DBMS {7-9 marks}

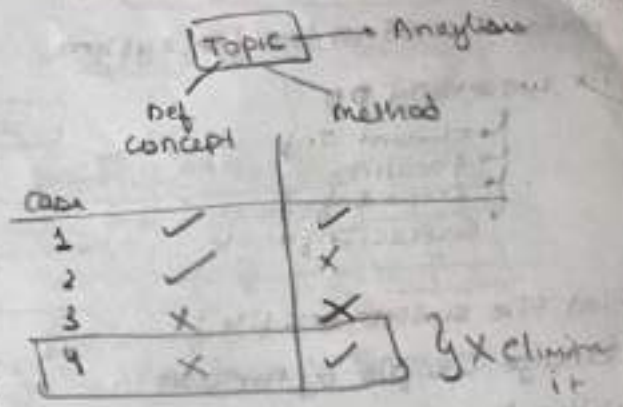
Milin Marathe

2 time note — 4-5Q WB/PYQ

⇒ Syllubus :-

1. Integrity & ER model {1-2
2. Normalization {2-4
3. Queries {4
   - RA, SQL, TRC
4. File Org. & Indexing {2-4
   [B/B+ tree index]
5. Transactions & concurrency control. {2-4

{Ravi Kumar 8074172708}



Complete DB concept
- Ullman (website)
  - {PPT}
  - Assignment
  - Sol & Ex problem
- Korth
- Carmen

ACID properties

** Serializable
   - View serializable
   - Conflict serializable

* → Recoverable [IRREC/REC/cascadeless/strict]

→ Concurrency control protocol
1. locking protocol
2. time stamp ordering protocol

SQL: Commerical Rel. Query lang.

formal Rel. query: {Rel. algebra, TRC, DRC} Rel. calculus

TRC > Rel. Alg.
↓
bcoz unsafe queries

- TRC
- 2PL & TimeStamp protocol.
- secondary index with nonkey.

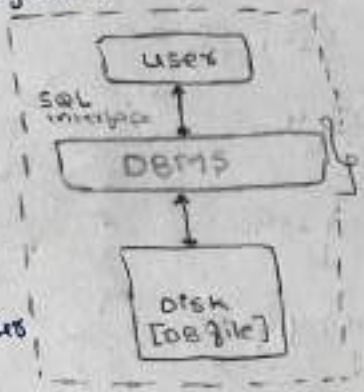
Introduction:

Data Base: collection of related data stored in the file system

i.e University DB
- Student Inf.
- faculties
- Emps Inf.
- Course Info. etc

Application

DBMS: S/w used to manage DB
- access DB files
- create DB file
- Insert data
- delete/update etc

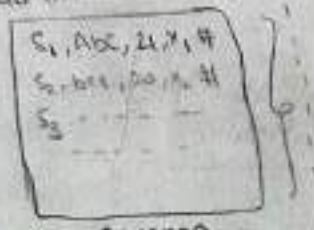


Interface b/w users & DB file

Flat file system [os file]

Manage DB file by the users without DBMS S/w

Stud Int



2000
450

comma seperated file

I/O cost: 2000 pg Sorts

Small DB: flat file system

Huge DB: failed to manage
[GB's by using flat files
TB's]

Limitation of flat file system [os file]

1) Too complex to manage applicatn program
2) More I/O cost [Access cost] to access required data.
3) less degree of Concurrency Control [Degree of Concurrency control: # of users allowed to access data simultaneously]
4) Too complex to manage non-redundant data.

Advantage of DBMS file System

1) Easy to manage DB files b/c Storage info. of the file under control of DBMS S/w.
2) b/c of Indexing to the Data of DB file reduce I/O cost to access required data [less I/O cost]
3) more degree of concurrny
4) b/c of normalization of DB table easy to manage non-redundant data

why call a table, a Relation?

"Relation" (in mathematics)

binary Relation $R: A \to B$ means $R \subseteq A \times B$

$A \times B = \{(a,b) / a \in A, b \in B\}$

ternary Relation: $R \subseteq A \times B \times C$

$A \times B \times C = \{(a,b,c) / a \in A, b \in B, c \in C\}$

(a,b,c) — order-triple (3-tuple)

n-ary Relation:

$R \subseteq A_1 \times A_2 \times \cdots \times A_n$

$A_1 \times A_2 \times \cdots \times A_n = \{(a_1, a_2, a_3 \cdots a_n) / a_i \in A_i\}$

so, $r_1 \in A \times B \times C$ (row) r_1

these combination will always be unique as it's set

A	B	C

r_2 r_3 r_4

Instance $\subseteq A \times B \times C$

Set of all records.

Table $\subseteq D_1 \times D_2 \times D_3 \times \cdots \times D_n$

Instead of these we will use Relation

there is the Correct word

Conclusion

Table R:

Domain D_1	Domain D_2			D_n
A_1	A_2	A_3	...	A_n

① every Record/Row r of R

$r \in D_1 \times D_2 \times \cdots \times D_n$

$(a_1, a_2, \cdots a_n) \in D_1 \times D_2 \times D_3 \cdots D_n$

ex: assume $|D_1| = 2 / |D_2| = 3 / |D_3| = 3$

① Cardinality of largest instance: no. of record

$I = D_1 \times D_2 \times D_3$

$I = 2 \times 3 \times 3$

$I = 18$

② no. of instance possible 2^{18} → every instance has 2 choice

② every Instance I of R

Set of Records

$I \subseteq D_1 \times D_2 \times \cdots \times D_n$

$R \subseteq D_1 \times D_2 \times \cdots \times D_n$

content of the table will be subset of $D_1 \times D_2 \times \cdots \times D_n$

Table ≡ Relation

Set of Record

unorder & no duplicate

- order of tuple doesn't matter
- order of attribute are fixed

ex: which of the following is a possible set of all C.K in a relation R(a,b,c,d,e,f)

(A) {a,bc} (B) {} (C) {abcde} (D) {a,b,c,d,e} (E) {ab,bc}

C.K can't be empty

(D) {a, a, b, bc}

if C.K → can't be C.K if (A) is C.K

ex: entity integrity constraint
no primary key value can be Null

ex:

A	B	
null	2	✗
null	null	✗
2	2	✓

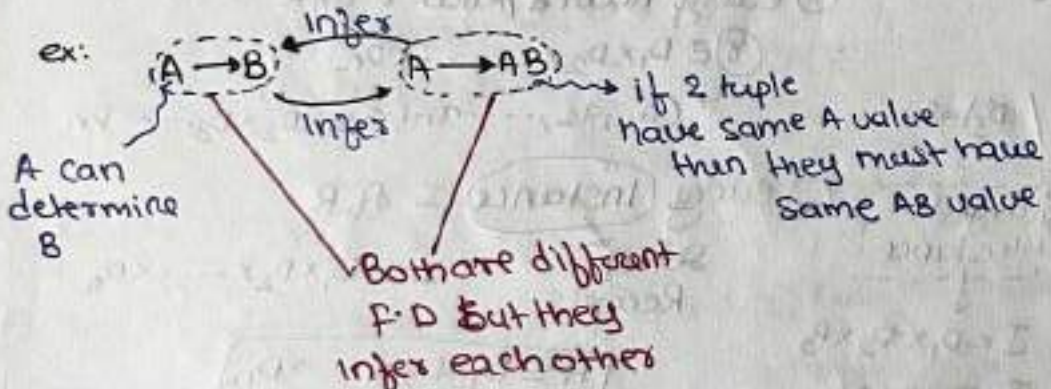
• doesn't say anything about uniqueness of primary key

• Primary key Integrity Constraint

↳ should be unique

note: if X is S.K (or) C.K & $y \to x$
then y is S.K
we can't say that y is C.K
iff y is minimal then y: C.K

ex:



• DBMS Integrity constraints are set of predefined rules used to maintain the quality of information

LIMITATION'S of flat file system

5) Too complex to maintain different access control to different user based on role of user.

Adv. of DBMS file system

5) b/c of VIEW's (virtual tables) easy to maintain access control based on role of user

Both file system are
- Based on large file DB

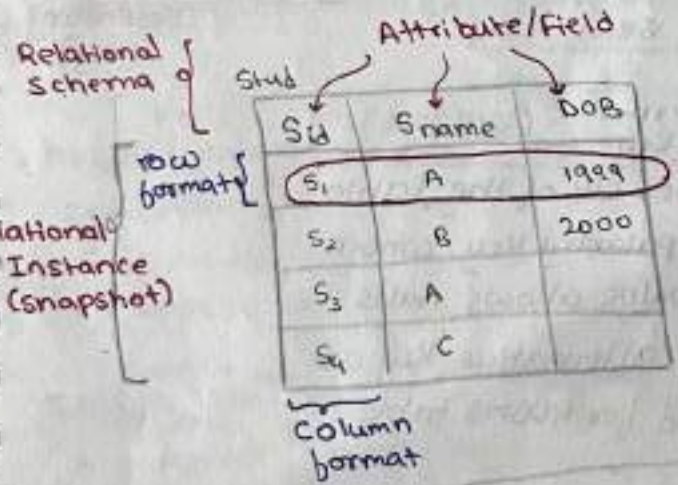
VIEW: virtual tables

Integrity Constraint :- (based on RDBMS)

Data Models
- RDBMS (widely used data model. (E.F. codd))
- ODBMS
- NW DBMS
- Hirarchial DBMS

Relational DBMS (RDBMS) [also called Codd's data models] → 1* Data of the DB file must be stored in tabular format [set of Rows column's]
2* No two row's of RDBMS table must be same.]

- codd proposed 12 Rules to design RDBMS s/w (RDBMS Guidelines)



Arity (Degree): no. of attribute of the table

Cardinality: no. of record of the tables

3* All Attributes of RDBMS table must be Atomic/single valued

Relational Schema Definition or Structure of RDBMS table

Relational Instance set of record of RDBMS table (snapshot of the table)

note: order of Attribute & record/Tuple doesn't matter

Candidate key: Minimal attribute set which can differentiate all record of the relation uniquely

[minimal Attribute set (a minimal Super key) + unique value for all records]

[but proper subset should not be C.K]

Ex: Stud (Sid, Sname, DOB)

C.K : {Sid} ✓

C.K : {Sid, Sname} ✗

Ex: Enroll (Sid, Cid, fee) (non-null value)

S_1	C_1	-
S_1	C_2	-
S_2	C_2	-
S_3	C_2	-

R

A	B	C
4	2	3
4	2	5
4	6	3
4	6	5
6	2	3

C.K: ABC

[Stud can enroll many Courses, course can enroll many stud's]

Candi-Key : {SidCid}

NULL : unknown / unExisted value / Notapplicable

Emp

eid	ename	DOB	panID	AdharNo	IFSC	A.no
e_1	A	1995	x_2		SBI01	101
e_2	B	1995	Null		CAN01	101
e_3	A	Null	x_5		CAN02	102
e_4	B	1992	Null		ICIC01	102
e_5	C	Null	x_1		SBI01	102

Candi-Key : {eid, panID, AdharNo, IFSC A.no}

AdharNo. → Primary Key

IFSC A.no → Alternative Key (secondary Key)

Primary Key	Alternative (UNIQUE) Key
1. any one C.K whose field value guarantee not null is called primary key (field value's not allowed NULL's)	1. all other C.K of the table except primary key whose field value allows nulls
2. atmost one Primary Key allowed for any RDBMS table	2. many alternative Key are allowed for RDBMS table → many [one or more]

3. If P.K is [multiple attribute] then (no) attribute of PK can have null value.

```
Create table emp
(   eid varchar(10) primary key,
    ename varchar(30) NOT NULL,
    DOB date,
    PanID Varchar(10) UNIQUE,
    adharNo Integer(12) UNIQUE NOT NULL,
    Ifse Varchar(6),
    Ano. Integer(10),
    UNIQUE (Ifse, Ano));
```

minimal super key

if CK = {A1, A2, ... An} then
∀i (CK − Ai) is not a Super Key.

Prime Attribute:-

Attribute belongs to (some / atleast one) C·K of relation

Emp (eid, ename, DOB, PanID, AdharNo, Ifsc, Ano)

Cand. Keys = {eid, panID, AdharNo, Ifsc, Ano}

Prime Attribute set of emp relation = {eid, PanID, AdharNo, Ifsc, Ano.}

nonprime Attribute set of emp. relation = {DOB, ename}

Simple Candidate Key:-
C·K with only one Attribute

Composite Candidate Key:- C·K with atleast 2 Attribute

- Atleast one of the C·K of the relation must be no null value

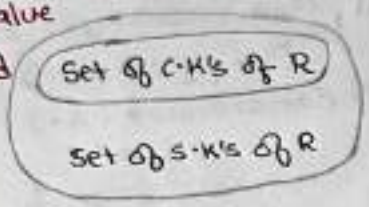
C·K with NO NULL
- Primary Key
- UNIQUE NOT NULL

Allowed

```
CREATE TABLE R
(A integer(3) Primary Key,
 B integer(3) UNIQUE,
 C integer(3),
 D integer(3),
 E integer(3),
 UNIQUE(C,D),
 UNIQUE(D,E))   -- 2 P·K
);
```

```
CREATE TABLE R
(A int(3) UNIQUE NOT NULL,
 B int(3) UNIQUE,
 C int(3),
 D int(3),
 E int(3),
 F int(3),                 (NO P·K)
 UNIQUE (C,D),
 UNIQUE (D,E)
);
```

Super Key:- Set of attribute used to differentiate all records of relation uniquely. (non-null values)
[may not minimal Attribute set]
↳ C can be sigleton also)
↳ null value are allowed



ex- Stud (Sid, Sname, DoB)

S_1	A	2000
S_2	A	2000
S_3	B	2001
S_4	B	2002
S_5	C	2002

C·K: {Sid}
Super Key: {Sid}, (minimal SK [C·K])
SidSname,
SidDoB,
SidSnameDoB }

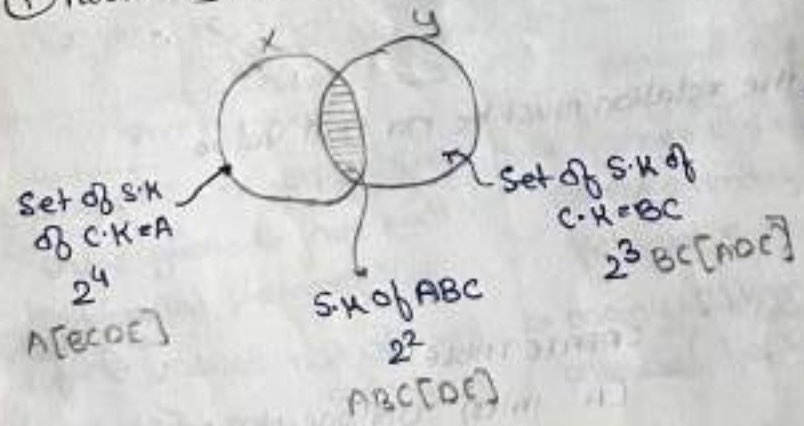
Ques] R(ABCD) How many S·K's in R if
C·K {A}?
$2^3 = 8$ [A, AB, AC, AD, ABC, ABD, ABC, ABCD]

A· [any Subset of (B,C,D)] ↔ S·K of R
2^3

R($A_1, A_2, A_3 \dots A_n$)
How many S·K's in R if {A} C·K?
2^{n-1} S·K

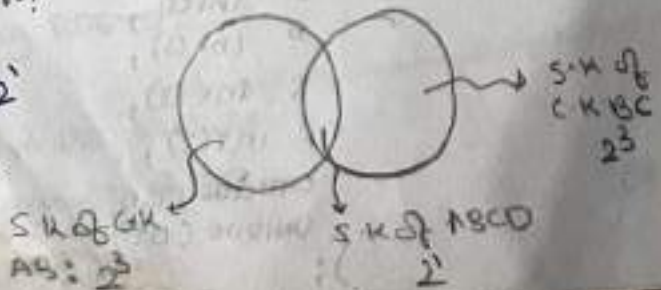
How many S·K in R if {A_1, A_2} C·K
2^{n-2} S·K

Ques] R(ABCDE)
(i) How many S·K in R's if {A;BC} C·K



$\Rightarrow 2^4 + 2^3 - 2^2$
$\Rightarrow$ 16 S·K

(ii) {AB, CD} C·K?
$\Rightarrow 2^3 + 2^3 - 2^1$
$\Rightarrow$ 14 S·K



Ques] R(A
Hasmani
(1) {A,A
$\Rightarrow 2^n$
$\Rightarrow$
Ques] R
How
(1)
2^n
$\Rightarrow$

Ques] R(A, A2 A3 ---- An)

How many S.K's in R if

(i) {A1, A2, A3 A4} C.K

$\Rightarrow 2^{n-2} + 2^{n-2} - 2^{n-4}$

$\Rightarrow \boxed{7 \cdot 2^{n-4}}$

(ii) {A1, A2, A2 A3} C.K?

$\Rightarrow 2^{n-2} + 2^{n-2} - 2^{n-3}$

$\Rightarrow \boxed{3 \times 2^{n-3}}$

(iii) {AA2, A3} C.K

$2^{n-3} + 2^{n-1} - 2^{n-3}$

$\Rightarrow \boxed{5 * 2^{n-3}}$

Ques] R(ABCDEF)

How many S.K in R if

(1) {A, BC, CD} C.D

$2^{n-1} + 2^{n-2} + 2^{n-2} - 2^{n-3} - 2^{n-3} - 2^{n-3} + 2^{n-4}$

$\Rightarrow 2^5 + 2^4 + 2^4 - 2^3 - 2^3 - 2^3 + 2^2$

$\Rightarrow 32 + 16 + 8 - 8$

$\Rightarrow 52 - 8$

$\Rightarrow (44)$

(2) {AB, BC, CDE} CK?

$2^4 + 2^4 + 2^3 - 2^3 - 2^2 - 2 + 2$

$\Rightarrow 32 - 4$

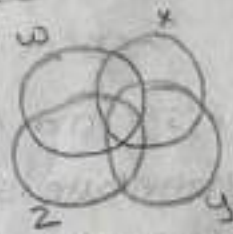
$\Rightarrow (28)$

(3) {A, B, C} C.K's

$2^5 + 2^5 + 2^5 - 2^4 - 2^4 - 2^4 + 2^3$

$\Rightarrow 32 + 16 + 8$

$\Rightarrow (56)$



$n(w \cup x \cup y \cup z)$

$\Rightarrow n(w) + n(x) + n(y) + n(z) - n(w \cap x) - n(w \cap y) - n(w \cap z) - n(x \cap y) - n(x \cap z) - n(y \cap z) + n(x \cap y \cap z) + n(w \cap x \cap y) + n(w \cap y \cap z) + n(w \cap x \cap z) - n(w \cap x \cap y \cap z)$

Ques] R(A, B, C, D, E)

How many SK in R if

C.K {AB, BC, CD, DE}?

