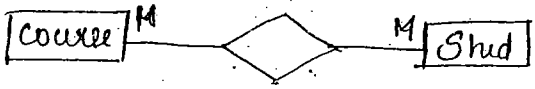
Types of relations

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

eg:- 

2) 

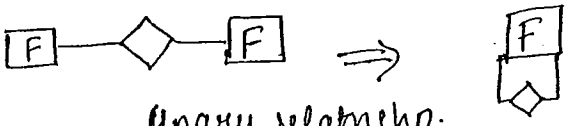
⇒ In general, it is one-one.

3) 

4) 

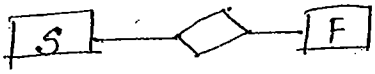
Degree of relationship:

It specifies no. of entities participating in a relationship.

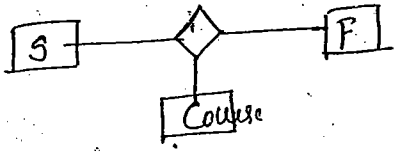
① 

Unary relationship.

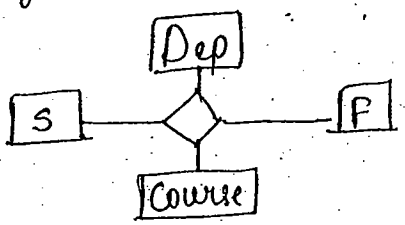
② Binary relationship

Only 2 entities participate. 

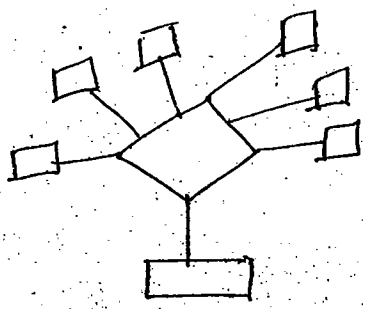
③ Ternary relationship.



④ Quarternary relation.



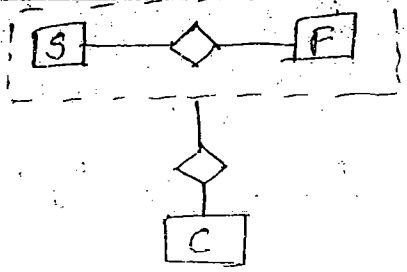
⑤ n-ary



ER diagrams 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.

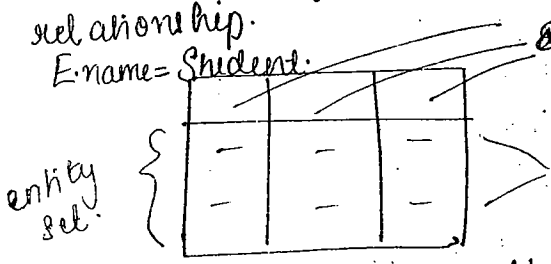
② Covering constraints

③ Overlapped constraints

② and ③ is used in EER (ER + OO)

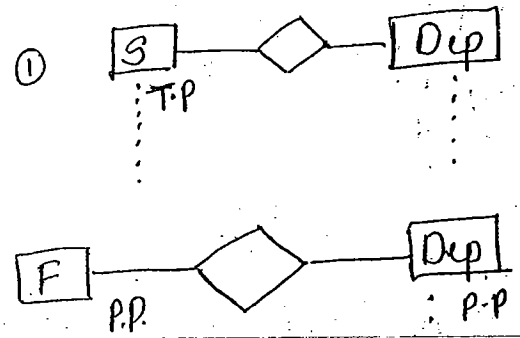
Participation constraints:

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



All students must have dept

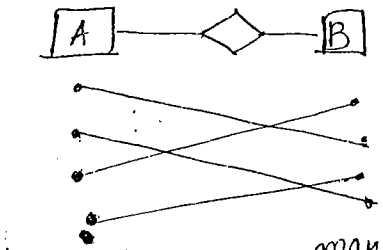
But all dept need not contain student

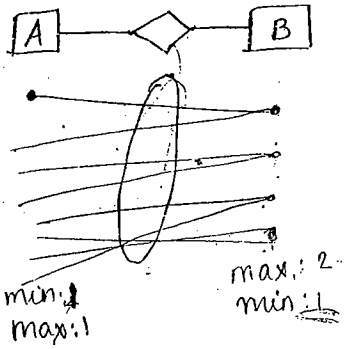
all students participate but depts donot.

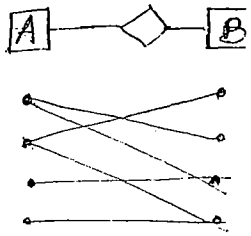
Dept may not contain a faculty. (Partial)

Cardinality ratio:

~~defns maximum no. of times an entity occurrance~~ participating in a relationship.



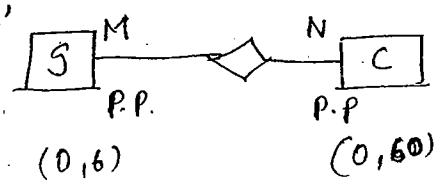




min cardinality : If 0 ⟹ partial participation.
If 1 ⟹ Total participation

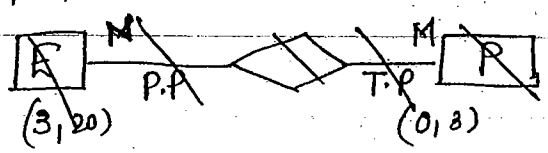
max cardinality : If 1 ⟹ entity occurance is participating thru a relation only once.
If N ⟹ then N no. of times an entity occurance participate in a relation.

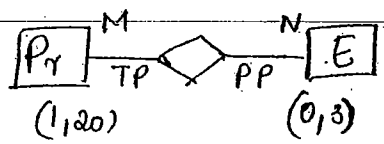
① Consider,



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.

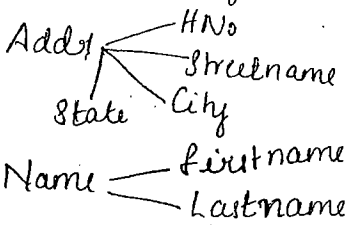
c) Stored and derived attr.

⇒ multivalued causes sm probs

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

Roll no, age

Composite – can be divided further into simple attributes



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

Multivalued : holds multiple values.

eg: address, telephn no, emailIDs

c) Stored : supplies values to derived attribute.

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

Brno	Bname	#
1	CSE	2
2	IT	1

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

Attr.

I° key attr

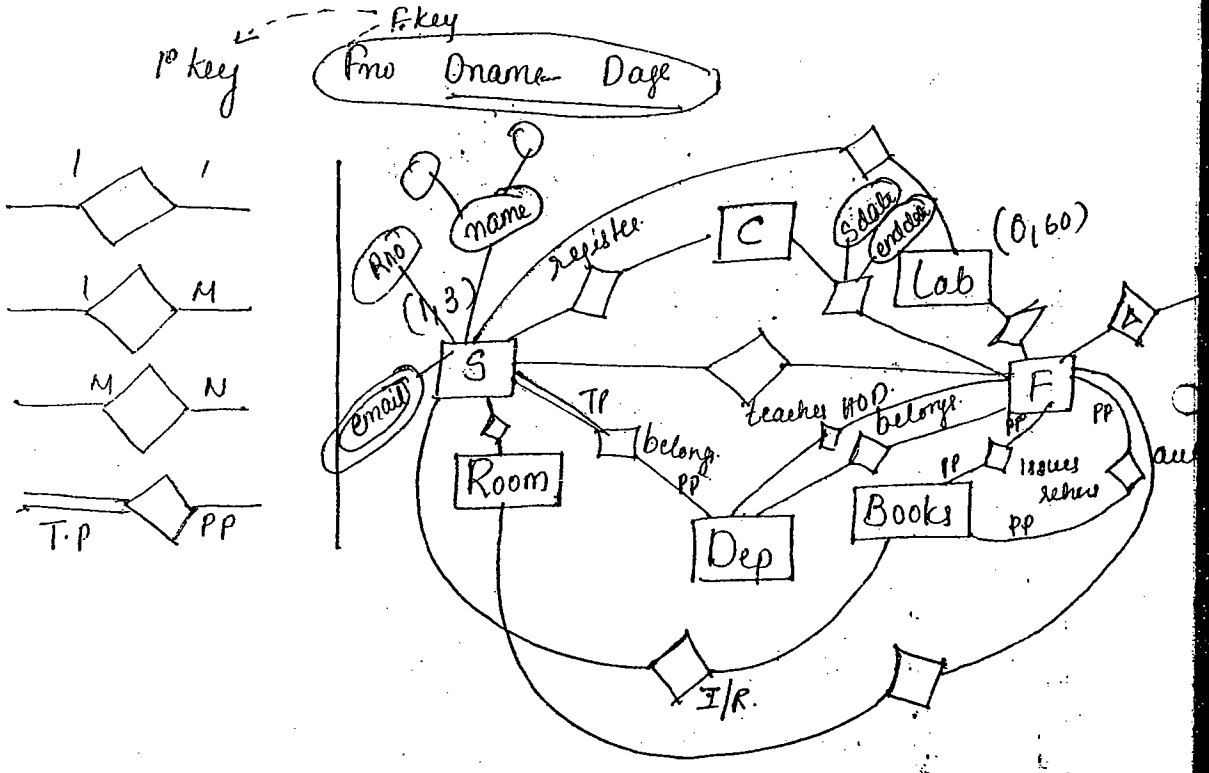
CA (composite)

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 I° key for the W.E is known as ----



Participation: ═ TP
— PP.

Cardinality ratio — (0,60) (1,3) ....

Another classification of entities

9 10:

Strong entity
Weak entity
Associative entity.

FNo (1° key)	FName
101	a
102	b
103	c

Dependent table.

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

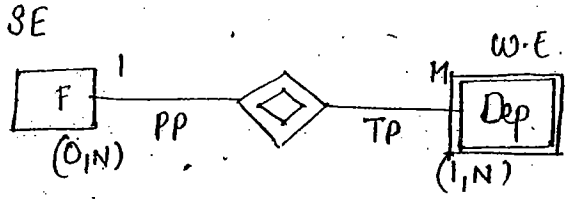
when 102 is removed, its dependent table entry also removd.

PNo (1° key)	Pname
101	a
102	b
103	c

Strong entity

PNo (1° key)	LNo (1° key)	Lhype
101	1A1	LMV
101	1B1	IDL
102	1L1	HMV

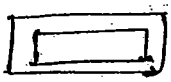
strong entity.

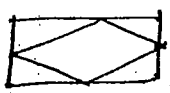


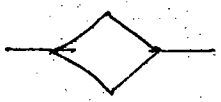
E-R Diagram notations:

Chen — Crow's feet → UML — Idejix X — Rein5 X


SE


WE


AE looks like relationship but actually its entity.


Relation


Identifying 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 & 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.

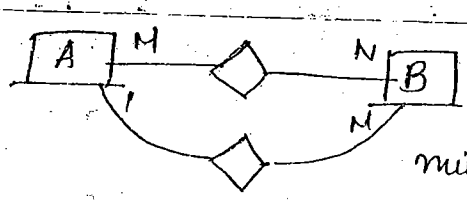
Step 7: Conversion of n-ary relationships.

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

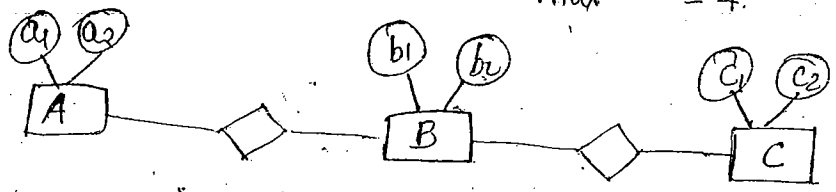
b) Declare the combination of foreign keys as 1° key.

Note:

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



min table = 3
max = 4.

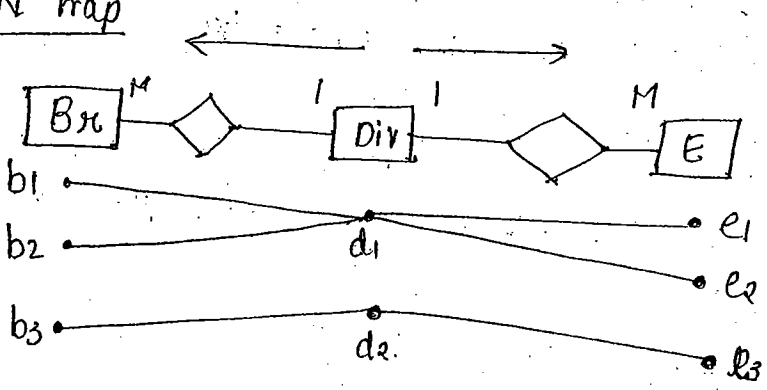


a) a_1 a_2 b_1
b) b_1 b_2 c_1
c) a_1 a_2 c_1
d) a_1 a_2 b_2

Trap

1) FAN trap
2) CHASM trap.

FAN trap



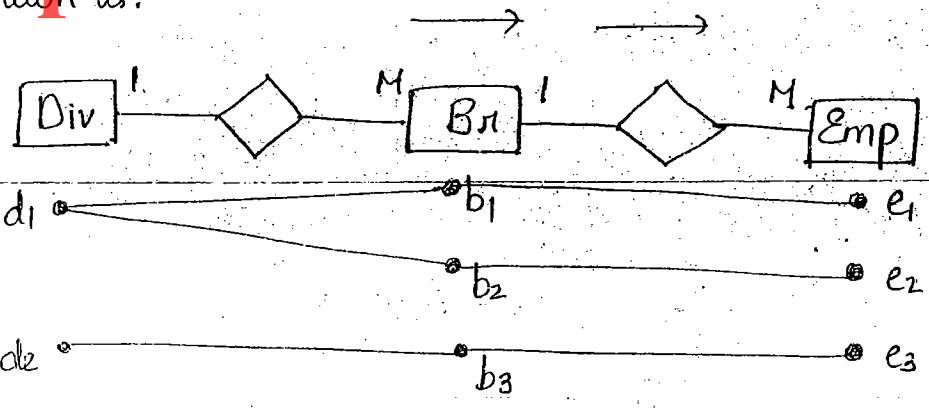
we cant say which e_1 belong to b_1 or b_2. Since d_1 has both.

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.

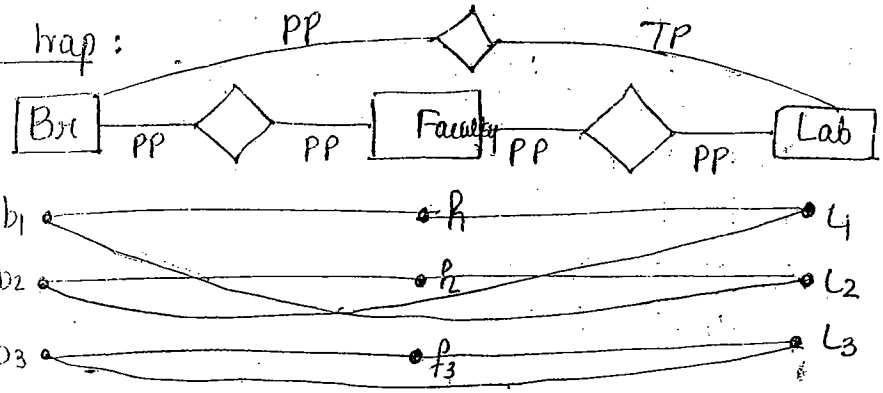
How to identify FAN trap?

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 not be correct.

This is due to CHASM trap.

How to identify?

If two directly related entities are connected thru another entity with partial participation 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 among database designer, domain expert & stakeholders.
2) It is ~~stio~~ tightly integrated with relational dB model.
3) It is easy to understand.

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 } attributes
Relation	relation b/w attr.	R	1:M } attributes
Attr	Entities	A	M:N } attributes
ER	FD, normalization	ER	
↓	↓	↓	
tables.	tables.	tables	
– big prjt	– small projt	↓	
– ✗	– purpul	FD, norm	
– easy	– difficult.	↓	
		tables.	

Defn: Functional Dependency:

It defns association amg entities.

RNo, name, Cname, Cno, Vno, Vchi 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) $A \to B$
$B \to C$
$C \to A$
~~$AC \to B$~~
$AB \to C$
$BC \to A$

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 ✗ not possible

2) deals with 1–1 relationshp.

3) It shd be non ~~dri~~ trivial ~~of~~ or completely non trivial.

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

Req
↓
F_1 : $A \to B$, $B \to C$
↓
F_2 (additional FDs) : $A \to C$

$F = F_1 + F_2$

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

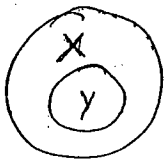
Total func. dependancies = $F_1 + F_2$.

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 $x \to y$, $y \to z$, $x \to z$.

③ $x \to yz$ then, $x \to y$, $x \to z$ — decomposition.

④ $x \to y$, $xz \to yz$

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

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

$C \to DE$: $C \to D$, $C \to E$

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

We don use inference rules to identify addn. FDs

wcf it has a certain drawbacks.

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 possible FDs.

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

Pg. A0

6. $A \to B$ ✓

$BC \to DE$ ✓

$AEG \to G$ ✗

compute AC^+

$X = AC$

$= ABC$

$= ABCDE$

$AC^+ = ABCDE$

(repeatedly go for 2 cycle)

→ determine

ie. $AC \to BDE$

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 addml 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)

Pg 41

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 possible.

$BCD^+ \to ABCDEH$
$\to AEH$

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

(i) $BE \rightarrow H$? True.

$BH \rightarrow AC$ X

BH^+ $X = BH$

$BH^+ = BH$

$BH \rightarrow BH$

Reqm

DB_1	DB_2
$A \rightarrow B$	$A \rightarrow B$
$C \rightarrow HD$	$D \rightarrow BC$
$F \rightarrow E$	$E \rightarrow AF$

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

Jan/10) F= $A \rightarrow C$, $AC \rightarrow D$, $E \rightarrow AD$, $E \rightarrow H$

G1 = $A \rightarrow CD$, $E \rightarrow AH$

Take F set and verify all its FDs can be ~~data~~ derivd from G1 or not

$A \rightarrow C$ compute A^+ from G1.

$AC \rightarrow D$ compute AC^+ from G1.

$E \rightarrow AD$ compute E^+ from G1.

Take 'G1' set and verify all its FD can be derived from F/no

$A \to CD$
$E \to AH$

compute A^+ from F ✓ A^+ —
E^+ from F ✓

⑩

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

⑫ Be →

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.

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

Candidate key : If we have one/more unique value columns in a table, then 1 of them can be elected as 1° key for a table. and these alternative keys are called candidate keys.

(or)

VehNo	EngineNo	DOP

R = ABCDEF

A → BCD
E → CD
F → BC

A
E
F

AG
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.

Dup	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.

10/10 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	BrNo	Bname
1	a	101	101	CSE
2	b	101	102	IT
3	c	102	103	ECE
4	d	105 X 103		

↳ Facultyno fname HOD Fkey.

P.key 1 a 2

2 b — since they are HODs

3 c —

4 d 3

5 e 3

GIQ. 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.

$A^+ = ABCDH$

$A \to BCDH$

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

Ans

AD - Superkey

CD -

BD -

(13) $R = ABCDE$

$A \to B$

$BC \to E$

$E D \to A$

$A^+ = AB$

$Bc^+ = BCE$

$ED^+ = EDAB$

$AB^+ = AB$

$BD^+ = BD$

$AD^+ = ADB$

$BCD^+ = BCDEA$ (key)

$ACD^+ = ACDBE$ (key)

ACD is the key.

Note:

ABCDE

$A \to BDE$

$C \to E$

$D \to BC$

A ✓ AC ✓ ACD ✓

C ✗ CD ✗

D ✗ AD ✗

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;

$AB^+ = ABGIFH$ includes mafority of attr.

$CJ = CJI$

$ABC^+ = ABGIFHCJI$

$ABD^+ = ABGIFHDE$

$ABCD^+ = ABCGHFEDI$

F_1

F_2 $\quad F = F_1 + F_2$.

$A \to B$

$B \to D$ $\quad \Rightarrow \quad$ $A \to B$, $B \to D$, $E \to F$

$E \to F$

$G \to H$ ✗

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; viz.