Method 1:

AB	BC	CD	DE
ABC	BCD	ACD	ADE
ABD	BCE	COE	BDE
ABE	BCDE	ACDE	
ABCD			
ABCE			
ABDE			
ABCDE			

19 S.K

method 2:

$\Rightarrow 2^5 - 1 - 5C_4 - (5C_2 - 4) - (5C_3 - 9)$

$\Rightarrow 32 - 1 - 5 - 6 - 1 = \boxed{19 \text{ S.K}}$

$\phi \quad 1 \quad 2 \quad 3$

H·W Q. R(ABCDEF)

{AB, BC, CD, DE, EF, BE, BF, CE, CF} C·K

How many C·K's

$2^6 - 1 - 6 - {}^6C_2 - 9$

(ϕ, 1, 6, $^6C_2$)

AB
ABC
ABD
ABE
ABF
ABCD
ABCE
ABCF
ABDE
ABDF
ABEF
ABCDE
ABCEF
ABDEF
ABCDEF

BC CD DE EF
BCD CDE DEF CFA
BCE CDF DEA EFC
BCF CDEF
BCDE
BCDF
BCEF

BE
BDE
BEF

Ques] R(A1 A2 A3 ------ An)

How possible S·K's in R?

→ $2^n - 1$ (ϕ)

∴ max possible S·K in a relation of 'n' attribute is [if each attribute of relation is C·K]

worst case:
∴ min possible S·K in relation with 'n' attribute is '1' [all attribute of the relation combinly form C·K]

Ques] R(A,B,C,D,E)

How many max possible C·K in R?

5 Attribute [ABCDE]

max { $5C_1 = 5$ {A, B, C, D, E}; $5C_2 = 10$ {AB, AC, AD, AE, BC, BD, BE, CD, CE, DE}; $5C_3 = 10$ {ABC, BCD, DEA, ...}; $5C_4 = 5$ {ABCD, ABCE, ..., ABDE}; $5C_5 = 1$ {ABCDE}

Ans 10

31 26 16 6 1

if all 1 attribute are C·K

if only 2 combined are C·K

Ques)
$R(A_1 A_2 A_3 \dots A_n)$

How many max. CK's in R?

n Attribute

$\max\{ {}^nC_1, {}^nC_2, \dots, {}^nC_{\lceil n/2 \rceil}, {}^nC_{n-1}, {}^nC_n \}$

$\max$

$^nC_{\lceil n/2 \rceil}$ or $^nC_{\lceil n/2 \rceil}$

Foreign Key: (Referential Key)

- it is used to relate data b/w the table it is used for relationship b/w database table
- foreign key is define over two relation
  1. Referenced Relation (parent table)
  2. Referencing Relation (child table)
- Foreign Key is set of attributes references to Primary Key / Alternate Key (candidate Key) of same relation or some other relation

i) R(A B C)
A: P.K, C: F.K
Referenced Attribute
Referencing Attribute

ii) R(A B) S(C D)
A: P.K, C: F.K, D: P.K
Referenced Relation
Referencing Relation

X (P.K / A.K) Y (F.K)
- F.K may or may not be P.K

Stud(Sid, Sname, DoB) Enroll(Sid, Cid, fee)

Stud Sid	Sname	DoB
S_1	A	-
S_2	B	-
S_3	A	-
S_4	B	-
S_5	A	-
S_6	C	-

Sid: PK

Enroll Sid	Cid	fee
S_1	C_1	5000
S_1	C_2	4000
S_2	C_1	5000
S_2	C_3	3000
S_6	C_1	4000
~~S_{12} C_1 5000~~		restricted

SidCid: PK

∴ F.K should be subset of the referenced key

- name can be different of F.K
- domain of F.K & P.K Should be same

```
CREATE table Stud
 ( Sid varchar(10), Primary Key,
   Sname varchar(20)
   DoB date,
 );
```

```
Create table Enroll
 ( Sid varchar(10),
   Cid varchar(10),
   fee integer(5)
   primary key(Sid,Cid),
   FOREIGN KEY (Sid) References Stud
                              (Sid)
   :
 );
```

restrict the data

Referenical key Integrity Constraints

F.K

Stud(Sid Sname DOB) Enroll(Sid Cid fee)

Stud Sid	Sname	DOB
S_1	-	-
S_2	-	-
S_3	-	-
S_4	-	-
S_5	-	-
S_6	-	-
S_7	-	-

Sid:PK — Referenced Relation (like Parent)

Enroll Sid	Cid	fee
S_1	C_1	
S_1	C_2	
S_2	C_2	
S_2	C_3	
S_6	C_1	

Sid,Cid:PK — Referencing Relation (like child data)

- Insertion
- Deletion
- Updation

1) Referenced relation	2) Referencing Relation
a) **Insertion**: No violation to F.K	a) **Insertion** may cause FK violation
b) **Deletion**: may cause F.K violation	b) **Deletion** No violation
(i) ON DELETE NO ACTION: (default) • not allowed to delete referenced record if F.K violation occurs.	
(ii) ON DELETE CASCADE: • Allowed to delete referenced record and DBMS also deletes all related referencing records.	
(iii) ON DELETE SET NULL: • If F.K field allowed to then allowed to delete referenced records & set NULL value in related referencing F.K field	
c) **Updation**: may cause F.K violation	c) **Updation** may cause F.K violation
i) on update no actions	
ii) on update cascade	
iii) on update set null	

note

- Insertion & updation of Referencing table are restricted if F.K violation occurs

note:- On DELETE set null or on update set null won't be able to set null if is a UNIQUE NOT NULL or its a part of P.K

```
CREATE TABLE Enroll
( sid  Varchar(10),
  cid  varchar(10),
  fee  Integer(5),
  primary key(sid,cid),
  Foriegn key (sid) References
              Stud(sid)
  ON DELETE CASCADE
  ON DELETE CASCADE
);
```

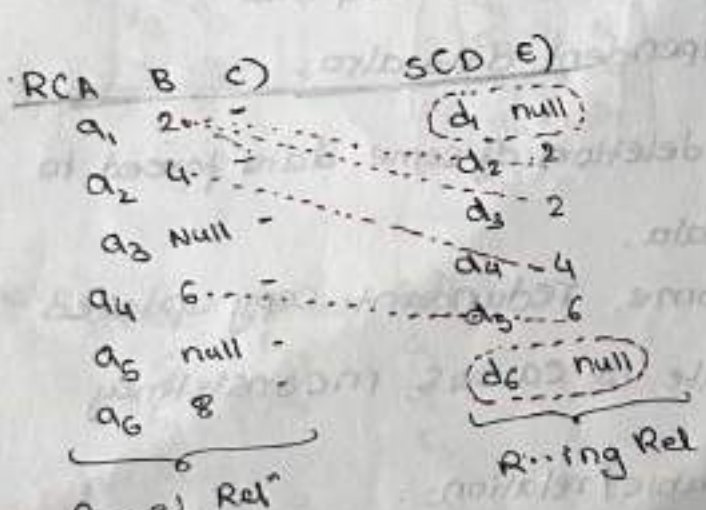
F.K

Emp(eid ename sup(D)) [superviser also must be emp.]

e_1	A	null
e_2	B	e_1
e_3	B	e_2
e_4	D	e_3
e_5	D	null
e_6	E	e_5

1) If F.K IS ON DELETE CASCADE then what record additionly deleted when tuple whose eid=e_1 is deleted? 3 records

2) If F.K is ON DELETE SET NULL then how many additional records deleted when eid=e_1 is deleted? 0 record.



1. F.K fields allowed Null values
2. Mapping b/w R.ed & R.ing record is 1:M
3. each record of R.ed rel is related to many record of R.ing Rel.
4. each record of R.ing rel is related to almost one record of R.ed Rel

many: 0 or more

Some: 1 or more

Normalization [Schema Refinement]

- Used to reduce/eliminate redundancy
- Redundancy in rel. b/c of two or more independent rel^n stored in single relation

F.D { Sid → Sname DOB } { Cid → Cname Instructor } { Sid Cid → fee }

R

Sid	Sname	DOB	Cid	Cname	Inst	fee
S_1	A	2000	C	DB	Korth	4K
S_2	A	2000	C	DB	Korth	5K
S_3	B	2002	C	DB	Korth	4K
S_3	B	2002	C_2	Algo	Cortw	3K
S_3	B	2002	C_3	DB	Nava	4K
			(C_4	OS	Galwin)	

Sid Cid Primary key

DB Anamalies:

[problem's b/c of redundancy]

Causes inconsistency

1) Insertion Anamaly :- To insert some independent data required other independent data also.
2) Deletion Anamaly :- B/c of deletion of some data forced to delete other independent data.
3) updation Anamaly :- If some redundant copy updated other copy failed to update is causes inconsistency

Normalization of the Database table/relation

- Decompose relation into two or more sub-relation to reduce/eliminate redundancy.

R_1(Sid Sname DOB)

S_1
S_2
S_3 B 2002
S_4

R_3(Sid Cid fee)

S_1 C_1 4000
S_2 C_1 5000
S_3 C_1 –
S_3 C_2 –
S_3 C_3 –

R_2(Cid Cname Instr)

C_1 ------
C_2 ------
C_3
C_4 OS Galvin

Normalized DB: [0% redundancy
No DB Anamalies]

Functional Dependency [FD]: $\{x \rightarrow y\}$

X, y some attribute set over R. $x \rightarrow y$ F.D exist in R $\{y \subseteq x\}$

if $t_1.x = t_2.x$ then $t_1.y = t_2.y$

note
- for every value of x there should be only one value of y

R

	X	Y
t_1	x_1	y_5
t_2	x_1	y_5
t_3	x_2	y_3
t_4	x_3	y_3
t_5	x_4	y_2
t_6	x_4	y_5
	x_3	y_3
	x_5	y_9

$x \rightarrow y$ exist in R

R

X	Y
x_4	y_2
x_1	y_2
x_1	y_3

$x \rightarrow y$ not exist in R

R

A	B	C
a_1	4	2
a_2	4	3
a_3	5	4
a_4	6	4

A: Super key
$A \rightarrow B$
$A \rightarrow C$

$x \rightarrow y$ exist in R if x is Super key of R

ex:- R(Sid Sname Cid)

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

Armstrong Rules over FDs:- [x,y,z some attribute set over R]

1. Reflexivity : $X \rightarrow Y$ always exist (Trivial FD) if $Y \subseteq X$

   ex: Sid → Sid
   Sid Sname → Sname

2. Transitivity : if $\{X \rightarrow Y$ and $Y \rightarrow Z\}$ then $\{X \rightarrow Z\}$

   ex: Eid → rating , rating → hwage ⇒ Eid → hwage
   AB → C , C → D ⇒ AB → D

3. Augmentation : if $X \rightarrow Y$ then $XZ \rightarrow YZ$

   A → B ⇒ AC → BC
   Sid → Sname ⇒ SidCid → SnameCid

4. Splite Rule : if $X \rightarrow Y \cdot Z$ then $X \rightarrow Y$, $X \rightarrow Z$

   i.e Sid → Sname DOB ⇒ Sid → Sname, Sid → DOB

5. Union Rule :- if $X \rightarrow Y$, $X \rightarrow Z$ then $X \rightarrow YZ$

   AB → C, AB → D, AB → E ⇒ AB → CDE

6. PSUDO transitive rule

   if $X \rightarrow Y$, $WY \rightarrow Z$ then $WX \rightarrow Z$

Completeness of FD's

derived properties [which can be derived by using 1, 2, 3]

Attribute closure: $[X^+]$

$X^+ = \{$set of all attributes those are determined by x $\}$

i.e FD set = $\{A \rightarrow B, D \rightarrow E, BC \rightarrow D, AB \rightarrow C, DE \rightarrow F, FG \rightarrow H\}$

$(A)^+ = \{ABCDEF\}$
A → ABCDEF

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

$(DG)^+ = \{DGEFH\}$

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

⇒ Superkey :- R(X ....)

X is S.K if Rel^n R iff
x^+ = {All Attribute of R}

R(ABCDE)
{AB → C, A → D, D → E}

SK → $(AB)^+$ = {ABCDE}
SK → $(ABC)^+$ = {ABCDE}
SK → $(ABCDE)^+$ = {ABCDE}
not SK → $(AC)^+$ = {ACDE}

⇒ Candidate Key[minimal S.K]: X is C.K of Rel. R iff

1] X must S.K of R
x^+ = {All Attri. of R}

2] NO proper subset of X is Super key
$\forall y \subset X$ s.t y^+ = {not all Attr. of R}

Ex$_1$: R(ABC) with no non-trivial FD C.K of R: ABC

note: with no non-trivial FD in R then the CK include all attribute combinedly

[no non-trivial FD]

R(A B C)
4 2 3
4 2 2
4 4 3
4 4 2
6 2 3

ABC: CK

Ex$_2$: R(ABCD) {A → B, B → C}
$(AD)^+$ = {ADBC}
AD: CK

Ex$_3$: R(ABCDEF) {C → A, F → B, D → C}
$(DEF)^+$ = {ABCDEF} DEF: CK

Ex$_4$: R(ABCDEF) {A → F, E → A, D → C}
$(BDE)^+$ = {ABCDEF}
BDE: CK

if non-trivial FD, X → Y with Y prime attribute of R then R has atleast two Cand Keys.

{ Non-trivial FD exist in R : X → Y (Y: prime Attri) } ⇔ { Atleast two C.K in R }

⇒ RC...) FD = {.... $y \to x, z \to y$}

PA

Sol. ① And one C.K

WX : C.K

② WX: C.K

$y \to x$ (PA)

SK → $(WY)^+$ = {All Attr. of R}

check minimal SK?

$z \to y$ (PA)

S.K → $(WZ)^+$ = {All Attr. of R}

check for minimal S.K?

Soon

⑥ R(ABCDEF)

{$AB \to C, C \to D, CD \to AE, DE \to F, EF \to B$}

$(AB)^+$ = {ABCDEF}

(A^+) = {A} & $(B)^+$ = {B}

$EF \to \underbrace{B}_{PA}$ ∵ more than one S.K exist

So, $(AEF)^+$ = {ABCDEF}

EF^+ = EFB

$DE \to \underbrace{F}_{PA}$

So, $(ADE)^+$ = {ABCDEF}

DE^+ = DEFB

C.K {AB, AEF, ADE, C}

4 C.K's in R

So, $(CD)^+$ = {ABCDEF}

C^+ = CDAEFB

AEF CD

ADE CD

AB CD

7) R(ABCDEF)

FD: {$AB \to C$, $C \to A$, $BC \to D$, $ACD \to B$, $BE \to C$, $EC \to AF$, $CF \to BD$, $D \to E$}

$(AB)^+$ → {ABCDEF} $A^+ = A$

$(CB)^+$ = {ABCDEF} $B^+ = B$

$(BE)^+$ = {ABCDEF} $C^+ = CA$

$(BD)^+$ = {ABCDEF} $D^+ = DE$

$(CF)^+$ = {ABCDEF} $F^+ = F$

$(CE)^+$ = {ABCDEF} $E^+ = E$

C.K = {AB, CB, BE, BD, CF, BC, EC, CD}

$(ACD)^+$ → $(CD)^+$ = {ABCDEF}

RC

$C \to A$

7. C.K

Membership test :- $X \to Y$ FD member of FD set (F)

iff X^+ should determine Y in FD set F.

i.e $F = \{ \ldots\ldots Y$ ← $X \to Y$

$X^+ = \{ \ldots Y \ldots Y$

Ex. FD set $\{ AB \to CD$
$AF \to D$
$DE \to F$
$C \to G$
$F \to E$
$G \to A \}$

which FD not member of FD set.

a) $BEG \to CF$ $(BEG)^+ = \{ BEGACDF \}$
b) $BG \to D$ $(BG)^+ = \{ BGACD \}$
c) $BCF \to A$ $(BCF)^+ = \{ BCFGAED \}$
~~d)~~ $AB \to E$

Equality of FD sets :- F & G FD set are logically equal

iff 1] F covers G : Every FD of G must be member of F i.e $F \supseteq G$

2] G covers F : Every FD of F must be member of G. i.e $F \subseteq G$

F covers G	G covers F
① YES	Yes ⇒ F = G
② YES	NO ⇒ F ⊃ G i.e F is superkey
③ ~~YES~~ NO	Yes ⇒ F ⊂ G
④ ~~YES~~ NO	NO ⇒ F & G not comparable

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

$A^+ = BCA$
$B^+ = ABC$
$C^+ = ABC$

$G : \{ A \to BC$,
$B \to AC$,
$AC \to B$,
$BC \to A \}$

$A^+ = ABC$
$B^+ = ABC$

$X : \{ \ldots A \to B \}$
$Y : \{ \ldots D \to D \}$

F covers G ? = True

F	G
$A \to B$	✓ $A \to BC$
$B \to C$	✓ $B \to AC$
$C \to A \}$	✓ $AC \to B$
	✓ $BC \to A$

G covers F ? → false

$G = \{ A \to BC$
$B \to AC$
$AC \to B$
$BC \to A \}$

F
$A \to B$ ✓
$B \to C$ ✓
$C \to A$ ✗

$C \to A$ not member of G

which is true

a) $F \subset G$
b) $F \supset G$
c) $F = G$
d) none

Qus] $F = \{A \to BCDEF, BC \to ADEF, B \to F, D \to E, AB \to EF\}$

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

F covers G ? TRUE

G
✓ $A \to BCD$
✓ $BC \to A$
✓ $D \to E$
✓ $B \to F$

$F = \{A \to BCDEF, BC \to ADEF, B \to F, D \to E, AB \to EF\}$

$A^+ = ABCDEF$

$BC^+ = BCADEF$

$D^+ = DE$

$B^+ = BF$

G covers F ? TRUE

F
✓ $A \to BCDEF$
✓ $BC \to ADEF$
✓ $B \to F$
✓ $D \to E$
✓ $AB \to EF$

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

$A^+ = ABCDEF$

$BC^+ = BCADEF$

So F = G

Irreducible FD set (or) Canonical Cover (or) Minimal Cover :- ∵ to make Normⁿ easy

- Minimal cover (F_m) of Given FD set (F) is minimal possible FD's which are logically equal FD set F.
- minimal cover F_m of Given FD set (F) is non-reducable equal FD set.

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

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

$AB \to B$

$AB \to C$

$B \to C \Rightarrow AB \to AC$

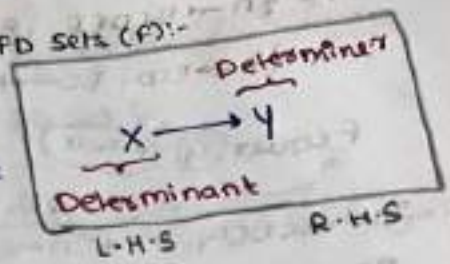
$AB \to A$

$AB \to C$

Procedure to find minimal covers of given FD sets (F):-

Step

1. Eliminate Extraneous Attr. from each determinants [LHS of FD] of FD's set F



Extraneous Attr:-

(i) $\{xy \to z, x \to y\}^+ \equiv \{x \to z, x \to y\}^+$

y extraneous if $x^+ \to$ derived z then (y) is extra

Ex:- $\{AB \to D, B \to D\}^+$

$\{ABC \to D, B \to A\}^+$

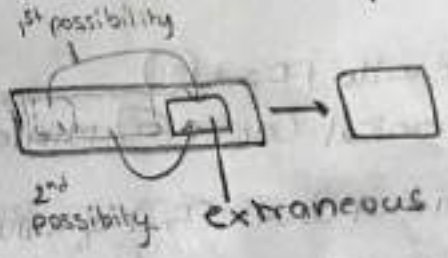
$\{ABC \to D, C \to F, F \to AB\}^+$

(ii) $\{wxy \to z, w \to x\}^+ \equiv \{wy \to z, w \to x\}^+$

x extraneous

(iii) $\{xy \to z, x \to z\}^+ \equiv \{x \to z\}^+$

y extraneous



2. Elliminate Redundant FD's from result of step (1)

Redundant FD:- $X \to Y$ is redundant in FD set F

iff $X \to Y$ must be member of $\{F - (X \to Y)\}^+$

Should be single attribute

If not then split

FD sets

By removing those FD still we are able to get y in the x^+ (x closure)

Ex:- Find minimal cover of FD set ?

1] $F = \{AB \to C, C \to A, BC \to D, ACD \to B, BE \to C, EC \to FA, D \to E\}^+$

$\{AB \to C, C \to A, BC \to D, CD \to B, BE \to C, EC \to F, D \to E\}^+$

$A^+ = A \; B^+ = B \; BC^+ = BCA$

$B^+ = B \; C^+ = CA$

$C^+ = CA$

extraneous Attr. of determiniant

remove redundant FDs

$\{AB \to C \to AB^+ = AB$

$C \to A$ ✓

$BC \to D$ ✓

$CD \to B$ ✓

$BE \to C \to BE^+ = BE$

$EC \to F$ ✓

$D \to E$ ✓

$\}^+$

F_m

HW (W-8)
1-10
47-51
T_4, T_5

5) R(TUVWXYZ)

FD: {TU → VW,
TY → W,
WX → Y,
V → Z,
Y → X,
Z → T}

no. of C.K?

(TU)$^+$ = {TU}
(TY)$^+$ = {TYXW}
(ZU)$^+$ = {ZUTWV}
(YU)$^+$ = {YUX}
(VU)$^+$ = {VUZTW}
(WU)$^+$ = {WU}
(XU)$^+$ = {XU}

TUYWXYZ

(VUX)$^+$ = {VUZTWXY}
(VUY)$^+$ = {" "}
(UUWX)$^+$ = {" "} ⊆ Rwt
(UWX)$^+$ = {UWXY} ×
(TUX)$^+$ = {TUXVWYZ}
(ZUX)$^+$ = (ZUXTVW Y)
(ZUY)$^+$ = {" "}

C.K = {VUX, VUY, TUX, TUY, ZUX, ZUY}
P.A = {V, U, X, Y, T, Z}

8) F = {A → BC,
CD → E,
E → C,
D → AEH,
ABH → BD,
DH → BC}

what is canonical cover of F?

F = {A → BC
~~CD → E~~
E → C
D → AEH
ABH → BD
DH → BC}

remove Extraneous
AH

F = {A → BC ✓
E → C ✓
D → AEH ✓
ABH → D ✓
DH → BC ×
}

$A^+ = A$

$DH^+ = DHABC$

10] minimal cover of F

F = {AB → C,
C → A,
BC → D,
ACD → B,
BC → C,
EC → FA,
CF → BD,
D → E,}

F = {AB → C
C → A
BC → D
~~ACD~~ → B
BE → C
EC → F
CF → BD
D → E}

F = {AB → C ✓
C → A ✓
BC → D ✓
ACD → B ✓
BE → C ✓
EC → F ✓
CF → BD ✓
D → E ✓
}

$AB^+ = AB$

$BC^+ = BCA$

$BE^+ = BE$

$CF^+ = CFA$

47) i) {A → BCDEF
BC → ADEF
B → F
D → G}

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

F = {A → BCDEF ✓
BC → ADE ✓
B → F ✓
D → E ✓
}

ii) P:{ AB → C
D → E
AB → E
E → C
}

F:{
~~AB → C~~
D → E
AB → EC
E → C
}

iii) F:{ AB → C
BC → D
A → BC
B → AC
C → AB
}

F:{
~~AB → C~~
~~BC → A~~
A → BC
B → AC
C → AB
}

P:{
A → BC
B → AC
C → AB
}

(48) How many C.K R(ABCDEH)

F:{ A → BC ✓
CD → E
E → C
D → AEH
ABH → BD ✓
DH → BC ✓
}

(A)⁺ = ABC
(DH)⁺ = (DH BC)
(ADH)⁺ = {ABCDEH}
(ABH)⁺ = {ABHDCE}
(AD)⁺ = {ABCDEH}

C.K: {ADB, ABH, DH}
P.A: {A, D, H, B}

(49) R(ABCDE)

FD: {A → C,
BC → E,
ED → A
}

(ABD)⁺ = {ABCDE}
(EBD)⁺ = {ABCDE}
(BCED)⁺ — ×
(BCD)⁺ = {ABCDE}

C.K = {ABD, EBD, CBD}

(3) CK

(50) R(ABCDEFG)

F{AB → CD
AF → D
DE → F
C → G
F → E
G → A
}

(AB)⁺ = {ABCDG}
(CB)⁺ = {CBGADE}
(DB)⁺ = {DB}
(FB)⁺ = {FBE}
(GB)⁺ = {ABCDG}

(DBE)⁺ = {DBEF}
(FBE)⁺ = {FBE}

C.K = {ABF, CBF, GBF
ABE, CBE, GBE}

(51) R(ABCDEH)

F:{ A → BC
CD → E
E → C
D → AEH
ABH → BD
DH → BC
}

Common mistake → in finding minimal cover

FD set $F = \{XA \to y, \ldots\}$

In FD: $XA \to y$ — single attribute

to check this A is Extraneous or not

mistake: 1st remove A then in the Resulting F.D set you find x^+

Correct method is

find x^+ in F without removing & if x^+ contain y then A is Extraneous.

note: minimal cover is not minimum

minimal cover ≡ Irreducible set of F.Ds

- we can have multiple minimal Cover & different minimal cover can have different no. of FD. } minimal cover may not be unique.

gate 2024

ex: FD ⇒ F: $X \to y$ is useful if

- X is not empty
- y is not empty
- & $X \cap y = \phi$

for Relation R with 4 attribute

total no. of possible useful F.D is → (50)

method 1 create the case's

R(ABCD)

Case 1: |LHS|=1

A → any non-empty subset of BCD → $2^3 - 1 = 7$ choices

B → ~~~

C → ~~~

D → ~~~

$^4C_1$

So, 7×4 = (28) in total

Case 2: |LHS|=2

AB → any non empty subset of CD → $2^2 - 1 = 3$ choices.

AC → ~~~

AD → ~~~

BC → ~~~

BD → ~~~

CD → ~~~

$^4C_2$

3×6 = (18) in Total

Case 3: |LHS=3|

ABC → 1 choice

ACD → ~~~

BCD → ~~~

ABD → ~~~

$^4C_3$

Total (4) choices.

method 2 → n=4

$^nC_1 * (2^{n-1} - 1) + {}^nC_2 * (2^{n-2} - 1) + {}^nC_3 * (2^{n-3} - 1)$ ⇒ Total ans.

Partial & Full F.D

- Fully dependent

child person A is "fully Dependent" on person (BC) : then

$BC \rightarrow A$

& $B \not\rightarrow A$ & $C \not\rightarrow A$ } A is fully Dependent on B.C

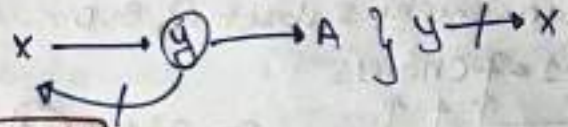
- partial dependent

$(\alpha \rightarrow B)$ is partial F.D iff

$\exists_{A \in \alpha} \ (\alpha - A) \rightarrow B$

B is partially dependent on α iff Some proper subset of α itself can Determine B

- Transitive dependent



$y \not\rightarrow x$

$X \rightarrow A$ is transitive FD iff $\exists y$

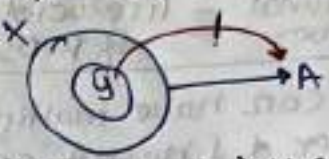
(or) another way to say

$(\alpha \rightarrow B)$ is Full F.D iff

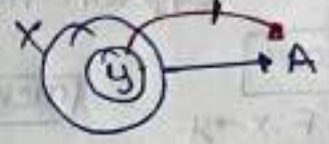
$\forall_{A \in \alpha} \ (\alpha - A) \not\rightarrow B$ removing anything

(or) B is Fully Dependent on α iff any proper subset of α can't determine B

Full: $X \rightarrow A$ & $y \subset X$



partial: $X \rightarrow A$ & $y \subset X$



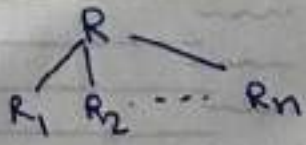
- Problem caused by Redundancy
  - ↳ Redundant Storage
  - ↳ Update anomalies
  - ↳ Insertion anomalies
  - ↳ Deletion anomalies
- Lossy (or) lossless is checked for schema not for instance

- Decomposition

$R = R_1 \cup R_2 \cup \dots \cup R_n$

(or)

$\bigcup_{i=1}^{n} R_i = R$ (attribute preservation)



To call it a Decomposition

$\bigcup_{\forall i} \text{attribute}(R_i) = \text{attribute}(R)$

T4) FD-?
- AB → C
- C → A
- BC → D
- ACD → B
- BE → C
- EC → AF
- CF → BD
- D → E

FD-?
AB → C
C → A
BC → D
~~ACD → B~~
BE → C
EC → AF
~~CF → BD~~
D → E

~~FD-?~~
~~A → CB~~
C → A
B → D
~~AC → B~~
~~EC → F~~
~~E → BDA~~
~~D → E~~

~~FD-?~~
~~A → BE~~
~~B → D~~
~~EC → F~~
~~E → BDA~~
~~D → E~~

∴ there 3 rule are left for those, whose record is give & FD not

Properties of decomposition :-

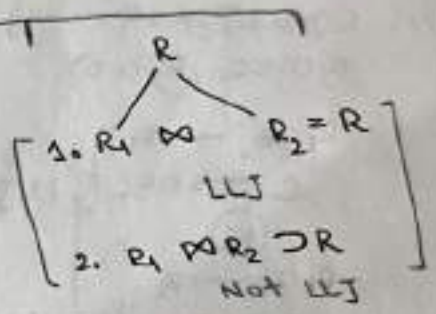
① Lossless Join Decomposition [LLJ]

Rel R with Instance (r) decomposed into sub relation $R_1, R_2, R_3 \cdots R_n$

(i) In general $[R_1 \bowtie R_2 \bowtie R_3 \cdots \bowtie R_n] \supseteq r$

(ii) If $[R_1 \bowtie R_2 \bowtie \cdots R_n] = r$
then Lossless join decomp. [LLJ]

(iii) If $[R_1 \bowtie R_2 \bowtie \cdots \bowtie R_n] \supset r$
then Lossy Join decomp [Not LLJ]
(extra spurious tuple in result of join b/w subrelation : cause inconsistency)



3. $R_1 \bowtie R_2 \subset R$ not possible

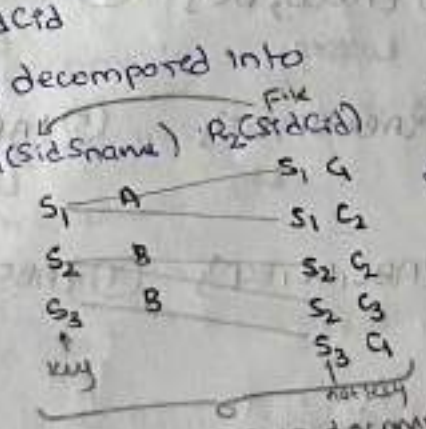
Q] R

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

Given Relation

$Sid \rightarrow Sname$
C.K : SidCid

i] decomposed into



Loss Less join decomp

$(R_1 \bowtie R_2) = R$

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

② ⇒ Rel^n R with FD set(F) decomposed into Sub Rel^n R_1, R_2 is lossless join decomp. iff (i) $R_1 \cup R_2 = R$

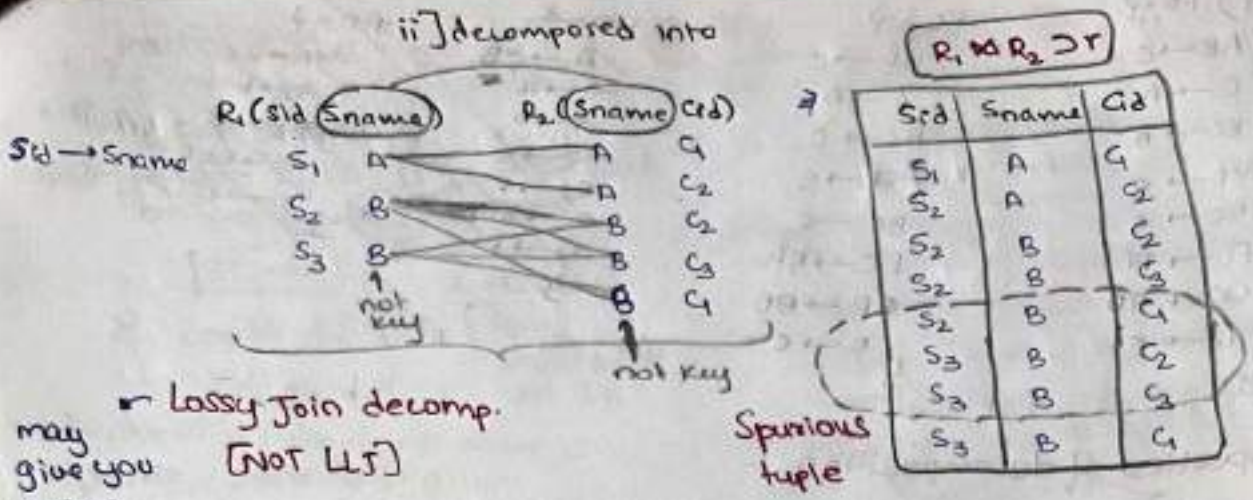
(ii) $R_1 \cap R_2 \rightarrow R_1$ [$R_1 \cap R_2$ is S.K of R_1]

(or) $R_1 \cap R_2 \rightarrow R_2$ [$R_1 \cap R_2$ is S.K of R_2]

Should be S.K

lossless join will never give wrong information

ii] decomposed into



Sid	Sname	Cid
S_1	A	C
S_2	A	C_2
S_2	B	C_2
S_2	B	C_3
S_2	B	C
S_3	B	C_2
S_3	B	C_3
S_3	B	C

- Lossy Join decomp. may give you wrong information (for some instance) [NOT LLJ]

SQL: Cid's

Ex: R(ABCD) F={- - - -}
R_1(ABC) R_2(CD)

$R_1 \cap R_2 \rightarrow R_1$
$C \rightarrow ABC$ } LLJ
(or)
$R_1 \cap R_2 \rightarrow R_2$
$C \rightarrow CD$

- lossy decomposition
  - ↳ not losing any original tuple (not loss of tuple)
  - ↳ getting spurious (wrong) information tuple

③ $R_1 \cap R_2 \not\rightarrow R_1$ and $R_1 \cap R_2 \not\rightarrow R_2$

Note
not required to be key of both relation but it should be for one then its LLJ else if it is not key for both then its lossy

Q] R(ABCDEF) Test given decomp. LLJ or not

{AB → C, C → D, D → E, B → F}

① {ABCD, DE} Lossy

② {ABCD, DEF} Lossy

③ {ABC, DEF} Lossy

④ {ABCDF, DE} LLJ

⑤ {ABCD, ABEF} LLJ

⑥ {ABC, DE, BCF} Lossy

⑦ {ABC, CD, BDF} Lossy

⑧ {ABC, CD, BF, DE} LLJ

① {ABCD, DE}
$R_1 \cup R_2 \neq R$
Attribute F not in subrel.
[Lossy]

② {ABCD, DEF}
$R_1 \cup R_2 = R$
$R_1 \cap R_2 = D$
$D^+ = DE$
not S.K for any sub-relation

③ {ABC, DEF}
$R_1 \cup R_2 = R$
$R_1 \cap R_2 = \phi$
not LLJ

(iv) {ABCDF, DE} LLJ

$R_1 \cup R_2 = R$

$R_1 \cap R_2 = (D)^+ = \{DE\}$

(v) {ABCD, ABEF} LLJ

$R_1 \cup R_2 = R$

$R_1 \cap R_2 = (AB)^+ = \{ABCDEF\}$

(vi) {ABC, DE, BCF}

R_1 R_2 R_3

$\Rightarrow R_1 \cup R_2 \cup R_3 = R$

$\rightarrow R_1(ABC)$ $R_2(DE)$ $R_3(BCF)$

$BC^+ = BCDEF$

$R_{13}(ABCF)$ $R_2(DE)$

Not LLJ

(vii) {ABC, CDE, BDF}: LLJ

R_1 R_2 R_3

$C^+ = CDE$

$R_{12}(ABCDE)$ $R_3(BDF)$

$BD^+ = BDEF$

$R_{123}(ABCDEF)$

(viii) {ABC, CD, BF, DE}

R_1 R_2 R_3 R_4

$C^+ = CDE$

$B^+ = BF$ $R_{12}(ABCD)$ $R_3(BF)$ $R_4(DE)$

$R_{123}(ABCDF)$ $R_4(DE)$

$D^+ = DE$

R_{1234}

(2) Dependency preserving decomp:-

Rel R with FD set (F) decomposed into sub rel $R_1, R_2, R_3 \dots R_n$ with FD sets $F_1, F_2, F_3 \dots F_n$ respectively

(1) In general

$[F_1 \cup F_2 \cup F_3 \cup \dots F_n] \subseteq F$: F cover subrelation

(2) If $[F_1 \cup F_2 \cup F_3 \dots \cup F_n] = F$

The DP decomp.

(3) If $[F_1 \cup F_2 \cup F_3 \dots \cup F_n] \subset F$

the not DP decomp.

R(ABCD) {AB → CD, D → A}

Find non trivials of R(BCD):

FD = {BD → C}

B⁺ = B
C⁺ = C
D⁺ = DA
BC⁺ = BC
CD⁺ = CDA
BD⁺ = BDAC

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

for R₁(ACD)

FD: {AC → D, C → D, AD → C}

A⁺ = A
B⁺ = B
C⁺ = DCEB
D⁺ = DEB
AC⁺ = ACDEB
AD⁺ = ADCEB

Ques] R(ABCDE) · decomp {AB, BC, CD, DE}

FD: {A → B
B → C
C → D
D → BC}

Those closures not present above

R₁(AB)
FD: {A → B}

R₂(BC)
FD: {B → C, C → B}

R₃(CD)
FD: {C → D, D → C}

R₄(DE)
FD: {D → E}

D⁺ = DEBC

{F₁ ∪ F₂ ∪ F₃ ∪ F₄}⁺ = F

DP decomp ✓

Ques] R(ABCD) FDs: {AB → CD, D → A}

decomp · {ABC, BCD, AD}

R₁(ABC) FD: {AB → C}
R₂(BCD) FD: {BD → C}
R₃(AD) FD: {D → A}

AB → C ✓
D → A ✓
AB → D ✗

AB⁺ = ABC

{F₁ ∪ F₂ ∪ F₃}⁺ ⊂ F

AB → D FD lost hence not DP decomp.