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 $AC \to D$ ✓

irreducible set

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

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

$AC^+ = ABC$

Step 3 remove A

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

redundant left

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

not equivalent - so cant remove A.

Remove 'C'

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

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

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

$AB \to C$	Step 1 single attribute rule
$C \to B$	① $AB = C$
$A \to B$	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

$A^+ = A$.

Step 3

~~$AB \to C$~~	Remove A
$C \to B$	$B \to C$
$A \to B$	$C \to B$
	$A \to B$

$B^+ = B \quad \neq \quad B^+ = BC$

can't remove A

remove B

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

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

$A \to C$
$C \to B$
$A \to B$ ✗

Step 4:
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

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

$\neq$

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

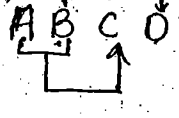
Question: To chk ans. correct → equivalence chk.
left hand side removal — step 3.
rt hand sid " — step 2.

Classifications 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

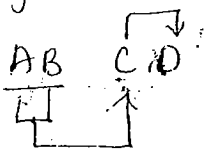
If there is a relation amg nonkey attr, then it is transitive dependency.

R = ABCD

AB → C

C → D

Key : AB



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

Under the following circumstances, no transitive

a) If table consists only 2 attr.

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

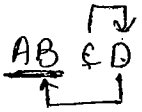
ABCD

AB → C

C → D, (Trans)

D → B ——→ non key attr identifies B.

BC → D



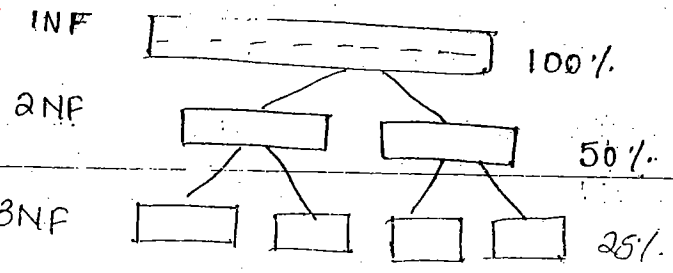
Full functional

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.



BCNF

4NF

5NF

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

Classification of dependencies:

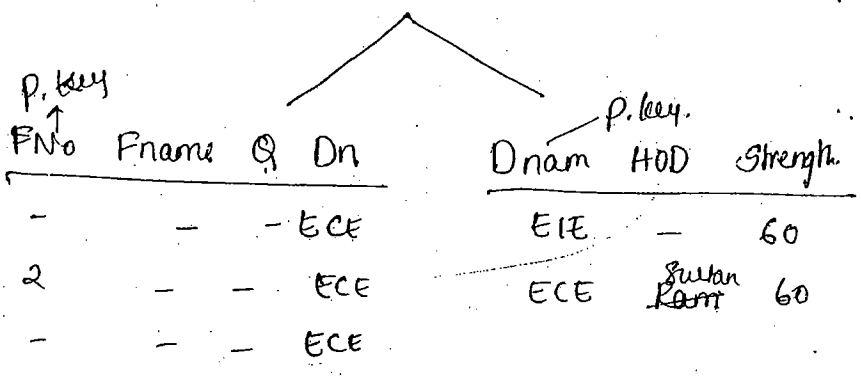
Functional multivalued Join Inclusion Template.

2NF 4NF 5NF
3NF
BCNF

Read: insertion, deletion, updation anomalies

Pg.17 Refer Table

Fno Fname & Dname HOD Strength



7.21 Normal Forms.

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

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

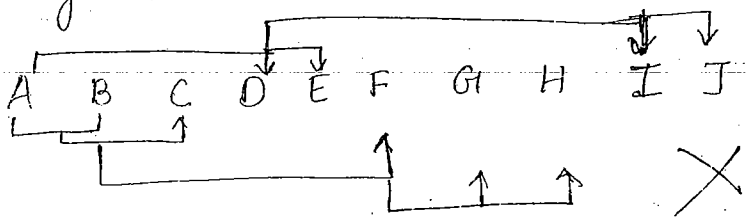
$A \to DE$ (PD)

$B \to F$ (PD)

$F \to GH$

$D \to IJ$

(a) Key: AB.



use closure method

$A^+ = ADEIJ$

$B^+ = BFGH$

2NF

$R_1 = \underline{A}\ DEIJ$

$R_2 = \underline{B}\ FGH$

$R_3 = \underline{AB}\ C$

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

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

b) 3NF

$\underline{C}\ D$

$\underline{B}\ C$

$\underline{A}\ B$

c) 3NF = BCNF

22. $ABD \to AC$
$C \to BE$
$AD \to BF$
$B \to E$

Step 1: i.

$ABD \to A$ ✗
$ABD \to C$ ✓
$C \to B$, $C \to E$ ✗
$AD \to B$ ✓
$AD \to F$ ✓
$B \to E$ ✓

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 $AD^+ = ADBCE$ (no F)
⑦ $B^+ = B$ (no E)

ii. $ABD \to C$
$C \to B$
$AD \to B$
$AD \to F$
$B \to E$

Remove A

$ABD \to C$
$C \to B$
$AD \to B$
$AD \to F$
$B \to E$

$BD^+ = BDE$
$AD^+ = ABCDFE$
$D^+ = D$.

$BD \to C$
$C \to B$
$AD \to B$
$AD \to F$
$B \to E$

$BD^+ = BDCE$
not equal

Remove B

$AD \to C$
$C \to B$
$AD \to B$
$AD \to F$
$B \to E$

$AD^+ = ADCBEF$
equal
hence B can be removed.

$ABD \to C$
$C \to B$
$AD \to B$
$D \to F$
$B \to E$
$B^+ = BDF$
can't removed

can't removed

iii. $AD \to C$
$C \to B$
$AD \to F$
$B \to E$
$AD \to B$

Union. $AD \to CFB$
$C \to B$
$B \to E$

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

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

$BE^+ = BEF$

$AE^+ = AEBCF$

$AB^+ = ABCDGH$

Step 3:

remove A

$BE^+ = BEHF$

not remove A

remove B

$AE^+ = AEBCF$

equal.

remove E

$AB^+ = ABCDGH$

not equal

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

Union

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

$AE \to H$

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

Step I satisfied.

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

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

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

remove D:
$BC \to BCAEFGD$ not equal.

remv C:
$BD^+ = BDAFG$

remv B:
$CD = C$

cnt remove:

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

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

$\equiv$

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

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

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

BCNF

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

$\Rightarrow$ 2NF

ACDE
AB

ACD
AB
AEC
BCE

no transitive dependency. so in 3NF also.

34) $A \to BC$

$ABE \to CDGH$

$C \to GD$

$D \to G$

$E \to F$

Key: AE

$AE^+ = ABCDEFGH$

b) 2NF

$A^+ = ABCGD$

$E^+ = EF$

A B C G D
E F
A E H.

2NF

c) 2 trans relations.

A B C
D G
C G D
E F
A E H

BCNF

A
Superkey A B E
C
D
E

A ✓
D ✓
C ✓
E ✓
AE ✓

BCNF

A B C
D G
C G D
E F
A E H

36) $R = ABCDE$

$AB \to CDE$

$A \to C$

$D \to E$

∴ AB

b) 2NF

$A^+ = CA$

$B^+ = B$

A C
A B D E

2NF

c)

A C
A B D
D E

3NF

BCNF

AB	A
A	AB
D	D

∴ also BCNF.

37) $AB \to CDE$ ABCDE

$A \to C$

$C \to D$

key: AB

b) $A^+ = ACD$

$B^+ = B$

A C D
A B E

2NF.

c)

A C
C D
A B E

3NF ← $3NF_2$ 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) $A^+ = AFCD$

$B^+ = BE$

$ABCDEF$

$R_1 = ACDF$

$R_2 = BE$

$R_3 = AB$

27. $R = ABCDE$

$B \to E$

$C \to D$

$A \to B$

a) Key: AC

b) $A^+ = ABE$

$C^+ = CD$

$ABCDE$

$R_1 = ABE$

$R_2 = CD$

$R_3 = AC$

28) R = ABCDEFGHIJ

AB → C
BD → EF
AD → GH
A → I
H → J

Key : ABD

AB^+ = ABCI
BD^+ = BDEF
AD^+ = ADGHIJ

R = ABCDEFGHIJ

R_1 = ABCI ← $A^+ = AI$ — $R_1 = AI$
$B^+ = B$ — $R_2 = ABC$

R_2 = BD EF ← $B^+ = B$, $D^+ = D$ — $R_3 = BD\ EF$

R_3 = AD GHIJ ← $A^+ = AI$, $D^+ = D$

R_4 : ABD

$R_4 = AI$
$R_5 = AD\ GHJ$
$R_6 = ABD$.

redundant

Finally, AI → 3NF → AI
ABC
BDEF
ADGHIJ
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 = ADEIJ
R_2 = B FGH
R_3 = ABC

⇒ DIJ, ADE, FGH, BF, ABC.

Q.26. $R_1 = ACF$ $R_2 = BE$ $R_3 = AB$

C.U: ACF, BE, AB

$A \to FC$, $C \to D$, $B \to E$

(18) AB ABF

$A^+ = AFCD$

$B^+ = BE$

10/10 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)

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

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

where $R_1, R_2 \ldots R_n$ 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

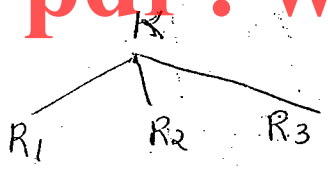
R_1 R_2

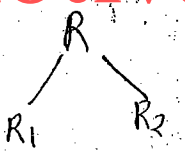
A	B
a_1	b_1
a_2	b_2
a_3	b_1

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	Hence not equal.
a_2	b_2	c_2	
a_3	b_1	c_1	
a_3	b_1	c_3	





9.30-1 calculus
307
2-5 digital
312.

If common column is key

$R_1 \cap R_2 \rightarrow R_1$
$R_1 \cap R_2 \rightarrow R_2$

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

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

$R_1 \cap R_2 \rightarrow R_1$ key.

$R_1 \cap R_2 \rightarrow R_2$ 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 $(F_1 \cup F_2 \ldots F_n)^+ = F^+$

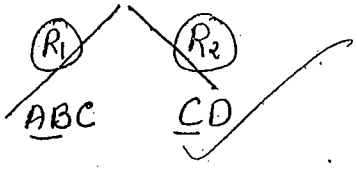
$F_1 \Rightarrow$ set of func. dependancies in R_1

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

Eg:- A table $R = ABCD$

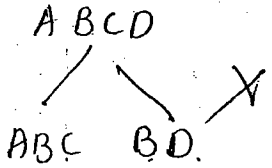
$AB \rightarrow C$
$C \rightarrow D$

decomposed into



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

If



F_1: $AB \rightarrow C$ F_2: ϕ

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

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, $X \to Y$, 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 $X \to Y$, 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, these RHS attr from 3NF tables.

1° → Primary

R: A1 A2 A3 K(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 = AD

R_2 = AB CEFGH

c) 3NF

transitive

R_1 = AD

R_2 = FG

R_3 = AB CEFH

AB CEF

FBH

d) BCNF

keys	determ
A	AB ✓
F	A ✓
AB	F ✓
	FB ✗
	HBC ✓
	EBC ✓

AD

FG

FBH

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 = CD

R_2 = ABC 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.
$R_4 = \underline{A}\,G$
$R_5 = A\,\underline{B\,C}\,E$

(d) BCNF (all other 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 = $\underline{AB}\;C\;D\;E$

(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

$R_1 = \underline{A}C$

$R_2 = \underline{AB}D$

3NF = 2NF

(d) BCNF

A key.	AB ✓ determinants
AB ✓	C ?
	A ✓

same.

$R_1 = \underline{A}C$

$R_2 = \underline{AB}D$

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.

c)

BCD	
CD	
EB	
EF	
ACE	

3NF T.D.

d)

A	B
B	C
C	E
E	E
E	A

∴ BCNF bad

15/10/10 35) (1) $R = ABCD$

① $B \to C$

$D \to A$

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

(2) $AB \to C$

$C \to A$

$C \to D$

C can det A & D

So Lossless join.

A	C	D

B	C

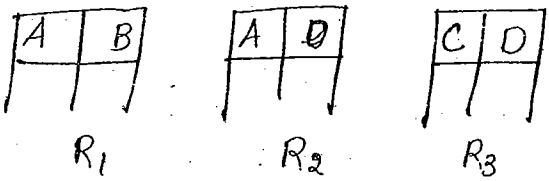
It satisfies lossless join property, BCNF but not dependency preserving property.

5) $R = ABCD$

$A \to B$

$B \to C$

$C \to D$



It is not preserving lossless join property.

∴ decomposition is bad.

Q.43 31) $R = ABCD$

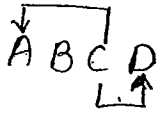
I) $C \to D$

$C \to A$

$B \to C$

a) key: B

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



c) $\underline{C}D$, $\underline{C}A$, $\underline{B}C$

C C
C C
B B

⇓

BCNF

II) $B \to C$

$D \to A$

Key: BD

1NF but not 2NF

BD AC

$\underline{B}C$, $\underline{D}A$, $\underline{B}D$ → 2NF = 3NF = BCNF

B B
D D

5) $AB \rightarrow C$
$AB \rightarrow D$
$C \rightarrow A$
$D \rightarrow B$

a) Key : AB

b) AB CD = 2NF = 3NF

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

CA	ABCD
DB	
ABCD	

↓

- 2 tables are subsets of ABCD.

Rakhu...

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.

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

1) SELECT * from student where age >17 or branch = IT.

2) SELECT * from student where (city = Hyd and branch = CSE) or age >20 ~~is greater than~~

Queries based on range optrs (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 ≠ 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. (23)

(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 + mark2) from student.

(7) Select max(distinct marks) from student

(8) Select ~~avg~~ aggr $\frac{(mark_1^2 * \sqrt{marks})}{\log marks}$ 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(name) from marks.

1/10

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

6.30-8.30 - Dbj
9-11 - Doks

Sum & Avg only with numeric columns.

eg: 1)

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

eg: 5 X
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 frequently used in select, having clause.

If there is a non aggr column along with aggr col. ~~not~~ 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 rollno.

↓ Not correct.

101 102 103 104	5

This violates INF 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 duplicates

count excludes null values.

in some dbms, it excludes duplicates.