Checking all these FD by checking the given FD

not DP decomp.

LJ test : R₁(ABC) R₂(BCD) R₃(AD)

R₁(ABC) ⋈ R₂₃(ABCD)

R₁₂₃(ABCD)

Do 2^n steps the produced FD will be used to check whether you can create the given FD by taking closure

ex. R(ABCDE) → universal relation (Database)

$F = \{A \to BC, CD \to E, B \to D, E \to A\}$ → FD's

Check for lossless/lossy?

$R_1 \cap R_2 = A \to R_1$; so lossless

$A^+ = ABCDE$

$R_1 = \{A, B, C\}$ $R_2 = \{A, D, E\}$

F_1 = (FD's Holding on R_1)

$R_1(A, B, C)$

$A^+ = ABCDE$

$B^+ = BD$

$C^+ = C$

$BC^+ = BCDEA$

R_1 can have FD

$A \to BC$

$BC \to A$

→ R_1 can preserve only these FD

F_1: FD preserved by R_1

F_1: FD Holding on R_1

$R_2(ADE)$

F_2 = (FD Holding on R_2)

$A^+ = ABCDE$

$D^+ = D$

$E^+ = ABCDE$

$R_2(ADE)$

$A \to DE$

$E \to A$

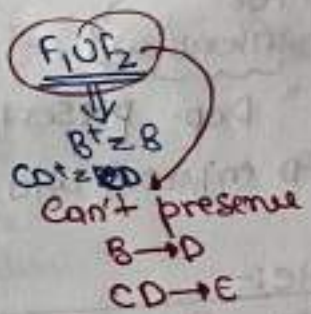
$E \to D$

$\equiv$ $A \to DE$, $E \to A$ → R_2 is preserving these FD

F_1: $\{A \to BC, BC \to A\}$ preserved by R_1

F_2: $\{A \to DE, E \to AD\}$ preserved by R_2

Combinedly $(F_1 \cup F_2)$ cover F??



$R_1 + R_2 \Rightarrow (F_1 \cup F_2)$ doesn't cover all F

Decomp. lossless

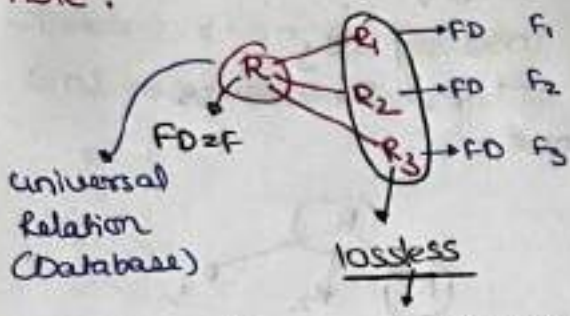
So, we can combine

$R_1 \bowtie R_2 = R$ → R: Already has $B \to D$ FD

So, $B \to D$ FD w/o joining tables can't be enforced

So, we can say "non-dependency preserving"

note:



Then we never lose any dependency
(OR) we can somehow enforce FD

if $(F_1 \cup F_2 \cup F_3)$ cover F
then there is "No need" to join table to enforce FD's
↓
Dependency preservation

if $(F_1 \cup F_2 \cup F_3)$ doesn't cover F
then there is need to join table to enforce FD only when it's lossless)
↓
non Dependency preservation

ex. In any Decomposition what is most Important
lossless

ex: In any Decomposition what is Desirable?
lossless & Dep. preserving
benefit: no need to join table for FD's

For Interview
"Dep. preserving"
FD enforced w/o joining table

"non-Dep. preserving"
To enforce FD's need to join tables.

note:-

- $F^+ \supseteq (\cup F_i)^+$ ---→ Always
- $F^+ \subset (\cup F_i)^+$ ---→ Never happen
- $F^+ = (\cup F_i)^+$ ---→ Dep. preserving
- $F^+ \supset (\cup F_i)^+$ ---→ non-Dep. preserving

note
lossless join
↳ (non-Additive join Decomp.)

ex: $R = (A, B, C)$
$F = \{A \to B, B \to C, A \to C\}$

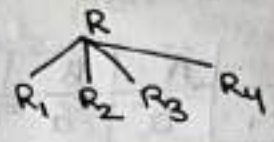
$R_1(AB)$ $F_1 \{A \to B\}$
$R_2(BC)$ $F_2 \{B \to C\}$

in $(F_1 \cup F_2)$ cover F

In these $A^+ = ABC$
So, $A \to C$ is covered

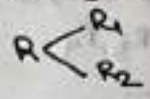
So, Dep-preserving

non-Binary Decomposition



$K \geq 3$

Binary Decomposition



lossless:

$R_1 \cap R_2 \rightarrow R_1$ (or)

$R_1 \cap R_2 \rightarrow R_2$

else lossy Decomposition

testing will work only for binary Decomposition

- Testing if any decomposition (binary or non-binary) is lossy or lossless

Chase Algorithm.

- Successive Combining method to check lossless/lossy decomposition
- this is one way method "lossless" then Decomposition will be "lossless"

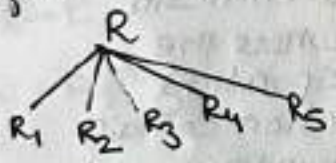
adding 2 at a time

- if this method ans "lossy" then Decomposition may or may not be lossless

ex: where Successive Combining will fail

$R = \{A, B, C, D, E\}$ →

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



$R_1(A,D)$
$R_2(A,B)$
$R_3(B,E)$
$R_4(C,D,E)$
$R_5(A,E)$

R_1 R_4 ~~$R_1 R_4$~~ no common attribute that is a S.K in one of the Relation

R_2 R_4 ~~$R_2 R_4$~~

~~$R_3 R_4$~~ ~~$R_1 R_5$~~

$D^+ = D$
$A^+ = ACD$
$B^+ = BCD$
$C^+ = CD$
$(DE)^+ = DECA$
$(CE)^+ = CEAD$
$E^+ = E$

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

$R_{12}(ABD)$

R_{12} R_3 ~~$R_{12} R_3$~~

R_{12} R_5 ~~$R_{12} R_5$~~

R_2 R_4 ~~$R_2 R_4$~~

So, no such possibility exist

So, Successive Combining methods is giving And as lossy But this Decomposition is actually lossless;
we will see it using chase Algo.

Chase Algo.

ex: R(abcde)

R_1, R_2, R_3, R_4

$R_1(ab)$ $R_2(ac)$

$R_3(bcd)$ $R_4(ade)$

Step 1: Draw Matrix

Step 2: place a_i in place of all attribute in that Relation

Step 3: for a given F.D if L.H.S of that F.D have same value to R.H.S then it should follow for all enter's.

using Matrix

attribute

Initial table	A	B	C	D	E
R_1	a	a	a		b
R_2	a		a		
R_3		a	a	a	b
R_4	a		a	a	a

Sub relation

Should be same for BC→E

if FD's $A \rightarrow C$, $BC \rightarrow E$

for every value of a there is only one c

(So in table if a come's in A so, a Should also come is c)

Step involve.

① Initial setup
Column: Attribute
Rows: Sub relation

② put a-entries in Cells where the corresponding Subrelation contain Corresponding attribute

③ Now repeatedly fill the table using FD
idea:- if $X \rightarrow y$ is F.D & in two tuples X value is same then y will should also be same in there 2 tuple

④ Repeatedly apply Step 3 until no more change happen in the table

⑤ if there exist a Row with all a-entries then lossless
else lossy

ex: R(ABCDE) FD's $A \rightarrow C, B \rightarrow C, C \rightarrow D, DE \rightarrow C, CE \rightarrow A$

finally we want all a rather than b

$R_1(A,D)$
$R_2(AB)$
$R_3(BE)$
$R_4(CDE)$
$R_5(AE)$

	A	B	C	D	E
R_1	a		a	a	
R_2	a	a	a	a	
R_3	a	a	a	a	a
R_4	a		a	a	a
R_5	a		a	a	a

all 'a' So it's lossless

ex: R(A B C D E)
$R_1(ABC)$ $R_2(BCD)$ $R_3(ACE)$ } use chase test Algo. & also find Dependency preserved.

(a) $B \to E$ & $CE \to A$

$A^+ = A$ / $B^+ = BE$ / $C^+ = C$

$(CE)^+ = CEA$

	A	B	C	D	E
R_1	a	a	a		b
R_2	a	a	a	a	b
R_3	a		a		a

So, lossy

$R_1(ABC)$ no F.D
$R_2(BCD)$ no F.D
$R_3(ACE)$ $CE \to A$

So, $B \to E$ is not covered by $F_1 \cup F_2 \cup F_3$. So non-Dep. Preserving Decomposition.

(b) $AC \to E$, $BC \to D$

$R_1(ABC)$ $R_2(BCD)$ $BC \to D$ $R_3(ACE)$ $AC \to E$

So, Dependency preserving

	A	B	C	D	E
R_1	a	a	a	a	a
R_2		a	a	a	
R_3	a		a		a

So, it's lossless

(c) $A \to D$, $D \to E$, $B \to D$

$R(ABC)$ $R_2(BCD)$ $B \to D$ $R_3(ACE)$ $A \to E$

So, non-Dependency preserving

	A	B	C	D	E
R_1	a	a	a	a	a
R_2		a	a	a	c
R_3	a		a	b	a

So, it's lossless

$A^+ = ADE$
$B^+ = BDE$

gate 2002

(ex) $R(L,M,N,O,P) \to R_1(L,M,N,P)$ & $R_2(M,O)$ } is a lossless.

$M \to O$

FD: { $M \to O$,
$NO \to P$,
$P \to L$,
$L \to MN$ }

$P \to LMN$
$L \to MNP$
$NM \to P$

So, it's not Dep. preserving b/c $NO \to P$

gate 2021

ex: R(PQSTxyzw)

$PQ \to x,\ p \to yx,\ Q \to y,\ y \to zw\ \}^{+} \to [PQ \to x, P \to y, P \to x, Q \to y, y \to z, y \to w]$

(D1) $R_1(PQST)\ R_2(PTx)\ R_3(Qy)\ R_4(yzw)$

	P	Q	S	T	x	y	z	w
R_1	a	a	a	a	a	a	a	a
R_2	a			a	a	a	a	a
R_3		a				a	a	a
R_4						a	a	a

} so it's lossless

(D2) $R_1(PQS)\ R_2(Tx)\ R_3(Qy)\ R_4(yzw)$

	P	Q	S	T	x	y	z	w
R_1	a	a	a			a	a	a
R_2				a	a			
R_3		a				a	a	a
R_4						a	a	a

} so, no change, so it's lossy

gate 2019.

X(PQRS)] not in BCNF

FD{ QR → S, R → P, S → Q }] C.K

	P	Q	R	S
(1)	a		a	
(2)	a	a	a	a

} it's lossless

X(PQRS)

Y(PR), R → P

Z(QRS), QR → S, S → Q — C.K for these is QR, SR

it's dep. preserving

is violation (S → Q, not S.K)

is in BCNF

So, not in BCNF

Q] R(ABCDEF)

FD's { AB → C ✓
C → D ✓
D → E ✓
A → F ✓ }

Decomp. {ABCD, DE, AF}

R1(ABCD)	R2(DE)	R3(AF)
AB → C C → D	D → E	A → F

So, DP is true

Normal Forms - used to identify degree of redundancy & used to reduce or eliminate redundancy

0% redundancy over FD's

1NF → 2NF → 3NF → BCNF → 4NF ← 0% redundancy over FD's & MVD's

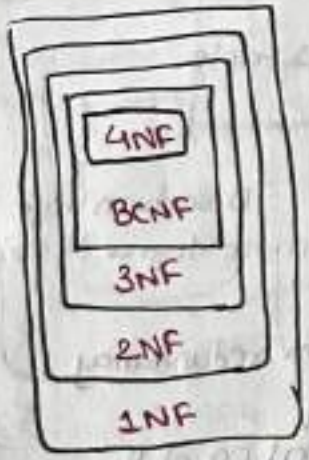
over FD's [x → y]
single valued dependencies

over MVD's [x →→ y]
(Multivalued dependency)

← not required for gate

- to eliminate redundancy over FD's decompose relation into to BCNF
- to eliminate redundancy over FD's decompose relation into 4NF (4NF: 0% redundancy over FD & MVD's)

i.e Relations



1 First Normal Form (1NF):

- relational schema is in 1NF if no multivalued in the relation then the relation is in 1NF
- every attribute of the relation should be single value / atomic
- Domain of every attribute is Atomic
- Default NF of RDBMS table

ex: R

eid	ename	DOB	Pid
e1	A	2000	P1/P2
e2	B	2000	P2/P3
e3	B	2002	P3

1NF Design

R not in 1NF
∴ not allowed in RDBMS

{Find FD's & Design candkey}

R

eid	ename	DOB	Pid
e1	A	2000	P1
e1	A	2000	P2
e2	B	2000	P2
e2	B	2000	P3
e3	B	2002	P3

{eid → ename, DOB}

not S.K

C.K : {eid, Pid}

∴ no MVA
∴ 1NF

Ex₂

R(eid	ename	DOB	Pid	email	pho.)
001	A	2000	P_1/P_2	$e_1/e_2/e_3$	$x_1/x_2/x_3/x_4$
002	B	2000	P_2/P_3	e_3/e_4	x_4/x_5
003	B	2002	P_4	e_4	x_5

1NF Design

{eid → ename DOB}

∴ eid → Pid b/c MVA

C.K : {eid, Pid email phno.}

not S.K

R	eid	ename	DOB	Pid	email	pho.
	001	A	2000	P_1	e_1	x_1
	001	A	2000	P_1	e_1	x_2
	⋮					⋮
						x_4
	001	A	2000	P_2	e_3	
	002	B	2000	P_2	e_3	x_4
	002	B	2000	P_2	e_3	x_5
	⋮					⋮
						x_5
	002	B	2000	P_3	e_4	x_5
	003	B	2002	P_4	e_4	

P_1 P_2 ② × e_1 e_2 e_3 ③ × x_1 x_2 x_3 x_4 ④ = 24 tuples

P_2 e_3 x_4 / P_3 e_4 x_5 : 2 × 2 × 2 = 8 tuples

1 tuple.

- 1NF
- RDBMS

Goal

Just to convert data into RDBMS (it doesn't remove redundancy due to FD) ⟵ start removing from 2NF

Disadvantage

- more degree of redundancy

$X \to Y$ FD form redundancy in Rel. R iff

① non trivial FD of R

& ② X is not SK of R

$X \to Y$ FD of R not forms any redundancy in R iff

① Trivial FD: $Y \subseteq X$

(or)

② X is SK of Rel R

Non trivial FD

$X \to Y$

not S.K

Ex3 R(eid, ename, DOB) {eid → enameDOB}
eid: SK

eid	ename	DOB
e1	A	2002
e2	A	2002
e3	B	2003
e4	B	2003
e5	B	2000

[not forms redundancy]
(no redundancy)

Ex4 R

eid	ename	DOB	rating	hourlywage
e1	A	2000	5	6000
e2	A	2000	5	6000
e3	B	2003	5	6000
e4	B	2000	6	7000
e5	B	2000	6	7000
e6	C	1998	6	7000
e7	C	2000	7	7000

is redundancy b/c hourlywage depends on rating

is not redundancy b/c DOB doesn't depend on ename

{eid → enameDOB rating, rating → hourlywage}
rating: not SK
⇓ forms redundancy

How many FD's can form redundancy?
4 FD are redundancy.

Ex5 R(ABCDEF)
(AB)+ = ABCDEF
SK

FD: {AB → C,
1. B → D,
2. D → E,
3. AE → F,
4. C → DA}

not SK; form redundancy

[C.K = AB, BC]

{∴ X → Y not trivial
X is not S.K
forms redundancy

Y

1. proper subset of CK determines Nonprime Attribute.
2. NPA determine another NPA
3. proper subset of CK combined with NPA determines NPA
4. proper subset of CK determine another proper subset of C.K

Second normal form [2NF]

Rel R is in 2NF iff
no partial depency in R

✗ C·K → non-prime attribute

Partial dependency [PD]

X: any c·k of R
Y ⊂ X
Z: non prime Attr. of R
Y → Z : partial dependency

third normal form [3NF]

Rel R is in 3NF iff
every non-trivial X → Y in R
with (1) X must be s·k of R
OR
(2) Y must be prime Attr. of R

non trivial FD's X → Y (X: SK OR, Y: PA)

- 3NF reduce redundancy Compare to 2NF but not eliminate redundancy over FDs
- Proper Subset of C·K → Proper Subset of other C·K is allowed in 3NF which form redundancy.

Non Trivial FD's which Forms redundancy

Forms redundancy : {X → Y non-trivial FD with X not Super Key}	1NF	2NF	3NF	BCNF	4NF
(1) X: proper subset of candidate key (Super Key) → Y: non-prime	allowed ✓	not allowed ✗	not allowed	✗	✗
(2) X: non prime attribute → Y: other non-prime attribute	✓	✓	✗	✗	✗
(3) X: proper Subset of C·K + non-prime Attribute → Y: Non-prime attribute	✓	✓	✗	✗	✗
(4) X: proper sub. set of C·K → Y: proper subset of other C·K	✓	✓	✓	✗	✗

1NF → no multivalued
↳ Domain of every attribute is Atomic

2NF → Every non-prime attribute should be full dependent on every C·K
X [Partial C·K ⟶ non-prime]
Should not determine

not in 2NF → Some non-prime attribute partially dependent on Some C·K

ex: R is in 2NF iff all non-prime attribute is fully dependent on all Super Key — False
it should be C·K

ex R(ABCD) & F·D = {AB → CD, C → D} — not in 2NF

non-prime: C, D
C·K: AB

ABC → D — Partial F·D
ABC: Super Key
D: non-prime

ex: if no composite C·K then R must be in 2NF (yes)
every C·K has single attribute

3NF → if no non-prime attribute is transitive Dependent on any C·K

(or) Every nonprime attribute is non-transitively Dependent on Every C·K.

Some C·K —Transitively→ Some non-prime attribute
Violation

ex: not a Super Key → non-prime attribute
Violation of 3NF

ex α → A then
CK → α → A

ex: 3NF can have transitive dependency if both side are proper subset.

note
X → Y
must be S·K (or) prime attribute

BCNF: X → Y
must be S·K

Note

- A Relation in 2NF may still have partial dependency but a relation in 2NF can never have a dependency where Some C·K is partially determining Some non-prime attribute
- A relation in 3NF may still have Transitive dependency but a relation in 3NF can never have Transitive have dependencies where Some C·K is transitively determining Some non-prime attribute

normal forms.