Eg: 107
—
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;
```

1: 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 clause:

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. can't 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(1), name(2) 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. can't be used here
5) select roll no, name from student order by roll no, where br = "CSE";
   should 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	br	A
CSE	13	IT	12
IT	12	CSE	3

after

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 following 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 following optrs alone must be used in
   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) Correlated subqueries

In this query 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 nth 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) Scalar subqueries where one col & 1 row will be displayed.

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

2) Row subquery

it retrieves multiple cols but single row.

3) Table subquery:

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	Brnam.
1	CSE
2	IT
3	ECE

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

Eg: Pg 51

List out details of all studs who have performed transactions in the library.

Select * from student where roll no 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 [ ]

If there 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' optrs is b/w IQ & OQ, then 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)

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

Note:

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

Blw inner and outer query if there is an 'exists' operator, this condtn is considered as true, if inner query returns non empty set.

If there 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.rollno=L.rollno)

Find details of customers who purchased both the pdts Po1, Po7

Select cid from order where pid = 'po1';

n

Select cid from order where pid = 'po7';

(or)

Select O_1.cid from Orders O_1 where O_1.pid = po1 and exists

(Select * from order O_2 where O_1.cid = O_2.cid and O_2.pid = 'po7')

any & all	exist
1) all queries retnd by exist opt may not be possible with any and all.	1) all queries retnd by any & all can be obtained by exist opt
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 condns.

① Select * from student S, library L where S.rollno = L.rollno.

② " " " on S.rollno=L.rollno.

③ " " " using rollno.

④ Select * from student S natural join library L

Types of joins:

① Natural join
② Self join
③ Equijoin
④ Inner join
⑤ Outer join — left " "
right " "
Full outer join

Select * from employee e1, emp2 where e1.empid = (3)
mgr.id ;

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) ∪ (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 full join / right 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;(2 rows)

a) returns 0 rows if p is empty

b) returns 0 rows if q or r is empty

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

d) all of the above.

1) Select rollno, 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.

b) It retrieves pid " atmost 2 customers.

c) " " cid who purchased atleast 2 pdts.

d) " " cid who purchased almost

Classification of relational algebra oprs

Oracle for quer

1) Native relational algebra oprs

- σ select
- Π project
- ρ rename
- $\leftarrow$ assign
- $\div$ division
- $\bowtie$ join

2) Set theory relational operators

- $\cup$
- $\cap$
- $-$
- $\times$

3) Extended relational algebra oprs

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

Select → 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.

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

after this we can use this new name

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

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

If there is no rename then

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

Division R

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

S

PNo
101
102

$R \div S = \frac{CNo, PNo}{PNo} =$

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

$S_1 = \frac{C}{C_1}$

$S_2 = \frac{C}{C_1, C_2, C_3, C_4}$

$S_3 =$

B	C
b_1	c_1
b_2	c_3

$R \div S_1 = \frac{AB\not{C}}{\not{C}} =$

A	B
a_1	b_1
a_1	b_2

$R \div S_2 = \frac{AB\not{C}}{\not{C}} =$

A	B
a_1	b_2

combination having all C_1 C_1, C_2, C_3, C_4

$R - S_3 = \frac{A\,B\,C}{B\,C} = A$

a_1

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
$\Pi_{name,age}(R_2)$ (we drops my rows so n_2, n_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
$R_1 \times R_2$	✓	$n_1 * n_2$	$n_1 * n_2$

Complete set of relational algebra oprs:

$\sigma, \Pi, \cup, -, \overset{X}{*}, \cap, \div, \bowtie$

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:

High.

Π

σ

$\times$

$\alpha, \div$

$\cap$

$\cup -$

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

SQL : select * from student where br=CSE

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

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

o/p attrs

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

Existential quantifier (∃)

$(\exists d)(c)$ 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, ~~letter~~ library L where S.Rno = L.Rno

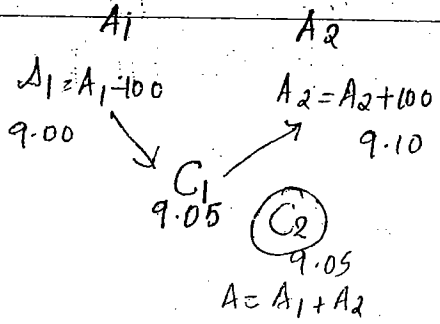
RA: $\sigma_{student.rno = library.rno}$ (student × library)

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

DRC: $\{ (Rno, name, age), (Rno, S/date, E/date) \mid$ ~~student~~ student(Rno, name, age) and $\exists$ (library) $(Rno, S/date, E/date$ and $Rno = Rno_1) \}$

Transaction:

Need for transaction:



here C_2 gets wrong amt since wind 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 properties:

- 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 opns 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:

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

- concurrency control protocol will ensure the

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 committing '600' can be aborted.

timing is imp. here.

② Non repeatable values

read write conflict.

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 Trans1
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	10.05	St Tx2
10.10	set M sal=600	10.06	sum(sal) 1600
10.20	C	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 TX2	
10.16	Set M = sal = 700	
10.17	C2.	

⑥ Unstable control

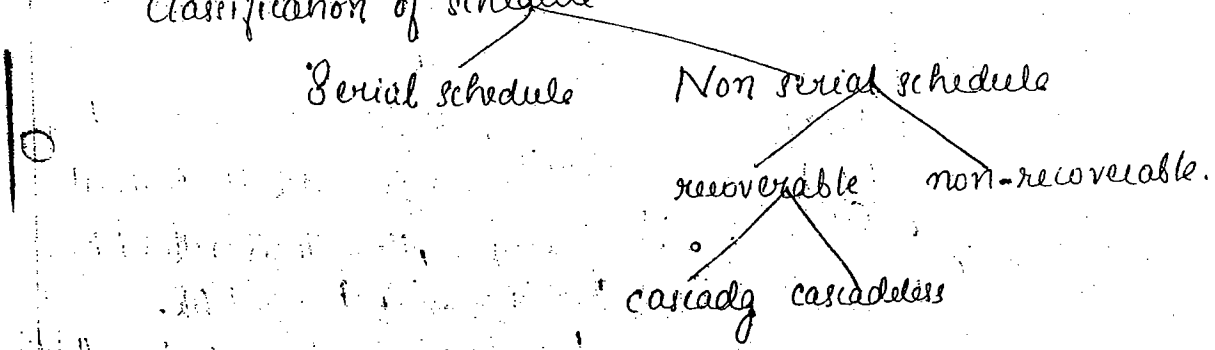
10	St TX1
10.10	Drop col sal
10.15	C1
10.05	St TX2
10.06	Read sal ✓
10.20	Read sal X

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:



T1

T2

S1	S2	
T1	T2	Serially from T1 then T2
T2	T1	S2 " " T2 then T1

ie we can have n! schedules posble

T_1 } if one is abk, it stays at halt
T_2 }

so we look for non serial

T_1
T_2
T_1
T_2

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 $= \frac{(5+3)!}{5! * 3!} = 56$

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 ✓	

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 optn, 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_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 leading 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 "

Rec).

① O/p result of any transaction heavily depends on ~~&~~ initial values of data items. For some initial values o/p ~~of~~ may be same, ~~the~~ 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, can't swap opn. So can't consider it as conflict opn.

$X_i(A)$, $Y_j(B)$

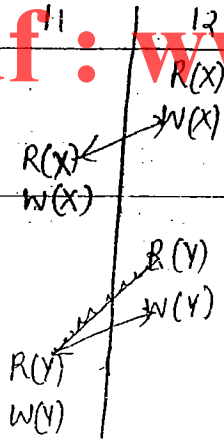
① $A = B$
② $i \neq j$
③ X or Y must be write opn.

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)



S_2 is conflict equi to S_1
(non serial) (serial)

view

Rules for serializability

a) A non serial schedule S^2 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: $F_1 = A \to B$, $A \to C$ $\quad F_2: C \to D$

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

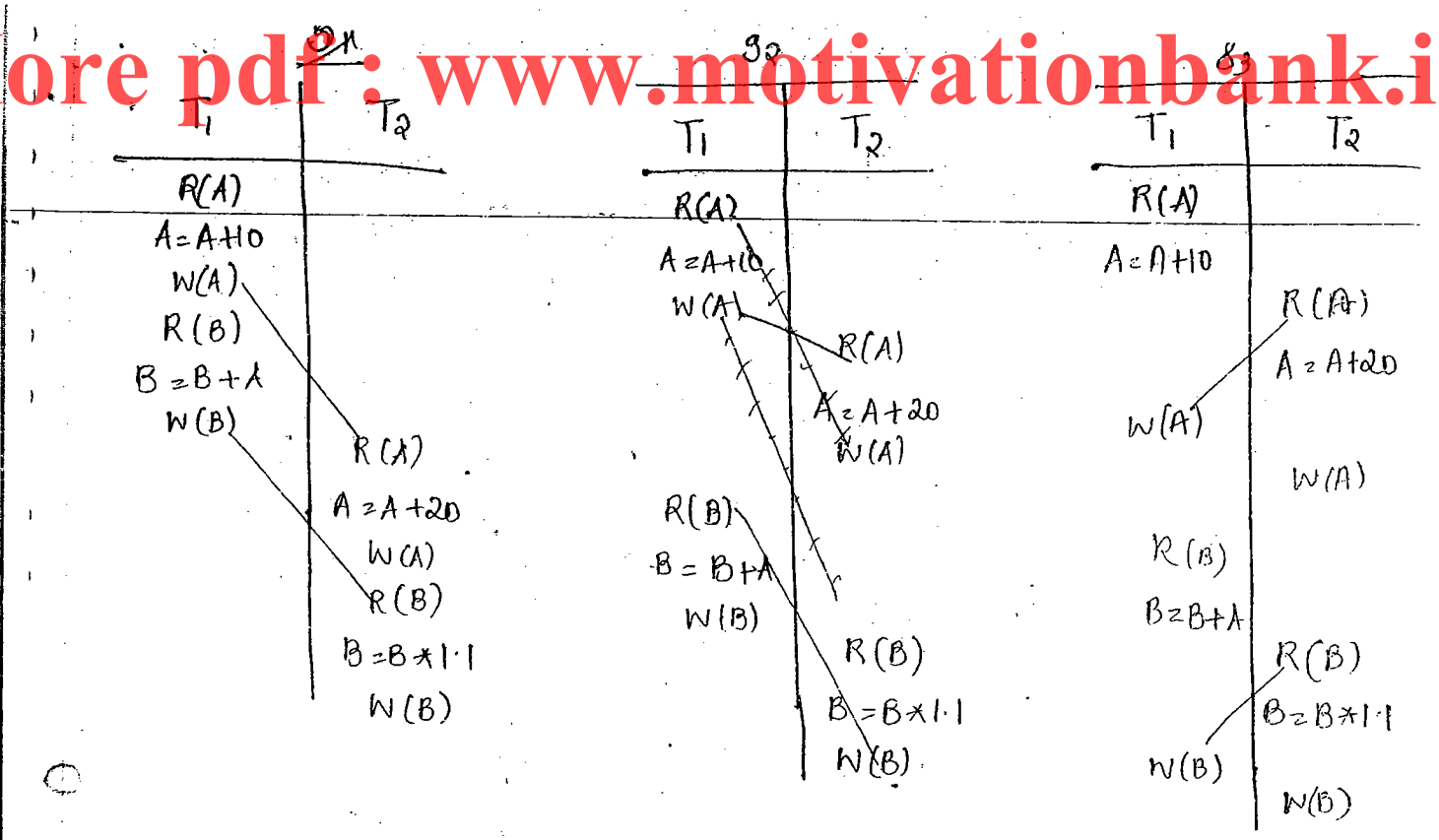
Eg 2: $A \to B$, $A \to C$, $C \to D$ $\quad R = (A, B, C, D)$

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

lossless join not satisfied

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

$A \to B$ ✓



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 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.

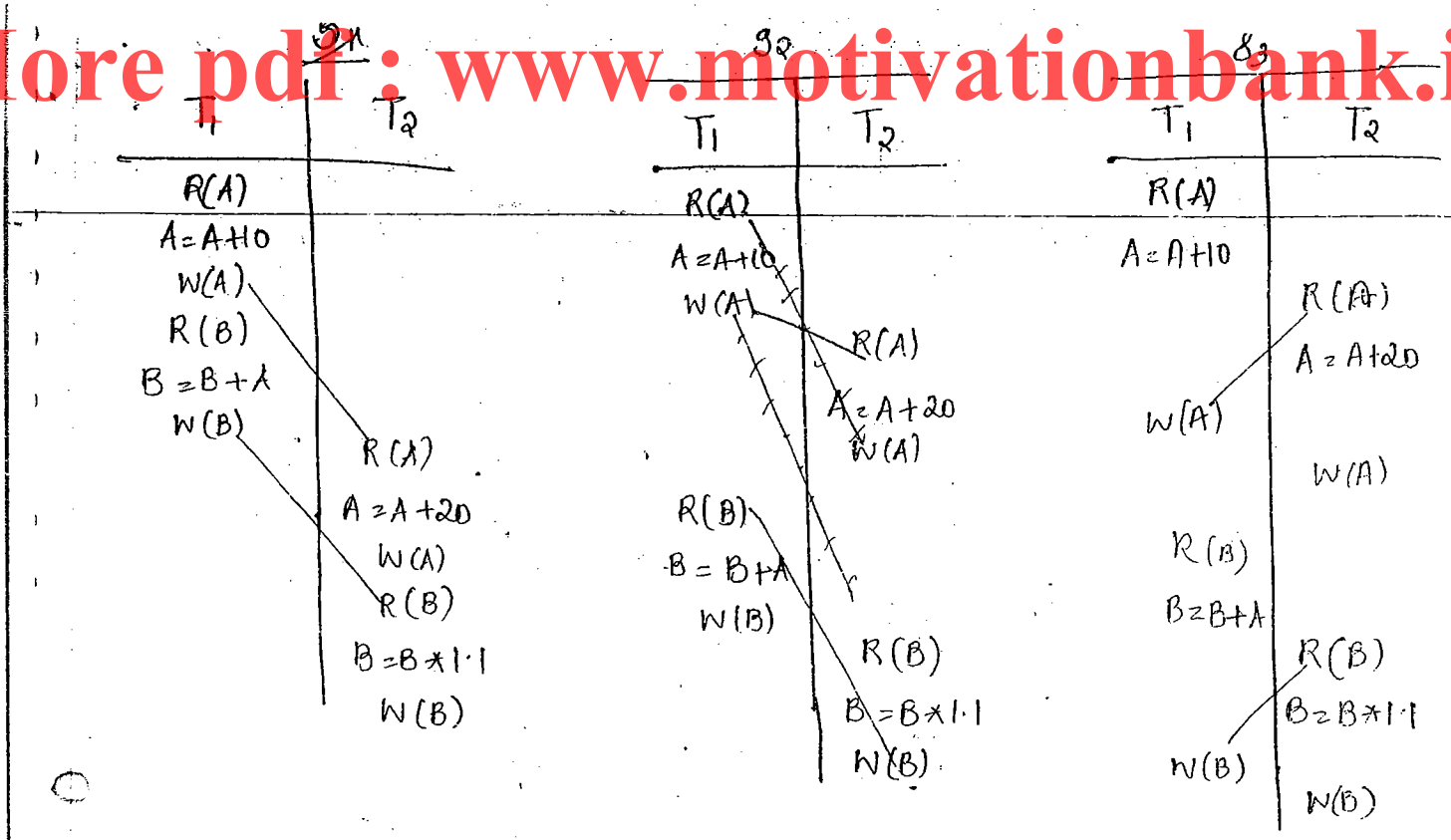
3) Majority of concurrency control protocols are based on conflict serial. except Thomas Write rule.

S_1	
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
$W_1(X)$	$W_2(X)$
$W_1(X)$	$W_2(X)$
$W_2(X)$	$W_1(X)$

S_3 is view equivalent to S.

but S_3 is not conflict equiv- to S.



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 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 serializ. except Thomas write 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.