1NF
2NF
3NF
BCNF

Redundancy Due to FD Decrease

0% Redundancy Due to FD But other Redundancy may be there (MVD's)

- BCNF Decomposition in which every sub-Relation is in BCNF
- 3NF Decomposition in which every sub relation is in 3NF

note: every Relation with atmost 2 attribute is always in BCNF

Boyce Codd NF [BCNF]

{R is in BCNF} ⟷ {0% redundancy in Rel R ovr FD's}

Rel R is in BCNF iff every non trivial FD $X \to Y$ with X must be superkey

Non Trivial FD: $\underbrace{X}_{S.K} \to Y$

** Find highest NF satisfied for the give relation:-

(1) R(ABCDE)

FD: { $ABD \to C$, $BC \to D$, $CD \to E$ }

{ $ABD^+ = ABCDE$, $ABC^+ = ABCDE$ }

	$ABD \to C$	not S.K $BC \to D$	not S.K $CD \to E$
BCNF	✓	×	×
3NF	✓	✓	×
2NF	✓	✓	✓

1NF

2NF test:- [proper subset of the C.K]$^+$ = {only prime attribute}

these means no partial dependency hence it's in 2NF

$BC \to D$, $CD \to E$ (?)

$BC \to E$ which is partial dependency

C.K: ABD ⇒ $A^+ = A$, $B^+ = B$, $D^+ = D$, $AB^+ = AB$, $BD^+ = BD$, $AD^+ = AD$

C.K ABC ⇒ $C^+ = C$, $AC^+ = AC$, $BC^+ = BCDE$ ($BC \to E$ partial dependency (not in 2NF))

if(exist) then not in 2NF

- Partial dependency (PD).
  - proper subset of C.K → nonprime.
- Transitive dependency (if exist then not in 3NF)
  - non-prime (Z) transitively determined by Super Key (X)

X: any C.K
Y: not S.K
Z: non-prime.

$Y \to Z$ ×

$X \to Y$ ✓

1) R(ABCDEGH)

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

(i) R_1(ABC)

FD: { AB → C , BC → A, AC → B }

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

1NF ✓
2NF ✓
3NF ✓
BCNF ✓

(ii) R_2(ABCD)

FD: { AB → CD
AC → BD
BC → AD }

$A^+ = A$
$AB^+ = ABCDEG$
$AC^+ = $ " —
$BC^+ = $ " —

2NF ✗
3NF
BCNF

(iii) R(ABCEG)

FD: {
AB → CEG
AC → BEG
BC → AEG }

2NF ✗

(13) R(ABCD)

FD: {
A → B, ✓
B → C, ✓
C → D, ✓
D → B ✓ }

the decomposition R into

	R_1(A,B)	R_2(B,C)	R_3(B,D)
	$A^+ = AB$ $B^+ = B$	$B^+ = BC$ $C^+ = CB$	$B^+ = BD$ $D^+ = DB$
	A → B	B → C C → B	B → D D → B

$R_1 \cup R_2 \cup R_3 = R$ ✓

R_1(AB) R_2(B,C)

$B^+ = BCD$

R_2(ABC) R_3(B,D)

$B^+ = BCD$

R_{123}

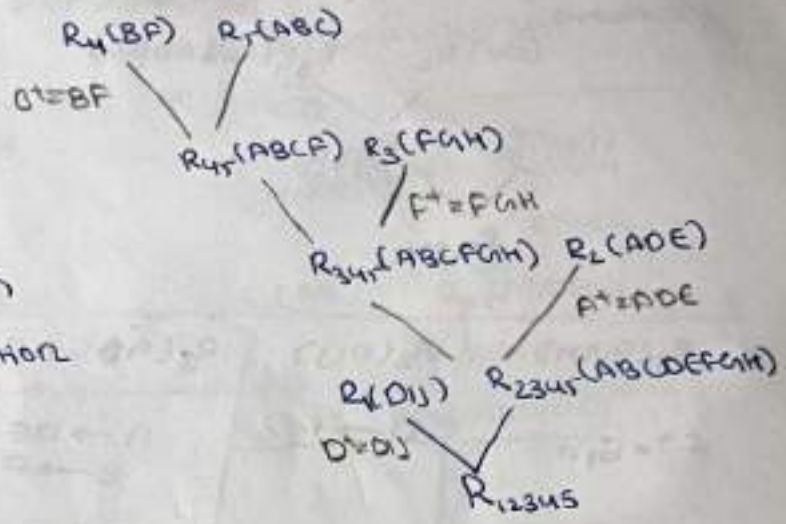
So. its both lossless join & Dependency preserving.

Q12) (a) R(ABCDEFGHIJ)

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

So, R is lossless join
& it also DP relation

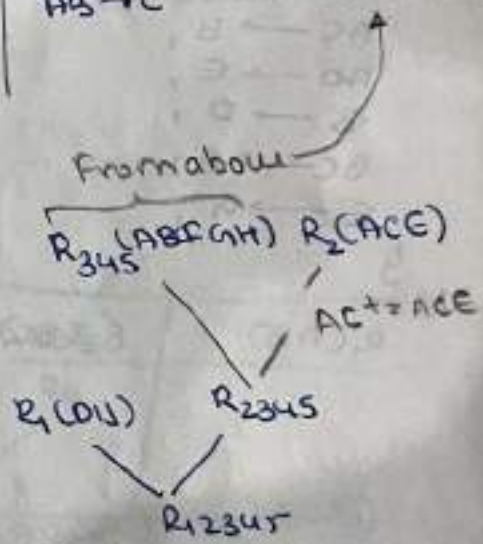
(i) $D_1 = \{DIJ, ADE, FGH, BF, ABC\}$
$R_1 \quad R_2 \quad R_3 \quad R_4 \quad R_5$



$R_1(DIJ)$	$R_2(ADE)$	$R_3(FGH)$	$R_4(BF)$	$R_5(ABC$
D → IJ	A → DE	F → GH	B → F	AB → C

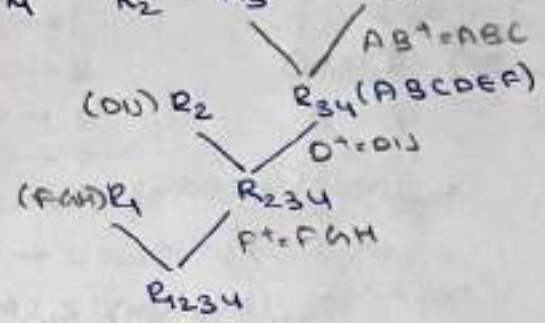
(ii) $D_2 = \{DIJ, ACE, FGH, BA, ADC\}$
$R_1 \quad R_2 \quad R_3 \quad R_4 \quad R_5$

So, its loosless join
but it not
DP relation



$R_1(DIJ)$	$R_2(ACE)$	$R_3(FGH)$	$R_4(BF)$	$R_5(ABC)$
D → IJ	A → E	F → GH	B → F	AB → C

(D3) ρ FGH, DIJ, ADEBA (ABC)
R1 R2 R3 R4



So its lossless join
& also DP relation

R1(FGH)	R2(DIJ)	R3(ADEBA)	R4(ABC)
F→GH	D→IJ	A→DE B→A	AB→C

(b) R(ABCDEGH)

FD: ρ AB → C,
AC → B,
AD → E,
B → D,
BC → A,
E → G,
}

(D1) ρ AB, BC, ABDE, EG }
R1 R2 R3 R4

R1 ∪ R2 ∪ R3 ∪ R4 ≠ R b/c H is missing

R1(AB)	R2(BC)	R3(ABDE)	R4(EG)
		AB→DE AD→E B→D	E→G

So, it neither lossless or dependency preserving

(D2) ρ ABC, ACDE, AD(G)
R1 R2 R3

R1 ∪ R2 ∪ R3 ≠ R (H is missing)

R1(ABC)	R2(ACDE)	R3(ADG)
AB→C AC→B BC→A	AD→E AC→D	AD→G

So its not lossless nor Dependency preserving.

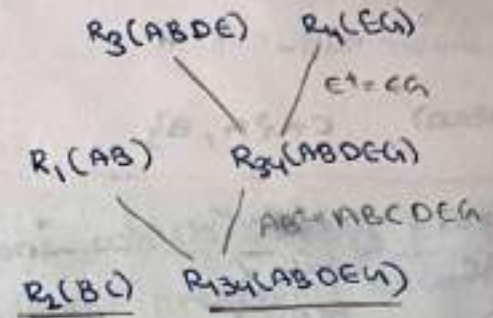
(c) R(ABCDEG)

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

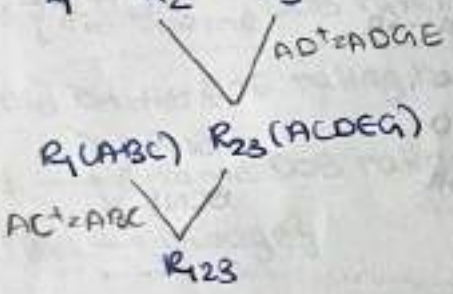
(D1) {AB, BC, ABDE, EG}
R1 R2 R3 R4

Same as above

So, it is not dependency preserving and it's lossless join



(D2) {ABC, ACDE, ADG}
R1 R2 R3



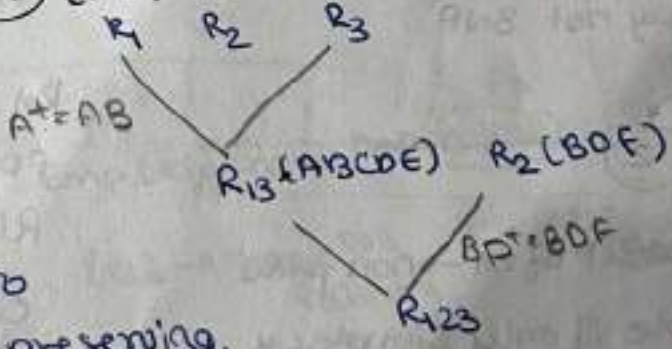
So, its lossless join & its not Dependency preserving.

(D) R(ABCDEF)

FD {A → B,
AC → DE,
BD → F
}

So, its lossless join & also Dependency preserving.

(D1) {AB, BDF, ACDE}
R1 R2 R3



R1(AB)	R2(BDF)	R3(ACDE)
A → B	BD → F	AC → DE

4NF

Find highest NF of given Relation schema?

① R(ABCD) C.K:{AB}
FD's
AB → C, 2NF ✓
BC → D 3NF ✗
BCNF ✗
(BC: not a proper subset of C.K)

② R(ABCD) C.K's {AB, CB}
FD's
AB → C, 2NF ✗
C → A, 3NF ✗
AC → D BCNF ✗
(AC: is union of a C.K; 2 proper subset)

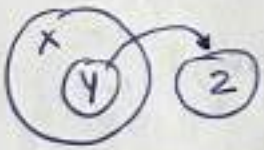
③ R(ABCD) C.K's {A, B}
FD's
AB → B, 2NF ✓
B → AC, 3NF ✗
C → D BCNF ✗

④ R(ABCDEF) C.K's {AB, FB, EB, CB}
FD's
AB → C, 2NF ✗
C → DE, 3NF ✗
E → F, BCNF ✗
F → A

⑤ R(ABCDEF) C.K's {AB, AEF, ADE, ACE, CDF}
FD's AB → C, 2NF ✓
C → D, 3NF ✓
CD → EA, BCNF ✗
DE → F,
Ef → B,

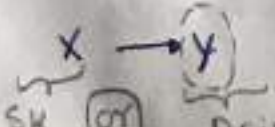
⑥ R(ABCDE) CK.'s {AE, DE, CE, BE}
FD's A → B, 2NF ✓
B → C, 3NF ✓
C → D, BCNF ✗
D → A

⑧ R is only simple C.K
2NF may not 3NF



Proper subset of CK → non-prime
not possible if only simple C.K

⑨ R is only prime attribute.
3NF but may not BCNF
[R with only prime attr.]



⑩ Rel" R with with no trivial FD's always

Rel R with no. non-trivial FD's
R(ABC) no. non-trivial FD's
CK : ABC
no redundancy over FD's
[R in BCNF]

Always in BCNF may not 4NF

(ip) Rel^n R with only 2 Attribute

R(A B)
- {A → B} A:CK
- {B → A} B:CK
- {A → B, B → A} A,B : CK
- {no non-trivial FD's} AB:CK

(12) If Relation R is in 3NF but not in BCNF
Then. (1) Atleast two compound C·K in R
(2) overlapped C·K in R
(3) Proper subset of C·K determine proper subset of other CK exist in R.

Decomposition of Rel into higher NF

• Decompose Rel. R into Sub Relations which satisfy 2NF, 3NF, BCNF with lossless join & dependency preserving decomposition

Q) Emp (eid ename DoB rating hwage Pid)

FD{ eid → ename DoB rating, eid → hwage } == PD { eid → hwage } PD

2NF decomp:-

eid$^+$ = {eid ename DoB rating hwage}

Emp$_1$(eid ename DoB rating hwage) {eid → ename DoB rating, rating → hwage}
C·K {eid}

Emp$_2$(eid Pid) {}
C·K {eid Pid}

2NF ✓ LLJ ✓ DP ✓

non trivial fd's
eid Pid : C·K

Into here

these eid can't be P·K & the other as F·K b/c F·K refers to P·K & there is no (eid) P·K, (eid Pid) logically P·K

3NF decomp:

Emp$_3$(rating hwage) {rating → hwage}
BCNF
rating : CK

Emp$_1$(eid ename DoB rating) {eid → ename DoB rating}
BCNF

FK

F·K

Emp(eid pid)
BCNF

LLJ ✓ DP ✓ BCNF ✓ 3NF ✓

(2) R(A B C D E F)

AB : C·K

FD = { AB → C,
C → D,
B → E,
E → F
}

PD

R in 1NF but not in 2NF

2NF decomp :-

Emp1(A B C D)
AB : C·K

Emp2(B E F)
B : C·K

B → E
B → F
PD

2NF decomp :- { min. 2 table 11 1FK }

B+ = { B E F }

FK

R1(B E F)
B → E
E → F

R2(A B C D)
AB → C
C → D

LLJ ✓
DP ✓
NO PD ✓
2NF

3NF decomp :-

R3(E F) R1(B E) R4(C D) R2(A B C)
E → F B → E C → D AB → C

AB → C
C → D

3NF ✓ LLJ ✓
BCNF ✓ DP ✓

G1 : R(A B C D E F G H)

FD = { AB → C,
C → D,
A → E,
E → F,
B → G,
G → H
}

C·K { AB }

2NF decomp :

R1(A E F) R2(B G H) R3(A B C D)

{ AB → C, C → D }
not BCNF

{ A → E, E → F }
not BCNF

{ B → G, G → H }
not BCNF

3NF decomp :-

R1(A E F)
A → E

R2(B G H)
B → G

R21(B G)
B → G

R22(G H)
G → H

R3(A B C D)

R31(A B C)
AB → C

R32(C D)
C → D

R12(A E)
A → E

R13(E F)
E → F

Q] R(ABCD)

FD's { A → B, C → D }

C.K = AC

How many min. Rel for 2NF decom-? (3)

$R_1(AB)$	$R_2(CD)$	$R_3(AC)$
$A \to B$	$C \to D$	no non-trivial FD's

Q] R(ABCDE)

FD's { AB → C, BC → A, AC → B }

C.K [ABDE, BCDE, ACDE]

3NF but not in BCNF

BCNF Decomposition :-

$R_1(ABC)$ — F.K — $R_2(ABDE)$

R_1: AB → C, BC → A, AC → B

[AB, BC, AC] C.K

R_2: no non-trivial FDs

LLJ ✓ DP ✓ BCNF ✓

Q] R(ABC)

FD's { AB → C, C → A }

C.K = AB, CB

3NF

BCNF Decomposition :-

$R_1(CA)$: C → A, BCNF

$R_2(ABC)$: AB → C

$R_2(BC)$: BCNF

R_2: LLJ ✓ BCNF ✓ DP ✗

[Above Rel not possible BCNF & DP decomp.

Q] R(ABCD)

FD's { ABC → D, D → C }

C.K { ABC, ABD }

BCNF decomposition :-

$R_1(DC)$: D → C

$R_2(ABD)$: no non-trivial FDs

R_1: LLJ ✓ BCNF ✓ DP ✗

- 3NF is most accurate normal form than others
- every relation can decompose into 1NF, 2NF, 3NF, BCNF with lossless join decomposition

DB Design goal	1NF	2NF	3NF (most accurate NF)	BCNF	4NF
1. LLJ decomposition	yes	yes	yes	yes	may be not
2. DP decomposition	yes	yes	yes	may be not	may be not
3. 0% redundancy	no	no	no	yes (FD) / no (MVD)	yes

preferred (NF)

- every relation can decompose into 1NF, 2NF, 3NF with lossless join decomposition & dependency preserving decomposition
- not every relation can decompose into BCNF, 4NF with dependency preserving decomposition
- 5NF: [Join dependency NF] [JDNF]

  decomposition of R into subRel $R_1, R_2 \dots R_n$ given decomposition in 5NF iff

  (1) every subrelation must be in 4NF

  (2) $\bowtie(R_1, R_2, R_3 \dots \bowtie R_n) = R$

  [lossless join]

- 6NF [Domain Key NF] [DKNF]

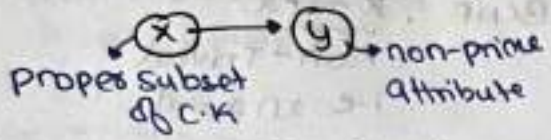
  decomposition of R into SubRel $R_1, R_2 \dots R_n$ is in 6NF iff

  (1) decomposition must be 5NF

  (2) every FD, MVD must preserved in subRels

  Dependency preserving must be true for FDs & MVD

• Decomposition into 2NF
let Relation R be not in 2NF
So, there is a vidation of 2NF



ex: R(a,b,c) FD={bc}

b→c violation of 2NF

$R_2(bc)$ $R_2(b,c)$

$R_1(a\,b)$ $R_2(b,c)$

Keeping for lossess decomposition

the violation of 2NF should be Separate from orginial table

$R(R-X^+, X)$ $R_2(X^+)$

↓ we are keeping in R

ex: R(a,b,c,d,e,f) Decompose into 2NF

FD{ a→bc
d→e
ad→f }

C.K: {ad}
prime: a,d
non prime: b,c,e,f

a→b
a→c
d→e
ad→f
} voilation of 2NF

So, spliting the table

you can also do it one by one

So
$R_1(R_1-a^+ \to a,\ -d^+ \to d)$ $R_2(at)$ $R_3(dt)$

$R_1(adf)$ $R_2(abc)$ $R_3(de)$
ad→f a→bc d→e

So, lossless, dep. preserving

▸ Decomposition in 2NF must be lossless

• every Relation can be Decomposed into 2NF such that Decomposition is lossless & Dep. preserving

ex: when do we say that S is in BCNF?

S={R1,R2,R3,R4}

A database {R1,···;Rn} is in BCNF if each releation Ri is in BCNF (must be in lossless)

↳ & dep. preserving if possible

• A database {R1,···,Rn} is in 3NF if each relation Schema Ri is in BCNF

BCNF Algo.

Guarrantee lossless BCNF Decomp.)

① if R is Already in BCNF : STOP

② if R is not in BCNF

Take a "violation" of BCNF : $X \to \alpha$ (not S.K)

non-Trivial i.e $X \cap \alpha = \phi$

Find $(X^+) = R_1$

$R_2 = (R - X^+) \cup (X)$

ex. R(ABC)

FD = {A → B} violation of BCNF

non-trivial

not S.K

C.K : AC

$R_1(AB)$ $R_2(AC)$

So, Lossless

$A \to B$ violation

$A^+ = AB = R_1(AB)$

$R_2 = (R - A) \cup (A) = A, C$

Remove RHS

Don't remove LHS

ex: R(ABCDE) with $A \to B$; $A \to C$

violation violation

Take a violation:

$A \to C$ violation

Find $A^+ = ABC = R_1(ABC)$

$R_2 = (R - A^+) \cup (A) = R_2(AD)$

$R_2(A,D)$ is in BCNF (2 variable attributes are always in BCNF so there is no non-trivial FD.)

$R_1(ABC)$ $A \to BC$ C.K (S.K) is in BCNF

note: you may get Diffrent BCNF Decompostn depending on the order of violations that we take.

ex: R(ABC) with {AB → C, C → B}

C.K : AB, AC

is creating voilation

$AB \to C$; $C \to B$

To preserve this A, BC must be in single table } not in BCNF

So, $R_1(AC)$ $R_2(CB)$

R(ABC)

lossless BCNF Decomp. Always possible

if we achieve these then (these is DOSH) [BCNF, Lossless, Dep. preserving } can't be together for these ex.

note

① for All relations, lossless BCNF Decomposition is Always possible

② for some relation, lossless, Dep. preserving BCNF Decomposition Doesn't exist

note: it's always possible for every Relation to get some lossless BCNF decomp.

3NF Decomposition

The Synthesis Algorithm } will guarantees lossless & dep. preserving decomposition

Step 1: find minimal cover (F_c) of FD's

Step 2: in F_c; if LHS is same then merge those FD's

$X \to A_1, X \to A_2$

$X \to A_1A_2$

Step 3: For "every FD" (not only for violation)

$X \to A_1A_2$

$R_1(XA_1A_2)$

↑ create a table

Step 4: If none of the Relation from Step 3 contain a C.K then take any one C.K & make one Separate relation for it

- 2NF Decomp. Algo. ≡ BCNF Decomp. Algo.
  Similar in term of process (Both focus on violation)
- 3NF Synthesis Algo. is very Different

Step 5: Delete Some relation: if $R_1 \supset R_2$ then Delete R_2.

note → it's doesn't matter whether the given relation is in 3NF or not
After applying the Algo. it will create 3NF decomp. which is lossless & Dep. presen

Ex: R(ABCDE)

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

C.K: ACE, ABE

Prime: A, B, C, E

nonprime: D

no need to find All violations

So, $A \to D$ (not S.K → non prime) So, not in 3NF

① Already minimal cover

② if common LHS, then merge

③ $R_1(ABC)$ $R_2(CB)$ $R_3(AD)$

④ Does any Relation of step 3. contain some C.K (no)
Add one table for any one C.K
- ABC, CB, AD, ACE

⑤ Remove Some Relation $ABC \supset BC$
ABC, AD, ACE] is 3NF decomp. ✓
{ Lossless ✓
Dep. preserving ✓

ex: R is Already in BCNF ⇒ "minimal cover"

Apply 3NF Synthesis Algo.

$X \to \alpha$ (S.K) → Algo → $X\alpha$ One Relation

can it make Some Relation non-BCNF? (no)

ex. Do we ever need to verify any FD violation?

→ while Insertion/updation yes

→ while deletion (no)

can never violate F.D

never trigger F.D checking

ex: R(ABC) (A→B, B→C, A→C): if

Decomp

AB — can enforce A→B

BC — can enforce B→C

A→C already enforced in individual Rel'n by transitivity

(automatically (Transitively))

Dependency preserving

Each dependency of F can be Enforced by dealing with an individual relation in D.

(note) → If your Decomp. is NOT Dep. preserving then before insertion, modification we may have to join two or more table to enforce some FD's.

R_1

A	B
1	4
1	4

A→B

R_2

B	C
4	2
5	3

Can't be same

B→C

A	C
1	2
1	3

never happen (can you guarantee that there will never happen in Database) ---- yes.

ex: R(ABC)

FDs: AB→C, C→B

R(ABC)

R_1(BC) — can enforce C→B

R_2(AC)

B	C
2	3
2	4

A	C
1	3
1	4

A	B	C
1	2	3
1	2	4

possible to occur if we only check individual Table

To make sure it doesn't happen for every Insertion you have to join R_1 & R_2 first

Database

AC BC

Joining them before every Insertion modification can enforce AB→C

ex: A Decomp. D of Relation R is not dependency preserving if some dependencies of F need join of one (or) more relation of D to enforce them.

Understanding Redundancies due to FD & MVD's

→ non-trivial F·D may create Redundancies in the database

→ trivial FD can't create redundancies

[non-trivial FD in which LHS is a Super key never create Redundancies in the database.

i.e $X \rightarrow \alpha$ — L·H·S is S·K

• FD which may cause Redundancy:

$X \rightarrow \alpha$ → non-trivial & X is (not S·K)

BCNF

Every non-trivial FD has L·H·S as S·K

→ Free from Redundancy ✗

→ Free from Redundancy caused by FD's

} FD can't cause redundancies but someone else can create

note

• BCNF is 100% free from "Redundancies due to FD's"

• 3NF may have "Redundancies due to FD's"

ex:

course(sid, cid) — many to many relationship

movies(sid, movies) — many to many relationship

merge

Student(sid, cid, movie)

no non-trivial F·D

(still Redundancies)

B/c we put more than one "many-many" relationships in single table } know as MVDs (multi valued dependencies)

there is no redundancies

sid → cid

sid, cid → movie

sid + course — join — sid + movie

sid + cid + movie

problem

a new type of Dependency

• every movie cause Repetition of all courses. or vice versa

- every BCNF Decomposition is lossless false
- every 3NF -"- -"- false

True Stmt.

- for every Relation R, there exist a decomposition into 3NF is lossless & dep. preserving
- for every Relation R, there always exist a decomposition into BCNF that is lossless But may or may not be dep preserving.

Queries

4marks

- Theorethical
1) procedural Query lang.
↳ Formulation of what data retrieved & how data retreives database table
[ex:- Relational Algebra]

- pratical
1. SQL
2. QUEL
3. Datalog.

RA/SQL/TRC

2) non-procedural Query lang.
↳ Formulation of what data retrieve from Database table
[ex: Relational Calculs]
Uses predicate calculus & 1st order logic formulas
↳ Tuple Relational calculus [TRC]
[SQL Queries clore to TRC].
↳ Domain Relational calculus [QBE (Query By example [DRC]
Query clore to DRC]

(Pure) Relational model
Relational Algebra
Variable ≡ Relation (Schema)
Value ≡ Instance
not required for gate

Basic operators:-
- π : Projection
- σ : selection
- ρ : Rename
(unary operators)
- ∪ : Union
- − : Minus
- × : Cross product
(Binary operators)

Derived operators:-
- ∩ : Intersection {using "−"}
- ⋈ : Join {using π, σ, ×}
- ÷ or / : Division {using π, ×, −}

Commutative
π × σ

(note) for any "Retrievel Query" original Database never change new Relation may be (temporarily) Created

- R(A, B, C)
order is fixed
Query using R:
R·1 → 1st attribute i.e (a)

π : Projection

π(R) : Retrieves attribute from Relation R those are in projection attr_list list

attr_list → in these order

∴ distinct Attribute

Result of Relation Algebra Query is always distinct record/tuple

- no. of attribute might be different from orginial

ρ : Rename

↳ used to Rename table name or attribute name for Query processing

i.e Stud(Sid Sname DOB) DB Schema

ρ(Temp, Stud) : Temp | Sid | Sname | DOB |

ρ(Stud) : Stud | I | N | D |
I, N, O

ρ(Stud) : Stud | I | Sname | D |
Sid → I
DOB → D

Note: Just used to process the data Attribute name will not be changed in Original table.

doesn't need union compatible

Cross product(x)

R×S: R cross product S result in all attribut of R and all attribute of S & each record of R is paired with each record of S.

Arity: x

R

A	B	C
6	4	8
3	4	6
4	7	8

n distinct tuple

Arity: y

S

C	D
7	8
8	4
9	5

m distinct tuple

Arity: x + y

R×S

A	B	C	C	D
6	4	8	7	8
6	4	8	8	4
6	4	8	9	5
3	4	6	7	8
3	4	6	8	4
3	4	6	9	5
4	7	8	7	8
4	7	8	8	4
4	7	8	9	5

Cardinality of R×S : m·n tuple

σ : Selection

σ_p(R) : retrieves Record of the relation 'R' those are satisfied predicate condition "p".

Set : {2, 6, 8}

Bag : {2, 6, 6, 8, 8} : i.e multi set

R

A	B	C
3	5	6
3	5	6
4	5	6
8	5	4
8	6	4

Bag

π(R) BC

B	C
5	6
5	4

distinct tuple

σ A<5 (R)

A	B	C
3	5	6
4	5	6

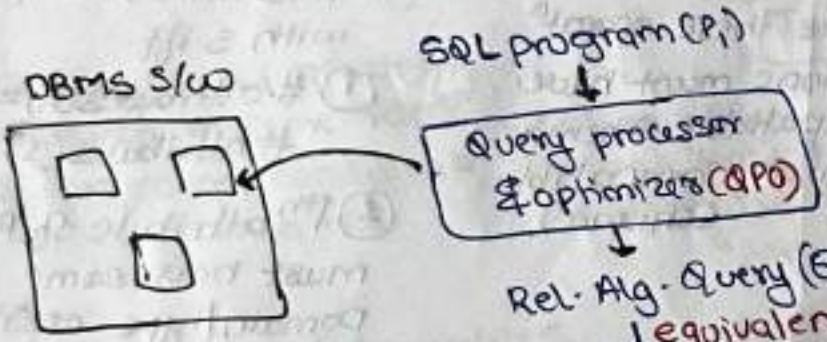
funct^n: $I_1 \rightarrow O_1$, $I_2 \rightarrow O_2$ } for given i/p what o/p will you get

Algo: step-by-step procedure/Recipe
- like spoonfeeding
- Quick sort

- Sort funct^n
  $14,10,18 \rightarrow 10,14,18$
  in what order you get it doesn't matter

▸ Rel. Algebra Query:
- you need to specify: ① what you want ② How to get it (Step by Step)
- Procedural Query lang^n (easier for machine)

(non-procedural)
▸ Rel. Calculs Query
- you need to specify: ① what you want
- Same with SQL
- Declarative Query lang^n (easier for users)



---

note: Relation instance $r(R)$

① $r_1 \leftarrow \sigma_p(r)$ (assignment) o/p r_1

② $\sigma_p(r)$ ⟹ no name for this new o/p table

ex: To make sure that cardinality of projected relation is necessary same as i/p relation for all instance.
▸ attribute list must be s.k

$[\pi_{sname, rating}(\sigma_{rating>8}[S_2])] \equiv [\sigma_{rating>8}(\pi_{sname, rating}(S_2))]$ b/c

"π" use those variable(attribute) that are used in selection.

i.e $\pi_A(\sigma_C(R)) \equiv \sigma_C(\pi_A(R))$

↓ If C uses only attribute of A

ex. which Transformation we can do?

$\pi_A(\sigma_C(R)) \xrightarrow{X} \sigma_C(\pi_A(R))$ } True when it's given that 'C' use's only A.

$\sigma_C(\pi_A(R)) \not\rightarrow \pi_A(\sigma_C(R))$

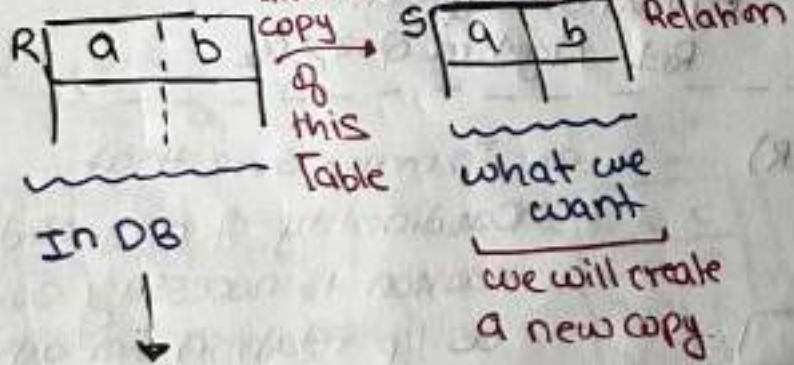
↓ if it's valid then C use's only A.

note

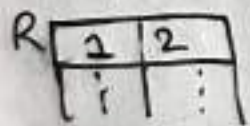
Union, $\cap$, $-$ } ⇒ To Define these operat^n Relations must have Compatiable Schema (union compatible schema).

- R is union-compatible with S iff
  1. #attributes(R) = #attributes(S)
  2. i^{th} attribute of R must have same Domain/type as i^{th} attribute of S.

- Rename(ρ)



$\rho_{(1,2)}(R)$



here table name remain same but attribute name change

ex: Find all true friends in twitter dataset

T(Uid1, Uid2)

A → B
B → A
B → C
A → C
C → B

x → y
y → x

Single table is not enough

idea
$T_1 \times T_2 \Rightarrow x\ y\ y\ x$
desired Rows

sol $\rho_{T_2(Sid_1, Sid_2)} T$

$T_1 \times T_2$

Uid_1	Uid_2	Sid_1	Sid_2
A	B	A	B ... x
A	B	B	A ... ✓

Condition:

$(Uid_1 = Sid_2) \wedge (Uid_2 = Sid_1)$ $(T_1 \times T_2)$

A

$\pi_{Uid_1, Uid_2}(A)$

$T_1 \times T_2$ → usually meaningless

$\sigma_c(T_1 \times T_2)$ → most frequently

$T_1 \bowtie_c T_2$ conditional join

- usually cross product is followed by some select^n & projection conditions to get desired ans.

} so frequently used that we create a new set of operators "join operators"

- Schema:

Schema$(R \bowtie_c S)$ = Schema$(R \times S)$

- natural join (equality of All common attributes)

R | a | b | c |

S | a | b | d |

⇒ o/p | a | b | c | d |

these pair should be same

no redundant attribute

(note) if no common Attribute in R, S then

$R \bowtie S = R \times S$

- Outer join: "Inner join" + Dangling Tuple
  By default 'natural join' + dangling tuples.

(Pure) Relational model
↓
Don't have nulls in the Table (except for outer join)

Rel. Alg. / TRC / DRC
→ Table is a (Set) of tuple
→ no nulls (except for outer join)
→ ∃! P.K
→ ∃CKs, ∃S.K

① Two Rows can never be entirely same
② There is exactly one P.K
③ There is ≥ 1 C.K
u ≥ 1 S.K

Set vs multiset

Set → no Duplicate
unordered
multiset → Duplicate matter

- Set{1,2,2} = Set{1,2}
  no. of element = 2
- {1,1,1,1,2} = {1,2} = {2,1} = {2,2,1,1}

- multiset {1,2,2} ≠ multiset {1,2}
  3 element 2 element
- {1,2} = {2,1}

SQL → Table is a (multiset) of tuple
→ Null's may exist
→ may or may't have any P.K
→ may or may not be any S.K (or) C.K
→ SQL is not case-sensitive

JOIN's

note

$Y = \{\}$ Y empty

$S = \{1,2,3\}$

$S \times Y = \{\}$ Y empty

$R \times S$ result empty record set $\{\}$ i.e zero tuple. if either R or S empty record set

(a) Natural join ($\bowtie$):-

no common attribute among them if $S \cap R = \phi$

$S \bowtie R = S \times R$ Cross product

$$R \bowtie S = \pi_{\text{Distinct Attr.}} \left(\sigma_{\text{Equality b/w Same name attr. of R \& S}} (R \times S) \right)$$

=

A	B	C	C	D

$$R \bowtie S = \pi_{ABCD} \left(\sigma_{R.C = S.C} (R \times S) \right)$$

A	B	C	D
6	4	8	4
4	7	8	4

note

if no common attribute b/w 2 relation then natural join is equal to cross product

$T_1(ABC)$ $T_2(BCDE)$

$$T_1 \bowtie T_2 = \pi_{ABCDE} \left(\sigma_{T_1.B = T_2.B \wedge T_1.C = T_2.C} (T_1 \times T_2) \right)$$

$T_1(AB)$ $T_2(CD)$

$T_1 \times T_2 = T_1 \bowtie T_2$

(b) Conditional join ($\bowtie_c$) (Theta join)

$$R \bowtie_c S = \sigma_c (R \times S)$$

$R \bowtie_{R.C > S.C} S = \sigma_{R.C > S.C} (R \times S)$

A	B	C	C	D
6	4	8	7	8
4	7	8	7	8

Arity of $R \bowtie_c S$ equal to $R \times S$

$R \bowtie_c S = R \times S$

Arity of $R \bowtie S \leq R \times S$

note rows which are not included in the final table are known as Dangling Rows.

C Outer joins:

i) left order Join (⟕) → Dangling tuples will be included using help of null

$R \overset{L}{\bowtie} S = (R \bowtie S) \cup$ (Records of R these are failed Join cond. with Null values over remaining attribute)

A	B	C	D
6	4	8	4
4	7	8	4
3	4	6	null

min max
n mxm

R --- n, S --- m → n+m

ii) Right order Join (⟖) or ($\overset{o}{\bowtie}_R$)

$R \bowtie S$

A	B	C	D
6	4	8	4
4	7	8	4
null	null	7	8
null	null	9	5

min (m)
max (mxn)

$R \overset{R}{\bowtie} S$ R.C>S.C

A	B	C	C	D
6	4	8	7	8
4	7	8	7	8
null	null	null	8	4
null	null	null	9	5

iii) Full outer join (⟗) or (⟗)

$R \overset{F}{\bowtie} S \equiv (R \overset{L}{\bowtie} S) \cup (R \overset{R}{\bowtie} S)$ →

A	B	C	D
6	4	8	4
4	7	8	4
3	4	6	null
null	null	7	8
null	null	9	5

min: max(n,m)
max: mxn

conditional can work as

[model : 1] Some/any/atleast one $\sigma_p(R \times S)$

$R \bowtie_C S$
$R \bowtie S$

R(A...) R(B...)

Retrieve 'A' value of R there are more than some 'B' value of S.