Q/3. a) strict schedules — view, recoverable and cascadeless
serial schedule is strict.

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
$R(X)$	
	$R(X)$
$W(X)$	
	$W(X)$

T_1 T_2 ✗

T_2 T_1

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

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

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

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

not view serializable.

R_2 reads value written by R_1

— 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)*

$C_2 < C_1$ ✗ recoverable
$C_1 < C_2$ recoverable.

since it has dirty read, it is cascading schedule.

Depending on commit recoverability is decided.

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

$T_1 < T_3$ $\quad$ $T_3 < T_2$

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)

$C_3 < C_2$ ⟶ recoverable.

$C_2 < C_3$ ⟶ not recoverable.

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

$T_1 < T_2$ $\quad$ ~~$T_2 < T_1$~~

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

$T_2 : R(y)$ $T_3 < T_2$ $T_2 < T_3$

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

$T_2 < T_3$

$T_2 : W(Z)$, $T_2 : C$

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

$T_1 < T_2$ $T_2 < T_1$

not conflict equivalent.

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

cascading schedule.

R(X)

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. hoo.

cascadg 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.

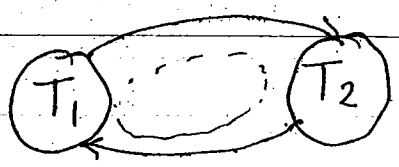
No. of edges in gph = no. of conflict opns in the schedule.

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

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

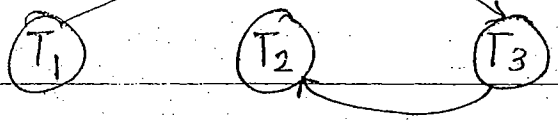
① conflict already ① is added so taken care

$T_2 < T_1$ $T_1 < T_2$ not conflict equalent.



not conflict equivalent.

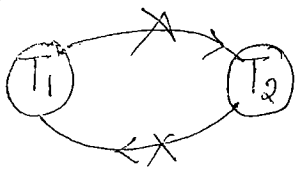
b) $R_1(X)$ $R_2(Y)$ $W_3(X)$ $R_2(X)$ $R_2(Y)$ C_1 C_2



not conflict equivalent.

conflict equivalent to $T_1\ T_3\ T_2$

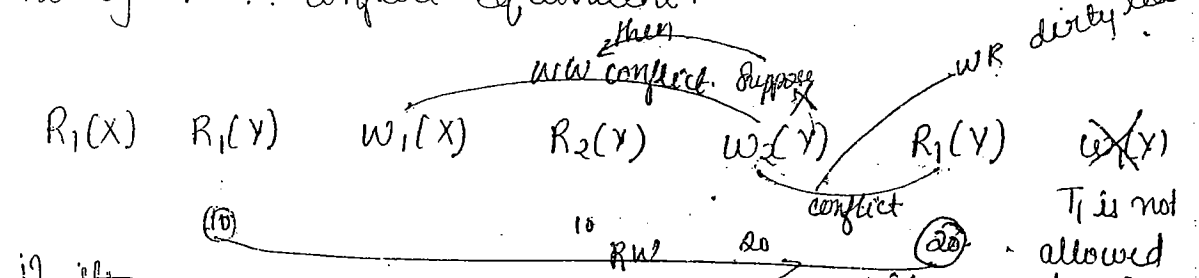
d) $W_1(X)$; $R_3(X)$, $W_1(X)$, C_2 , C_1



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

(i) no cycles ∴ conflict equivalent.

19/2. $R_1(X)$ $R_1(Y)$ $W_1(X)$ $R_2(Y)$ $W_2(Y)$ $R_1(Y)$ ~~$W_2(Y)$~~



T_1 is not allowed to write Y

~~if it~~

W-R conflict present.

~~Non repeatable reads~~

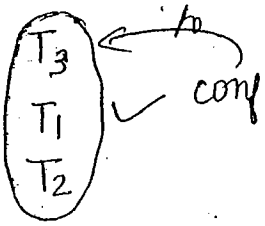
R-W conflict.

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

99/1

(i) $r_1(x)$ $r_3(x)$ $w_1(x)$ $r_2(x)$ $w_2(x)$

$T_3 < T_1$ $T_1 < T_2$



conflict equivalent

Classification of concurrency control protocol

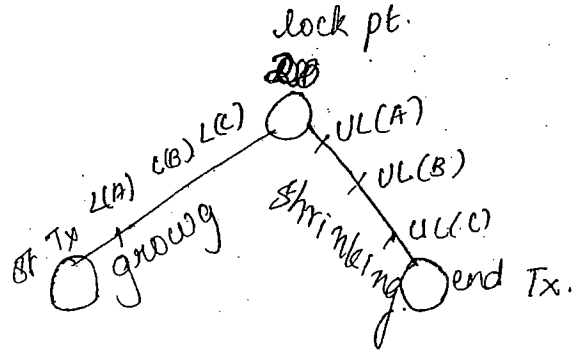
1) log-based protocol.
   a) 2 phase locking
      (i) B2PL
      (ii) C2PL conservative
      (iii) S2PL
      (iv) R2PL rigorous
   b) Graph based

2) Time stamp based protocol
   a) Timestamp ordering protocol
      a) ~~Fitho~~ Thomas Weight.

3) Multiple granularity protocol.

4) multi version protocol
   a) MV two phase prob.
   b) MV timestamp ordering protocol.

Locks
Shared – R.
Xclusive – W.



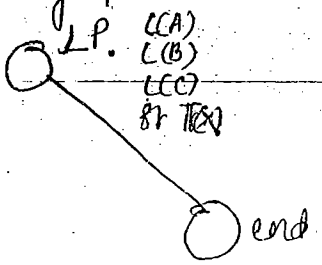
all locks in one phase
after unlock no locking.
ie all unlocks together.
growing – locking
shrinking – unlocking.

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.

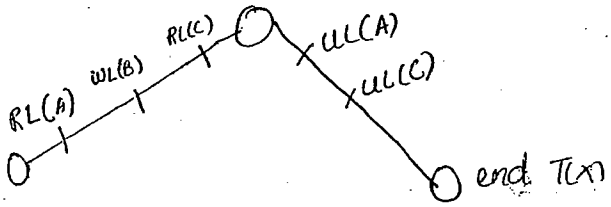


If A9 items all locked I waiting then all cant start.

if items more go for B2PL

3) Strict 2PL

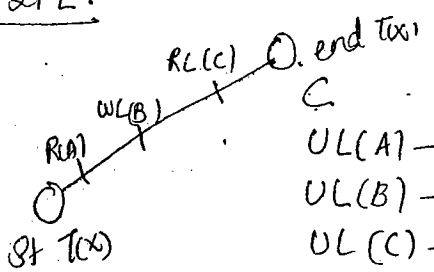
St T(x)



Commit
UL(B)

unlock write locks only after committg.

4) Rigorous 2PL:



C
UL(A) — R
UL(B) — W
UL(C) — R

③ & ④ are used nowadays.

---

eg:

St T_{x_1}
$R_1L(A)$
St T_{x_2}
⋮
$W_2L(B)$ ⋯⋯⋯⋯ it is strict 2PL.
UL(A)
C1
end T_{x_2}

C_2
UL(B) after committing

disadv : ① deadlocks
② Starvation

A	RTS	9.00
	WTS	9.00

T_3.

T_{x_1} — 9.00
⋮
R(A) — 9.30
⋮
W(A) — 10.00
End T_x — 10.30

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.

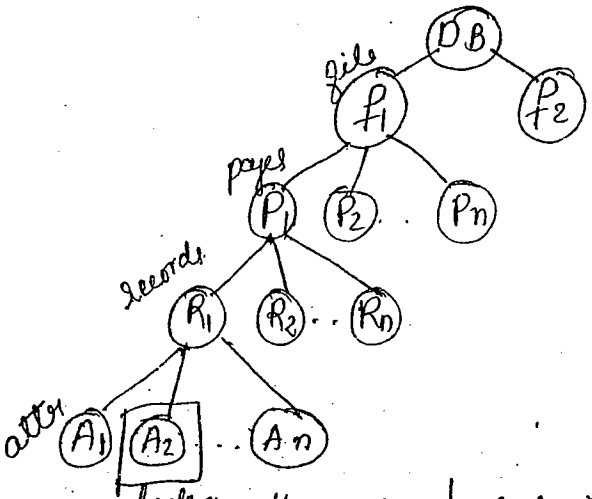
8.50	9.00
I_1	T_2
$W_1(X)$	
	$W_2(X)$
$W_1(X)$ ←	

is also allowed.

TS → conflict serializable
Thomas right — view serializ.

Multiple Granularity protocol

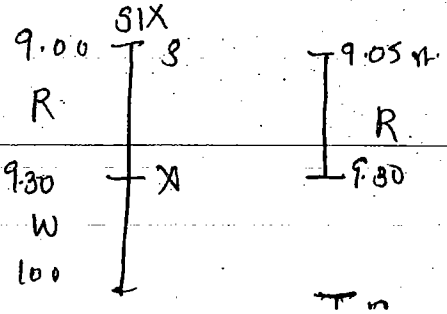
deals with size of db.



Lock on roll no. only for particular roll no

Select * from student then lock on file. F1.

SIX condition that T_2 shd execute when T_1 makes it X



XPS is meaningless