>any

R

A
10
20
30
40

S

B
15
25
30

$\pi_{R.A}(\sigma_{R.A>S.B}(R \times S))$

$\pi_{R.A}(R \bowtie_{R.A>S.B} S)$

A
20
30
40

ex: Emp (eid sal dno)

Retrieve eids whose salary more than any emp sal

$\pi_{Emp.eids}(R \bowtie_{Emp_1.sal > Emp_2.sal \wedge Emp_2.dno=5} S)$ ✗ need to rename

> some of dep: 5

>Some

Emp (eid	sal	dno)
e_1	10	2
e_2	20	5
e_3	30	2
e_4	40	5
e_5	50	3

Emp (eid	sal	dno)
e_1	10	2
e_2	20	5
e_3	30	2
e_4	40	5
e_5	50	3

$\pi_{emp.eid}(Emp \bowtie_{Emp.sal > Temp.sal \wedge Temp.dno=5} \rho(Temp, Emp))$

$\pi_{emp.eid}(\sigma_{Emp.sal > Temp.sal \wedge Temp.dno=5}(Emp \times \rho(Temp, Emp)))$ OR $\pi_{eid}(Emp \bowtie_{Sal>5 \wedge D=5} \rho_{I,S,D}(Emp))$

$\pi_{eid}(Emp \bowtie_{sal>5} \rho_{I,S,D}(\sigma_{dno=5}(Emp)))$ → efficient

(3rd, 2nd to be execute, 4th)

ex: Stud(Sid, gen, age)

Retrieve sid of female Student who age is less than any male Student age

(S) Sid	(G) gen	(A) age
e_1	M	20
e_2	M	21
e_3	F	15
e_4	F	24
e_5	F	16

Sid	gen	age
e_1	M	20
e_2	M	21
e_3	F	15
e_4	F	24
e_5	F	16

Query 1

$\pi_{sid}(Stud \bowtie_{age<A \wedge gen=Male} \rho_{S,G,A}(\sigma_{gen=Female}(Stud)))$

(efficient) Query2.

$$\pi_{sid}\left(\sigma_{gen=female}(Stud) \bowtie_{age<A} \rho_{I,G,A}\left(\sigma_{gen=male}(Stud)\right)\right)$$

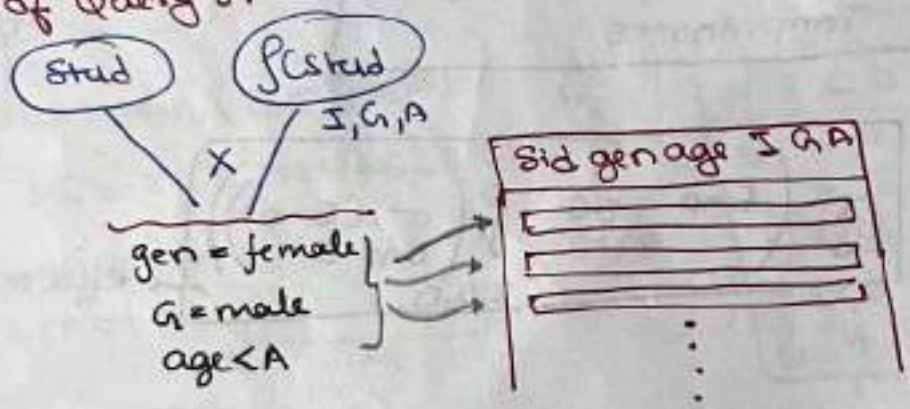
Query3:

$$\pi_{sid}\left(Stud \bowtie_{\substack{gen=female \\ \wedge G=male \\ \wedge age<A}} \rho_{I,G,A}(Stud)\right)$$

Cost of Query: # of comp. required to execute Query

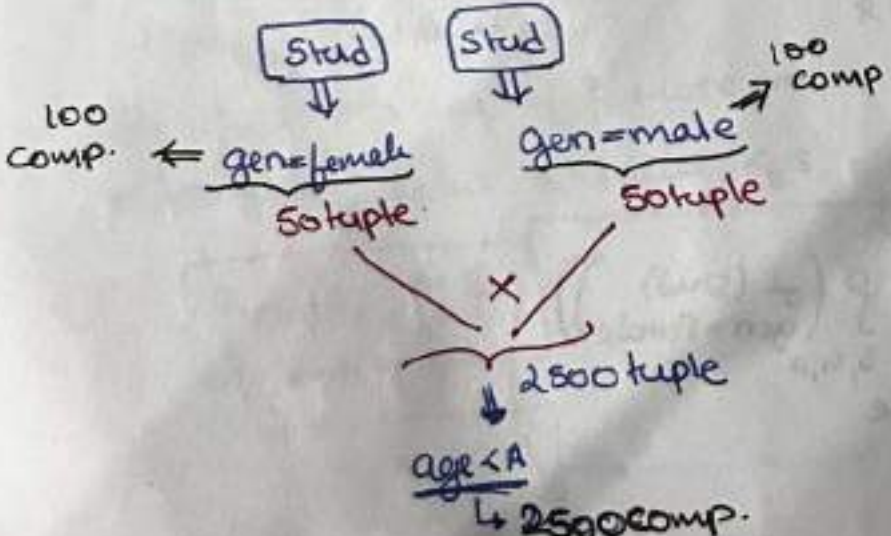
Assume: 100 students.
[50 female, 50 male]

Cost of Query 3:



10000 * 3 = 30000 Comp.

Cost of Query 2:



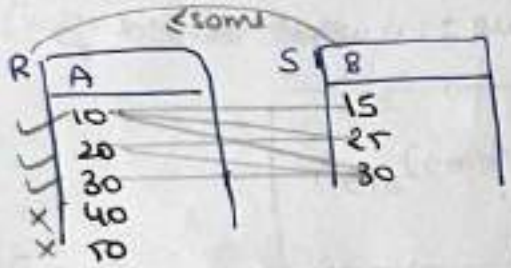
Cost of the query:
100 + 100 + 2500
= 2700 Comp.

Model 2 [All/every]

R(A...) S(B...)

Retrieve 'A' value of R those are (more) than (every) 'B' val of S

$>$



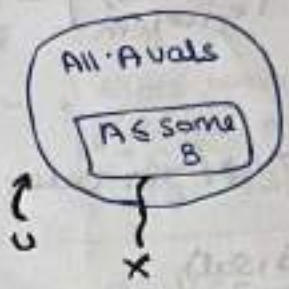
{All 'A' value of R} − {'A' val of R those are ≤some 'B' val of S}

$$\pi_A(R) - \pi_{R.A}(R \bowtie_{R.A \leq S.B} S)$$

not (A ≤ some B) ≡ A > every B

not (A > some B) ≡ A ≤ every B

not (A > every B) ≡ A ≤ some B



{All 'A' values} − {'A' val which are ≤ some B}

Ex: Emp(eid sal dno)

Retrieve eid's whose salary more than every emp sal of dept 5

{eid of all emp's} − {eid's whose sal ≤ some emp sal of dept 5}

$$\left\{ \pi_{eid}(Emp) - \pi_{eid}\left(Emp \bowtie_{sal \leq S} \rho_{E,S,D}\left(\sigma_{dptno=5}(Emp) \right) \right) \right\}$$

Ex: Stud(sid gen age)

Retrieve sid of female student whose age less than every male student

$$\left\{ \pi_{sid}(\sigma_{gen=female}(Stud)) - \pi_{sid}\left(\sigma_{gen=male}(Stud) \bowtie_{age \leq A} \rho_{S,G,A}\left(\sigma_{gen=female}(Stud) \right) \right) \right\}$$

{ Sid's of all female stud } − { sid of female whose age ≥ some male student age }

$\pi_{sid}(\sigma_{gen=fem}(Stud)) - \pi_{sid}(\sigma_{gen=female}(Stud) \bowtie_{age \neq A} \rho_{I,G,A}(\sigma_{gen=male}(Stud)))$

Model3 Max/Min over Attr. value : [it will ignore null value]

Emp (eid, sal)

Retrieve eid's whose salary max.

{ Eid's whose sal ≥ every emp sal } ≡ { eid's of all emps } − { eid's whose sal < some emp sal. }

max — not max

$\pi_{eid}(Emp) - \pi_{eid}(Emp \bowtie_{Sal<S} \rho_{I,S}(Emp))$

* Max val of Attr. X ≡ X val ≥ every value of X
* Min val of Attr. X ≡ X val ≤ every value of X.

eid	sal
~~e_1~~	40
~~e_2~~	60
✓ e_3	90
~~e_4~~	80
✓ e_5	90
e_6	50

eid	sal
e_1	40
e_2	60
e_3	90
e_4	80
e_5	90
e_6	50

Model 4 max/min for group of records. $\forall$(≥every) ≡ ∄(<some)

Emp(eid, sal, dno)

Retrieves eids whose salary maximum for each dept.

{Eids whose sal (≥every) emp sal of his/her dept} ≡ {eid of all emps} − {eids whose sal (<some) emp sal of his/her dept}

$$\pi_{eid}(emp) - \pi_{eid}(emp \bowtie_{sal<s \wedge dno=D} \rho_{I,S,D}(Emp))$$

eid	sal	dno
e1	40	2
e2	60	2
e3	90	2
e4	80	3
e5	70	3
e6	50	4

eid	sal	dno
e1	40	2
e2	60	2
e3	90	2
e4	80	3
e5	70	3
e6	50	4

Ex:- Stud(sid gen age)

Retrive youngest students of each gender.

$$\{\pi_{Sid}(Stud) - \pi_{Sid}(Stud \bowtie_{age>A \wedge gen=G} \rho_{S,G,A}(Stud))\}$$

{sids where age (≤every) age of same gender} ≡ {sid of all studs} − {sids whose age (>some) age of same gender}

ex: Enroll (Sid cid fee) Retrieve sid who paid max fee for each course.

$$\{ \pi_{sid,cid}(Enroll) - \pi_{Sid,cid}(Enroll \bowtie_{fee<f \wedge cid=c} \rho_{S,C,f}(Enroll)) \}$$

{ Sid's paid fee ≥ every student fee of same course } = { all sid's of enroll's } − { sid's paid fee < some stud fee of same course }

note

- It is of some case in which logic of the Query doesn't take full key else its always when we take full key for projection

note

- projection should be done with some key not with non key else you will lost some data

ex:- Stud (Sid Sname Cid) Retrieve Sname who enrolled for only gate

S_1	A	gate
S_1	A	gre
S_2	A	gate
S_3	B	gre
S_4	A	gate

$$\pi_{Sname}(\pi_{Sid,Sname}(\sigma_{Cid=gate}(Stud)) - \pi_{Sid,Sname}(\sigma_{Cid=GRE}(Stud)))$$

question

workbook

28
Q) R(ABCDEFG)

FD's { AB → C, A
BC → A, B
B → D, B
D → E }

ABFG⁺ = BFGDEAC
CBFG⁺ = - - " - -

R1(ABC) | R2(BDE) | R3(ABFG)

R1: AB → C, BC → A

R2: B → D, D → E

R2 → R21(BD): B → D

R2 → R22(DE): D → E

gate 2013
31) R(ABCDEFGH)

FD's { CH → G,
A → BC,
B → CFH,
E → A,
F → EG }

AD⁺ = ABCDEFGH
ED⁺ = - - " - -
FD⁺
BF⁺

so 4 C.K

35) R(STUV)

FD's { S → T,
T → U,
U → V,
V → S }

C.K { S, V, U, T }

R1(ST) R2(UV)

when R1 ∩ R2 = ϕ

so there is not in 2NF

40) R(ABCDE)

FD's { A → BC, ✓
CD → E, ✓
B → D, ✗
E → A }

R1(ABC)	R2(CDE)
A → BC	E → CD
AB → C	CD → E
AC → B	CE → D
BC → A	DE → C

lossy & not DP

(43) (i) R(ABCDE)

FD = {AB → C, C → AB, C → D, D → E}

C.K = AB, C

$R_1(ABC)$	$R_2(CDE)$
AB → C	C → D
C → AB	D → E

$R_{21}(CD)$	$R_{22}(DE)$
C → D	D → E

So its lossless join & DP

(ii) R(ABCDE)

FD = {AB → C, C → A, C → D, D → E}

C.K : AB, CB

$R_1(ABC)$: AB → C, C → A

$R_2(CDE)$: C → D, D → E

B/c of two it not in BCNF

(H.W) R(ABCD)

FD = {A → B, C → D}

C.K = AC

$R_1(AB)$

$R_2(CD)$

$R_3(AC)$

So 3 Reln required for 2NF

R(ABCDEF)

FD = {A → D, B → E, C → F}

C.K ABC

$R_1(AD)$

$R_2(BE)$

$R_3(CF)$

$R_4(ABC)$

So. 4 Reln required for 2NF

R(ABCDEFGH)

FD = {AB → C, C → D, A → E, E → F, B → G, G → H}

C.K = {AB}

$R_1(AEF)$

$R_2(BGH)$

$R_3(ABCD)$

$R_{11}(EF)$, $R_{12}(AE)$

$R_{21}(BG)$, $R_{22}(GH)$

$R_{31}(ABC)$, $R_{32}(CD)$

In 2NF, 3 Reln Required

In 3NF, 6 Reln Required

R(ABCD)

FD's { AB → C, BC → A, AC → B } 1's in 2NF ✓ 3NF ✓ BCNF ✗

C.K's { DAB, DAC, DBC }

So,

R_1(ABC)	R_2(ABD)
AB → C	
BC → A	
AC → B	

1's in BCNF

➡ Set operators:

- ∪ : Union (UNION) ——— SQL ——→ OR
- − : Minus (EXCEPT / MINUS) ——→ only / but not
- ∩ : Intersection (INTERECT) ——→ AND

➡ R∪S, R−S, R∩S

R & S must be union compatable } same SQL set operation

→ means same no. of attribute & corresponding attribute have the same data type

➡ R & S Union Compatable iff

① Arity of R = Arity of S and

② Domain of each Attr. of R must be same domain for S Attributes respectively

Domain(A): possible val's Accept by Attr.

note

order doesn't matter for join but matter for set difference

i.e SidSname ≠ SnameSid

union compatable

1] $\pi_{Sid}(\ldots) \cup \pi_{Sname}(\ldots)$ ✗

2] $\pi_{Sid\,Sname}(\ldots) \cup \pi_{Sid\,DOB}(\ldots)$ ✗

3] $\pi_{Sid\,Sname}(\ldots) \cup \pi_{Sname\,Sid}(\ldots)$ ✗

Sid, Sname are different

4] $\pi_{Sid,Sname}(\ldots) \cup \pi_{StudID,StudName}(\ldots)$ ✓

different name but store same value (Domain of both should be same)

ex R(AB) S(BA)

$R \cap S = \pi_{AB}(\sigma(R \times S))$

R.A = S.A

R.B = S.B

Model: 5 (set difference)

⇒ R∪S, R∩S, R−S

1. Schema of result
   Same as R
   ⇓
2. Distinct tuple in result

} Same in SQL

R

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

S

D	E	F
2	4	6
2	4	6
4	3	5
7	4	2

R∪S

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

Tuple belong to R or S [distinct]

R−S

A	B	C
3	5	6
4	8	7

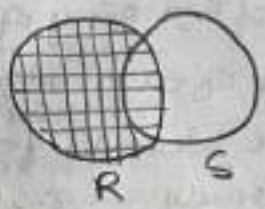
Tuple belong to R but not S

R∩S

A	B	C
2	4	6

Tuple from R ∩ S

∪, −: Basic operator
∩: Derived operator



R−(R−S) = R∩S

① $R \cap S = R - (R - S)$ → Best possible ways

② $R \cap S = \pi_{A,B}(\sigma_{R.A=S.C \wedge R.B=S.D}(R \times S))$

③ $R \cap S = \pi_{A,B}(R \bowtie_{R.A=S.C \wedge R.B=S.D} S)$

④ $R \cap S = R \bowtie_{C \to A,\ D \to B} f(S)$

ex: Stud(Sid Sname age gender)

S_1
S_2
S_3
S_4
S_5
S_6

Course(Cid Cname Instructor)

C_1	DB	Korth
C_2	DB	Korth
C_3	DB	navathe
C_4	DB	navathe
C_5	Algo	coreman
C_6	OS	galvin

Enroll (Sid Cid fee)

S_1	C_1
S_1	C_2
S_1	C_3
S_2	C_2
S_3	C_1
S_4	C_5

Stud (Sid Sname age) Course (Cid ...)

Enroll (Sid Cid fee)

i) Retrieve Sid enrolled (some) Course taught by Korth

$$\pi_{Sid}\left[\text{Enroll} \bowtie \sigma_{\text{instructor}=\text{Korth}}(\text{course})\right]$$

ii) Retrive Sid's enrolled for only Course taught by Korth

{Sid's enrolled some Course} − {Sid's enrolled some non-Korth}

$$\pi_{sid}(\text{enroll}) - \pi_{sid}(\text{Enroll} \bowtie \sigma_{\text{Inst}\neq\text{Korth}}(\text{Course}))$$

iii) Retrieve Course name enrolled "_" only female student.

$$\pi_{cname}(\text{Temp} \bowtie \text{Course})$$

- Set difference is used when Keyword is used "only but not".

$$f\left(\text{Temp}, \pi_{cid}(\text{Enroll}) - \pi_{cid}(\text{Enroll} \bowtie \sigma_{\text{gen}\neq\text{female}}(\text{Stud}))\right)$$

> Cid's enrolled by some female stud

$$\pi_{cid}(\text{Enroll} \bowtie \sigma_{\text{gen}=\text{female}}(\text{Stud}))$$

note

$\sigma_{gen=female}$ is used to calcula for one value no for group of val

iv) Retrieve Sid's enrolled some course taught by Korth or some course taught by navathe

$$\pi_{sid}\left(\text{Enroll} \bowtie \left(\sigma_{\text{inst.}=\text{Korth}}(\text{Course}) \cup \sigma_{\text{inst.}=\text{navathe}}(\text{Course})\right)\right)$$

Query 1

$$\pi_{Sid}\left(\text{Enroll} \bowtie \sigma_{\substack{(\text{Course}) \\ \text{Inst.} = \text{Korth} \vee \\ \text{Inst.} = \text{Navathe}}}\right)$$

Query 2

$$\pi_{Sid}\left(\text{Enroll} \bowtie \sigma_{\substack{(\text{Course}) \\ \text{Inst.} = \\ \text{Korth}}}\right) \cup \pi_{Sid}\left(\text{Enroll} \bowtie \sigma_{\substack{(\text{Course}) \\ \text{Inst.} = \\ \text{navathe}}}\right)$$

(v) Retrieve Sid's enrolled some course taught by Korth and some course taught by navathe

wrong (a)

$$\pi_{Sid}\left(\text{Enroll} \bowtie \sigma_{\substack{(\text{Course}) \\ \text{Inst} = \text{Korth} \wedge \\ \text{Inst} = \text{Navathe}}}\right)$$

Always empty record in result

Inst = Korth ∧ Inst = Navathe → empty

(b)

$$\pi_{Sid}\left(\text{Enroll} \bowtie \left(\sigma_{\substack{(\text{Course}) \\ \text{Inst.} = \\ \text{Korth}}} \cap \sigma_{\substack{(\text{Course}) \\ \text{Inst.} = \\ \text{navathe}}}\right)\right)$$

(empty) ∩ (empty)

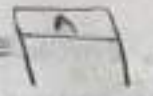
empty Sid record set

(c)

$$\pi_{Sid}\left(\text{Enroll} \bowtie \sigma_{\substack{(\text{course}) \\ \text{Inst.} = \\ \text{Korth}}}\right) \cap \pi_{Sid}\left(\text{Enroll} \bowtie \sigma_{\substack{(\text{Course}) \\ \text{Inst.} \\ = \text{navathe}}}\right)$$

Sid: S_1, S_2, S_3 ∩ S_1

Division [÷ or /]

using $\pi, \times, -$ ≡  π_{AB} / π_B

R(AB) S(B)

$\pi_{AB}(R) / \pi_B(S)$: Retrieves all "A val of rel R for with there must be tuple (a,b) in R for every 'B' value of S

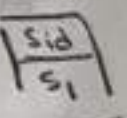
[pair with every]

R(A B)	S(B)
a_1 b_1	b_1
a_1 b_2	b_2
a_1 b_3	b_3
a_2 b_1	
a_2 b_3	
a_3 b_2	
a_3 b_3	

O/P : a_1

Qus] Enroll(sid cid) Course(cid ....)