Consider a db with 2 files f_1 and f_2 and f_1 consists pg 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 → page no. R → record no. For each of the following opns specify the sequence of blk request

① Read records from $P_1.98$ to $P_2.2$.

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

But if $P_1.50$ 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 $P_2.2$
S on $P_2.1$
IS on P_1
S on $P_1.100$
S on $P_1.99$
S on $P_1.98$

good method.

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

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 $P_{50.1}$ to $P_{500.1}$

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

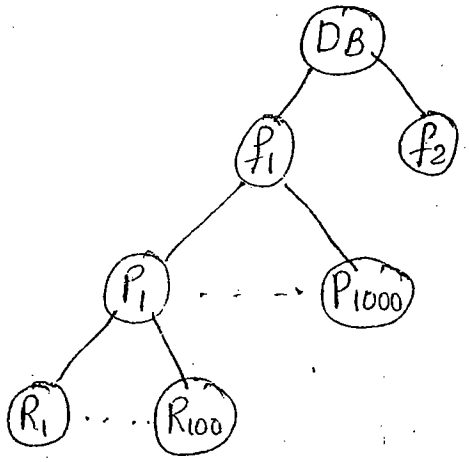
Locks

4) Delete first record in each page.

to delete - exclusive locks needed.

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
①	②	③

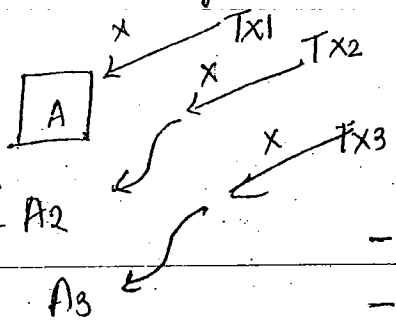
Q: Select * from student. Assume all student recs are in f_1.



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 possible
- 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 making copy takes longer time than getting actual lock on that file, then its disadvantageous.

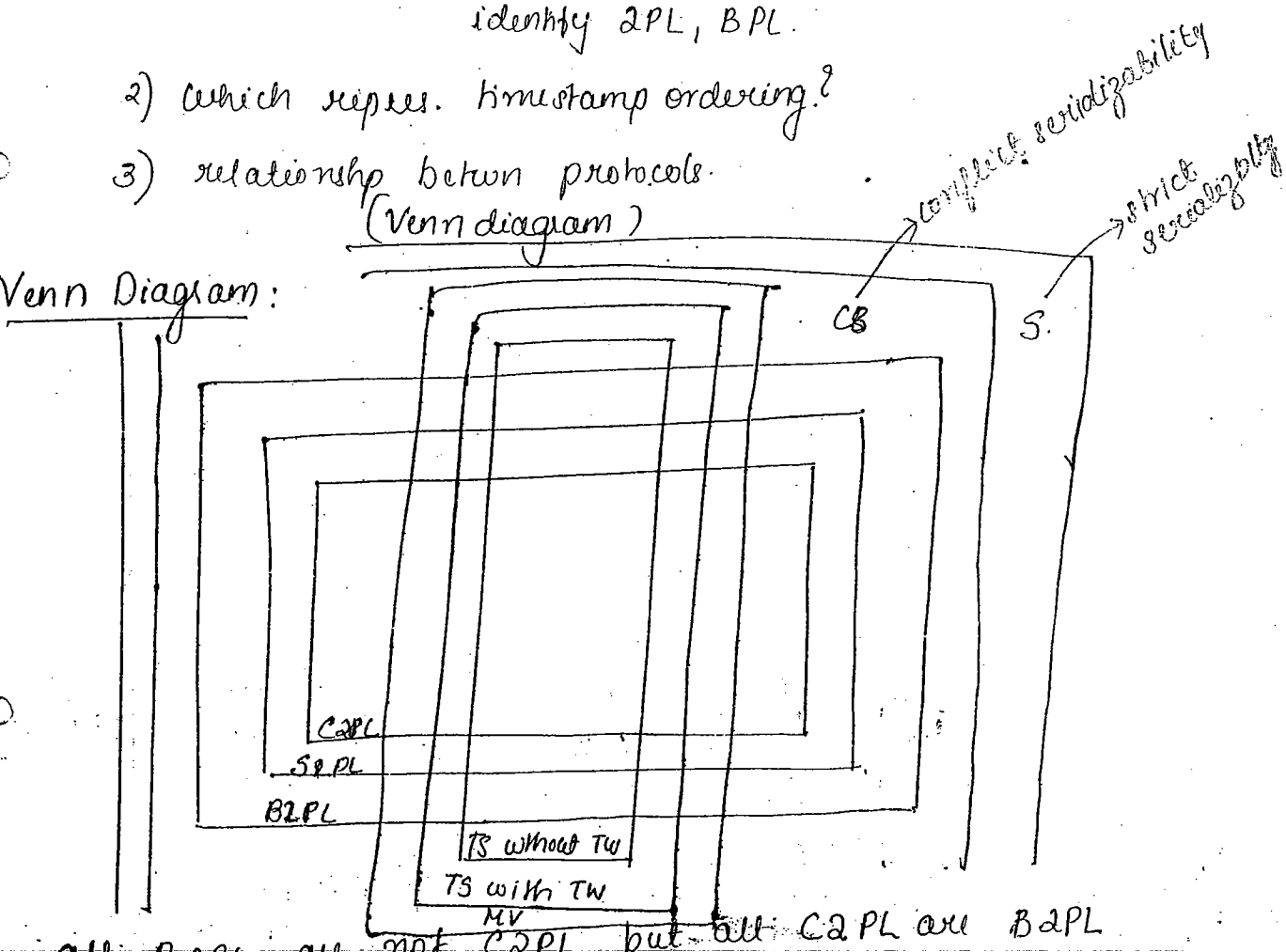
From files onwards to DB, multiversion is not advan. because it is time consuming.

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

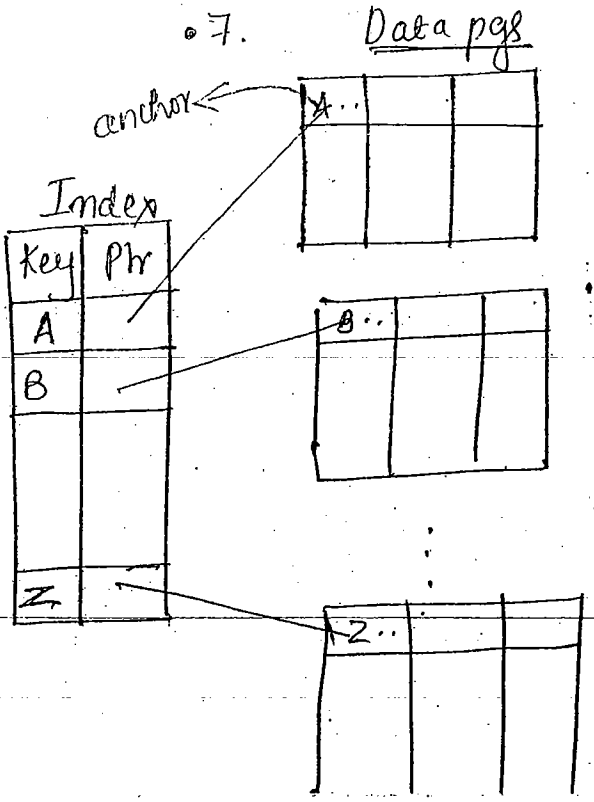
① It is created on primary key of a table hence there shd be only 1 primary index per table.

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.

(6) 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+6 = 15 B

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

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

Case @ Search operation reqd without index

No. of searches reqd = $\log_2 3000$
(binary)
= 12

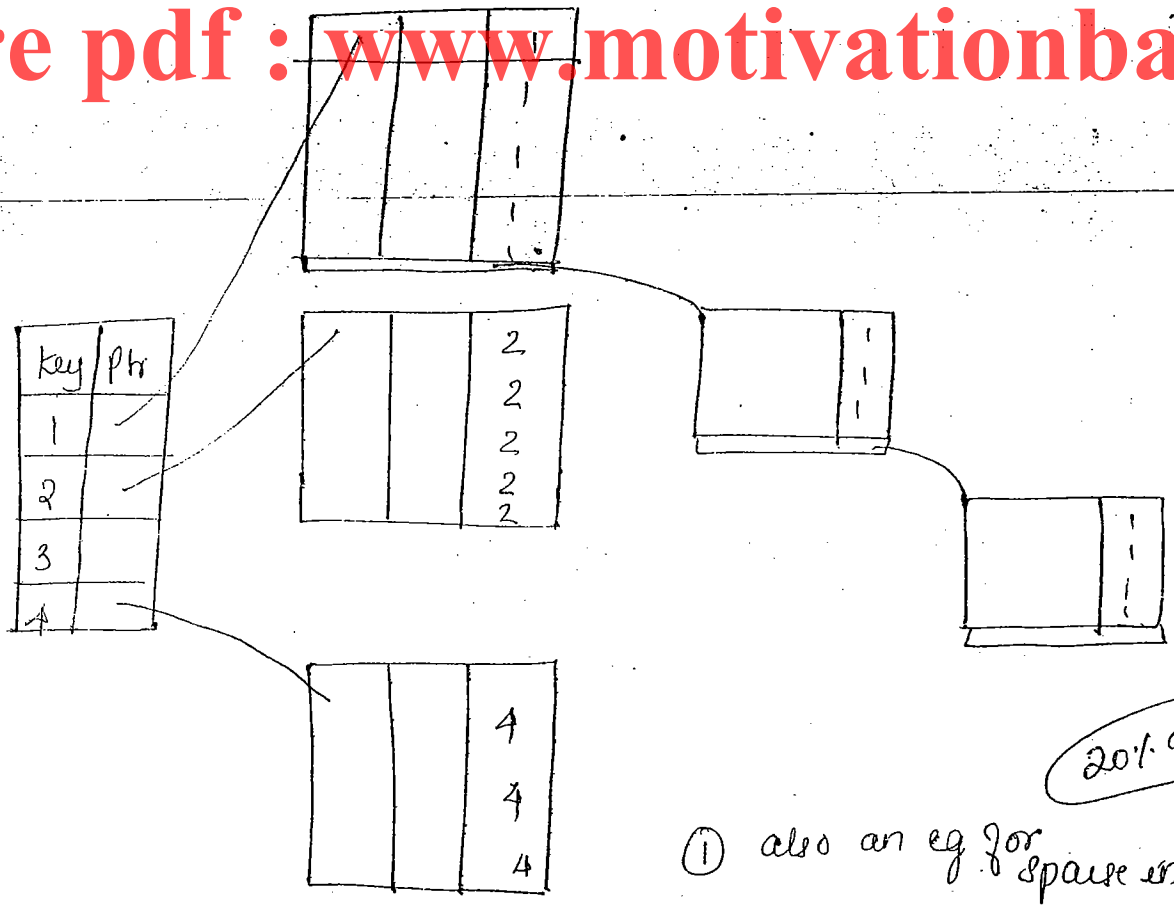
No. of searches with indx pgs = $\log_2 45$ = 6.

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 indx 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 col.

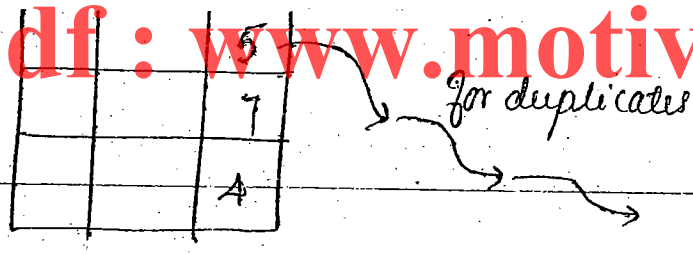
② It is an example for dense indx.

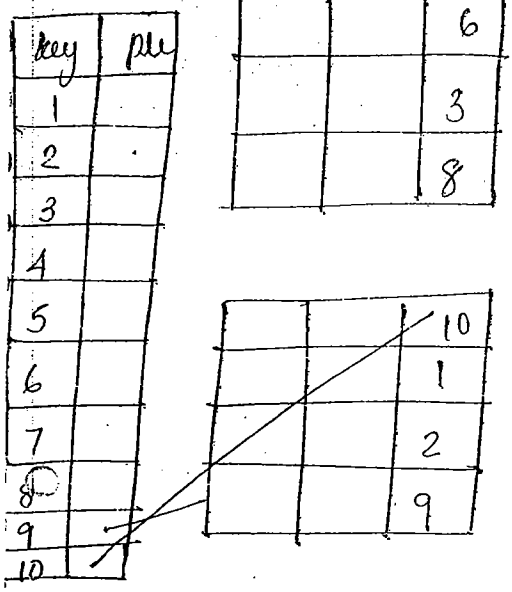
③ Each index consists 2 attr.

- ① Search key value
- ② Record ptr

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

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

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

∴ No. of indx rec = 30,000

key = 9 B

ptr = 6 B

len of each record = 9+6 = 15 B

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

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

Here linear search,

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

b) with indxg = $\log_2 450 = 9$

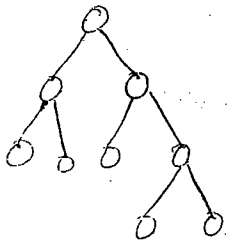
most effective indxg
- 2° indx

B and B+ trees are multilevel indexes.

most useful indxg
- 1° indx

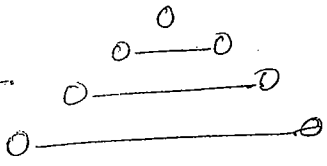
Sol: Why we use B & B+ in db?

Binary

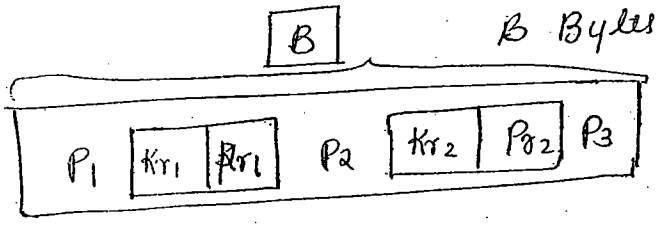


levels are more.
grows vertically

B tree



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



P → blk ptr
K_v → key value
P_r → data ptr.
B → size of pg
n → order of pg

$$nP + (n-1)K_v + (n-1)P_r <= B$$

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

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

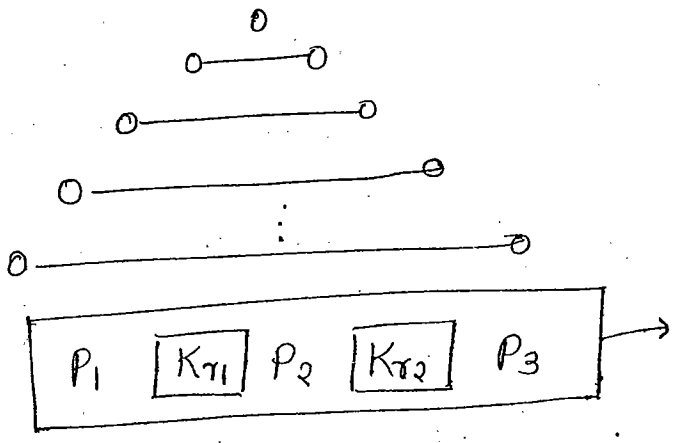
n = 23

fill factor = 0.65

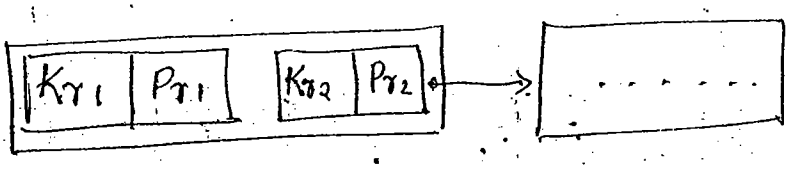
n = 16

	Nodes	ptrs	Data
R	1	16	15
L_1	16	16x16 = 256	16x15 = 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 B+ trees.



10



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

P=6 , K=9

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

n = 32

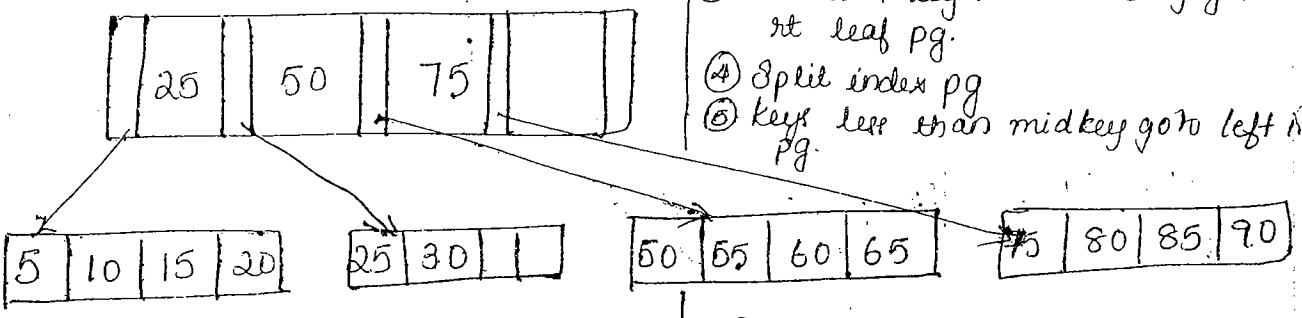
fill factor = 0.65

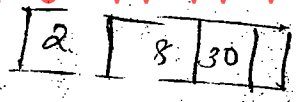
n = 23.

	nodes	ptr	Data
R	1	23	22
L1	23	529	506
L2	529	12,167	11638
L3	12,167	2,79,841	2,67,674

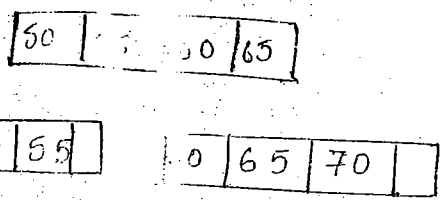
Inserting data item:

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 goto rt leaf pg. ④ Split index pg ⑤ keys less than midkey goto left idx pg. ⑥ keys greater than midkey goto rt index pg. ⑦ Midkey goes to next higher level of index.

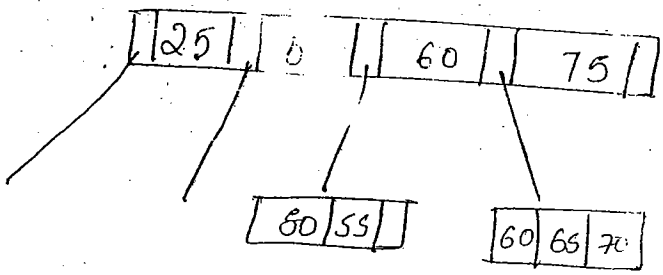




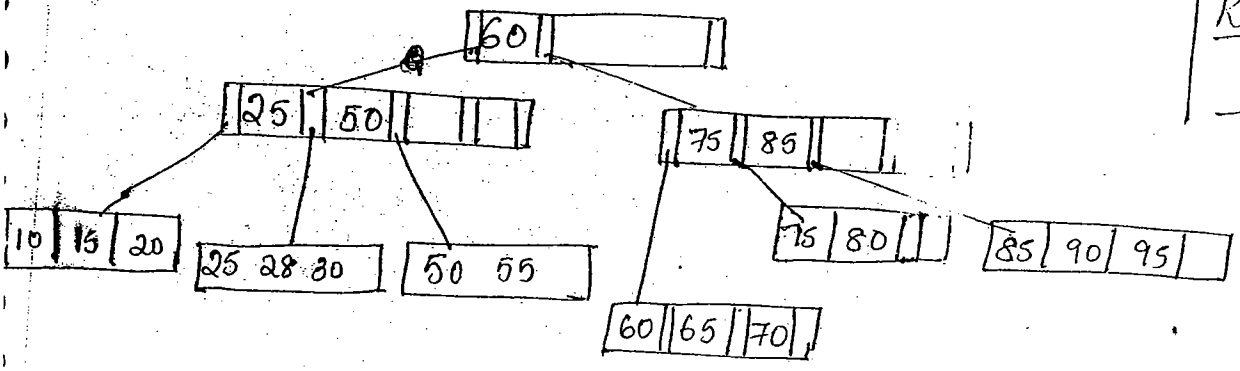
2: add ... with key value 70.



push 60 to root



3: add 95.



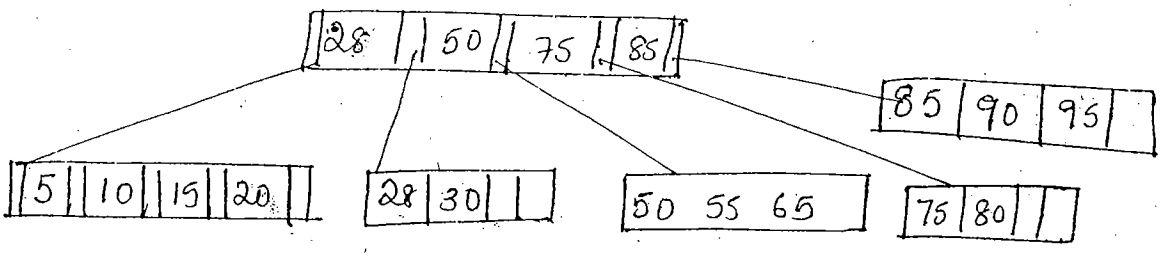
Root > only

>= 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 &
ERdiag