Retrieves sid's enrolled every course.

$\pi_{sid,cid}(Enroll) / \pi_{cid}(Course)$ = 

O/P

a particular sid should be paired with every course of Course table

model 6 (Division question)

Expansion of the Division (/):-

$\pi_{sid,cid}(Enroll) / \pi_{cid}(Course)$: Sid's enrolled every course.

1 Sid's not enrolled for every course

[Sid's enrolled for proper subset of all courses]

π_{sid} ($\pi_{sid}(Enroll) \times \pi_{cid}(Course)$ − $\pi_{sid,cid}(Enroll)$) : disqualified tuples for division

every sid of enroll rel pair with every course

given enrollment

Sid	Cid
S_2	C_1
S_3	C_2
S_3	C_3

=

Sid	Cid
S_1	C_1
S_1	C_2
S_1	C_3
S_2	C_1
S_2	C_2
S_2	C_3
S_3	C_1
S_3	C_2
S_3	C_3

−

Sid	Cid
S_1	C_1
S_1	C_2
S_1	C_3
S_2	C_2
S_2	C_3
S_3	C_1

2. {Sid's enrolled every course} = {Sid's enrolled some course} − {Sid's not enrolled every course}

[enrolled proper subset of every course]

$$\pi_{Sid,Cid}(Enroll) \,/\, \pi_{Cid}(course) \equiv \pi_{Sid}(Enroll) - \pi_{Sid}\left(\pi_{Sid}(Enroll) \times \pi_{Cid}(course) - \pi_{Sid,Cid}(Enroll)\right)$$

$$\pi_{AB}(R) \,/\, \pi_{B}(S): \quad \pi_{A}(R) - \pi_{A}\left(\pi_{A}(R) \times \pi_{B}(S) - \pi_{AB}(R)\right)$$

$$\pi_{ABC}(R) \,/\, \pi_{C}(S): \quad \pi_{AB}(R) - \pi_{AB}\left(\pi_{AB}(R) \times \pi_{C}(S) - \pi_{ABC}(R)\right)$$

$$\pi_{ABCD}(R) \,/\, \pi_{CD}(S): \quad \pi_{AB}(R) - \pi_{AB}\left(\pi_{AB}(R) \times \pi_{CD}(S) - \pi_{ABCD}(R)\right)$$

$\pi_{Sid,Cid}(R) \,/\, \pi_{Sid,Cid}(R)$ ✗

$\pi_{Sid,Cid}(R) \,/\, \pi_{Sid}(S)$ ✗

Should be a proper subset

NOTE

Some → atleast 1

Every → All

Some course all → Enrolled some ⊃ Enrolled all Course

not enrolled every course ≠ enrolled some course

Student enrolled for 0 course

Student enrolled every course

not enrolled some Course = enrolled no Course

$\sim \exists x (P(x)) \equiv \forall x (\sim P(x))$

• Division operator in mathematical algebra.

ex: $8/4 = 2 \Rightarrow 2 \times 4 \leq 8$

$8/3 = 2$ ← largest integer

↓ Quotient

such that $2 \times 3 \leq 8$

$\frac{N}{d} = Q$ } largest integer s.t $Q \cdot d \leq N$

ex: $19/5$ (N / d)

$5 \times 2 \leq 19$ ← not largest

largest integer

$Qd \leq N$

$3 \times 5 \leq 19$

$\frac{A}{B} = \text{Result} = Q$

s.t Q is 'largest' integer to satisfy

$Q * B \leq A$

▸ Same for Relation

Relation R, S

$\frac{R}{S} = Q$

↳ Relation

Such that Q is the largest Relation s.t

$Q \times S \subseteq R$

ex.

a	b
1	1
1	4
2	1
2	2
2	3
2	4
3	1
3	3
3	4

(a) r_1 (dividend)

b
1
3

(b) r_2 (divisor)

$\frac{r_1(a,b)}{r_2(b)} = Q$ } $Q \times r_2 \subseteq r_1$

schema schema

Schema of Q: ~~a b~~

$Q \times r_2$

a b b

Schema of Q: a

$Q \times r_2$

a b

$Q \times r_2 \subseteq r_1$

Q must be largest

$\frac{r_1}{r_2} = Q(a)$

a
1 X
2 ✓
3 ✓

$\frac{r_1(a,b)}{r_2(b)} = Q(a)$

a
2
3

→ largest Relation to satisfy

▸ $\frac{R(A,B)}{S(B)} = Q(A)$

$Q \times S \subseteq R$

To Define $R \div S$

attribute(R) ⊃ attribute(S)

note

$\frac{R(ABCD)}{S(BD)} = Q(AC)$

none of R, S, Q can be same

↓

none of them have same schema.

$\frac{R(AB)}{S(A,B)}$ ✗ invalid b/c then what will Q take (every Relation will have atleast one attribute)

$\frac{R(ABCD)}{S(AC)} = Q(BD)$ $(x,y) \in B.D$

(x y) → Those value of <BD> which pair with Every value of S(A,C) in R

$R(\underbrace{A_1, A_2 \cdots A_n}_{A}, \underbrace{B_1 \cdots B_m}_{B}) \div S(B_1, B_2 \cdots B_m) = Q(A_1 \cdots A_n)$

$Q(\underbrace{A_1, A_2 \cdots A_n}_{B}, \underbrace{B_1 \cdots B_m}_{A})$

Those value of A which pair with every value of S in R.

ex: r_1 (a | b) r_2 (b | c)

$\frac{r_1}{r_2}$ } invalid $\frac{r_2}{r_1}$ } invalid

ex: relation R, S;
$(R \times S)/S = R$

Note Division is just Reverse/opposite of cross product.

R, S
(R×S) } { R×S/R = S] To separate S
R×S/S = R] To separate R

Will combine everything

ex: R(Kids, games) S(games) Those kids who don't play some S·game

$\pi_{kid}(R) \times S - R \rightarrow T$

all (kids, S·game) pairs

$x \in y-2$
then $x \nsubseteq z$ & $x^{\wedge} \in y$

$\Rightarrow$ what ever kid, game

$(K_1, G_1) \notin R$
K_1 NOT play G_1
&
$G_1 \in S$ ← $(K_1, G_1) \in M$

▸ $\pi_{kid}(T)$ = Those kids who don't play some S·game

▸ R/S: kids who play all Sgame

$\Rightarrow$ [all kids − kids who do n't play some Sgame]

$\pi_{kid}(R) \times S - R \leftarrow T_{(kid, game)}$ → S.game

all (Kid, S·game) pair

Does not play

$\Rightarrow \pi_{kid}(R) - [\pi_{kid}(\pi_{kid}(R) \times S - R)]$

- Healy Implementation
  - $R(A,B,C,D) \div S(A,B) = Q(C,D)$

$$\pi_{C,D}(R) - \left[\pi_{C,D}\left[S \times \pi_{C,D}(R) - R\right]\right]$$

another way

- $R(A,B) / S(B)$

$$\bigcap_{\forall b_i \in B} \pi_A(\sigma_{B=b_i}(R))$$

for every b_i we are find those A value that is related to that b_i

ex. R(Kid, Game) & S(Game)

Assume R, S are non-Empty

$|R| = n$, $|S| = m$

what is min. & max. no. of tuple in R/S

$S = \phi$ } $|S| = 0$

$R \times S = \phi$

Anything $\times S = \phi$

$|R \times S| = |R| \times |S|$ (0)

$|R \times S| = 0$

- $R(AB), S(B)$ $|S| = 0$, $|R| = n \neq 0$

$|R/S| = \pi_A(R) =$ # unique A Rows in R

{min = 1, max = n}

R(K,G) S(G)

$|R/G|$: min = 0 (when no kid play all S game)

$max = \left[\frac{|R|}{|S|}\right]$

↳ Healy Implementation : R(AB) S(B) → ϕ

$R/S = Q_{(A)} \Rightarrow$

$\Rightarrow \pi_A(R) - \left[\pi_A\left[\pi_B(R) \times S - R\right]\right]$

$\pi_B(R) \times S = \phi$, $\pi_A(R) - \phi = \phi$, $\pi_A(R) - \phi = \phi$

$\Rightarrow \pi_A(R)$

- $R(AB), S(B)$ $|S| =$ anything $|R| = 0$

$|R/S| = 0$

$R/S = \phi$

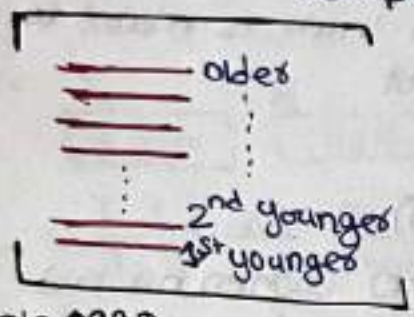
$R/S = \pi_A(R) - \phi$

$R/S = \phi$

× (product) — Equally costly operation in terms of Comparisons (time)

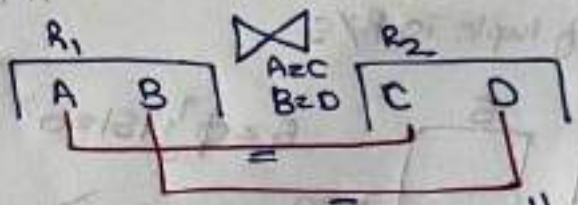
Join → better in term of storage (space)

note
↳ union, intersection join, any type of join, when done on non-key attribute might lead wrong ans. But not Always

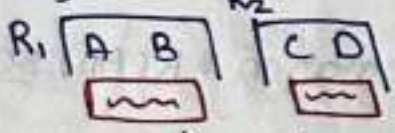


gate 1998
ex: $R_1(A,B)$ & $R_2(C,D)$ &

$R_1 \bowtie_{A=C \wedge B=D} R_2$

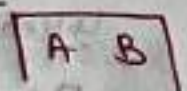


SO,

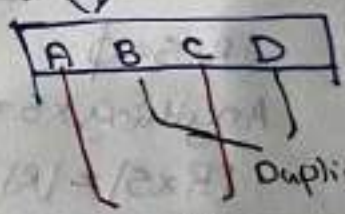


Should be Same

⇒ O/P



⇒ RNS



Stud(Sid gen) Course(Cid Instr.) enroll(Sid Cid)

1) Retrive sid enrolled every Course taught by Korth.

$$\pi_{Sid,Cid}(Enroll) \,/\, \pi_{Cid}(\sigma_{Instr.=Korth}(course))$$

$$\pi_{Sid}(E) - \pi_{Sid}\left(\left(\pi_{Sid}(E) \times \pi_{Cid}\left(\sigma_{Instr.=Korth}(C)\right)\right) - \pi_{Sid,Cid}(E)\right)$$

↑ not enrolled every Korth course.

2) Retrive Cid's enrolled every female student

$$\pi_{Cid,Sid}(Enroll) \,/\, \pi_{Sid}\left(\sigma_{gen=femal}(Stud)\right)$$

$$\pi_{Cid}(Enroll) - \pi_{Cid}\left(\left(\pi_{Cid}(Enroll) \times \pi_{Sid}\left(\sigma_{Instr.=female}(Stud)\right)\right) - \pi_{Cid,Sid}(Enroll)\right)$$

Rel R with x set of Attr.
44 n distinct tuple

Rel S with y set of Attri
44 m distinct tuple

To use R/S required condition

$X \supset Y$ (strictly)
X should be a superset of Y

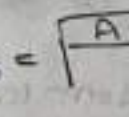
R/S resulted Attribute set

$X - Y$

cardinality of R/S: {0 to $\lfloor \frac{n}{m} \rfloor$}
(min-max tuple)

$\pi_{AB}(R) \,/\, \pi_{B}(S) =$ 

$\pi_{ABC}(R) \,/\, \pi_{C}(S) =$ 

$\pi_{AB}(R) \,/\, \pi_{B}(S) =$ 

} empty

n_a { $a_1 b_1$, $a_1 b_2$, $a_2 b_2$, $a_3 b_3$, $a_3 b_1$ } m { b_1, b_2, b_3 }

$\pi_{AB}(R) / \pi_{A}(S)$ ✗ not allowed

$\pi_{nB}(R) / \pi_{BC}(S)$ ✗ not allowed

Cardinality of R/s : 0 to $\lfloor \frac{n}{m} \rfloor$; i.e $\pi_{(R)}$ / $\pi_{(S)}$ =

A
a_1
a_2

$\lfloor \frac{n}{m} \rfloor$

(min-max tuple)

Atmost n, if m=0

AB	B
$a_1 b_1$	b_1
$a_1 b_2$	b_2
$a_1 b_3$	b_3
$a_2 b_1$	
$a_2 b_2$	
$a_2 b_3$	

n, m

model 7

(*) Atleast too, only too, Atmost:

Qus] Enroll(sid cid)

Retrieve sid enrolled for atleast 2 course.

Enroll (sid cid) × Enroll(sid cid)

S_1	C_1	S_1	C_1
S_1	C_2	S_2	C_2
S_1	C_3	S_1	C_3
S_2	C_1	S_2	C_1
S_2	C_2	S_2	C_2
S_3	C_1	S_3	C_1

Kenneth Rose D.M

P - Q ≡ P ∧ ¬Q

π_{Sid} (Enroll $\bowtie_{S_1 = S \wedge C_1 \neq C}$ ρ(Enroll))

i.e C(x): x is student in class

S(x): x stud. Math.

(a) some student in class studied math $\exists_x (C(x) \wedge S(x))$

(b) All student in class studied math $\forall_x (C(x) \rightarrow S(x))$

(c) atleast (2) student in class studied math

$\exists_x \exists_y (C(x) \wedge S(x) \wedge C(y) \wedge S(y) \wedge x \neq y)$

[Some x student in class studied math] ∧ [Some y student in class studied math]

Retrieve sid enrolled atleast 3 course.

$$\pi_{T_1.sid}\left(\sigma_{\substack{T_1.sid = T_2.sid \wedge \\ T_2.sid = T_3.sid \wedge \\ T_1.cid \neq T_2.cid \wedge \\ T_2.cid \neq T_3.cid \wedge \\ T_1.cid \neq T_3.cid}}\left(\rho(T_1, Enroll) \times \rho(T_2, Enroll) \times \rho(T_3, Enroll)\right)\right)$$

① Retrieve sid enrolled **only one** course.

{Sid Enroll some course} − {Sid's enroll atleast 2 course}

$$\pi_{sid}(Enroll) - \pi_{sid}\left(Enroll \bowtie_{\substack{S.id = S \\ \wedge\, C.id \neq C}} \rho(Enroll)\right)$$

$\exists_x(C(x) \wedge S(x)) \wedge$
$\neg \exists_x \exists_y (C(x) \wedge S(x) \wedge C(y) \wedge S(y) \wedge x \neq y)$

② Retrieve sid enrolled only two course.

{Sid's enrolled for atleast two course} − {Sid's enrolled for atleast three Course}

Enroll(sid cid) Stud(sid Sname)

③ Retrieve Sid's enrolled for atmost one course.

{All Students} − {Sid's enrolled for atleast two courses}

$\exists_x(C(x) \wedge$
$\neg \exists_x \exists_y (C(x) \wedge S(x) \wedge C(y) \wedge S(y) \wedge x \neq y)$

$$\pi_{sid}(Stud) - \pi_{sid}\left(Enroll \bowtie_{\substack{S.id = S \\ \wedge\, C.id \neq C}} \rho(Enroll)\right)$$

Retrieve Sid's enrolled for atmost two courses.

{ all sids of Student } − { sids enrolled atleast 3 course }

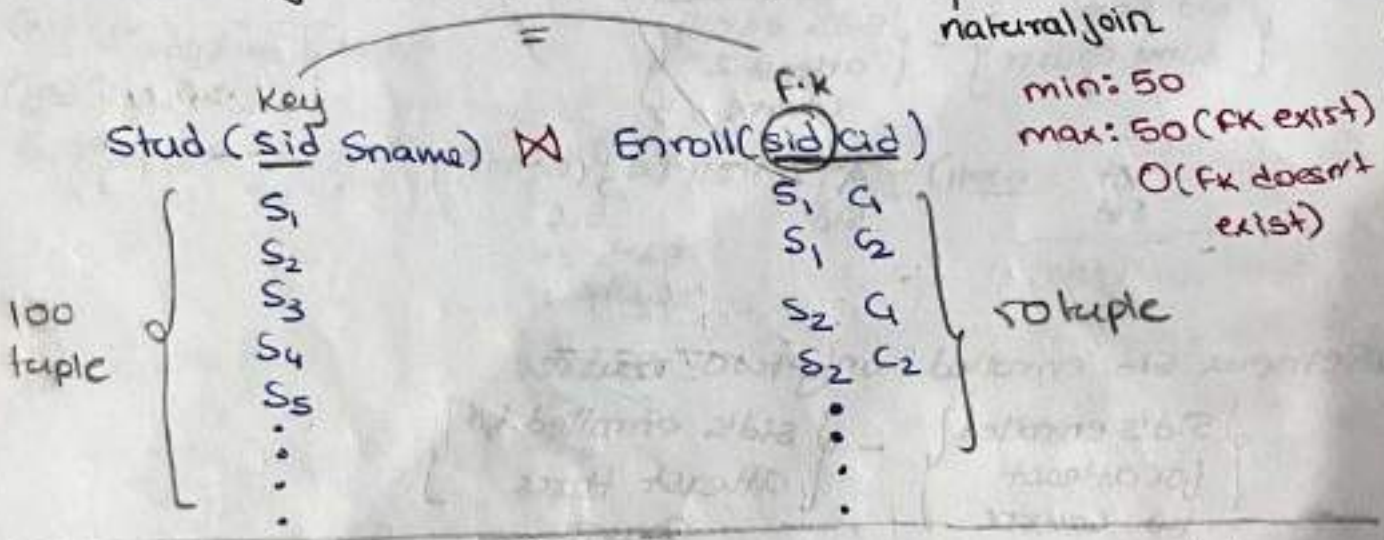
model 8

Q] Consider given relational schemas

Stud (sid sname) with sid (P.K) & 100 tuple

Enroll (sid cid) with sidcid (P.K) & 50 tuple

How many (max,min) in result of Stud ⋈ Enroll ?
(natural join)



Relation Algebra operation	Cardinality [min - max tuple]	Commutative
1. $\pi_A(R)$	atmost n tuple	
2. $\sigma_c(R)$	0 to n	
3. R × S	n * m	
4. R ⋈ S, R ⋈$_c$ S	0 to n*m	
5. R ⟕ S	n to n*m	
6. R ⟖ S	m to n*m	
7. R ⟗ S	max(n,m) to n*m	
8. R ∪ S	max(n,m) to n+m	
9. R ∩ S	0 to min(n,m)	
10. R − S	0 to n	
11. R / S	0 to $\frac{n}{m}$ if m>0	

• SQL

ex: why does SQL allow table to have "duplicate" tuple?

- elimination of these is expensive operation
- user may want to see duplicate tuple in result.

SQL keywords: case-insensitive

WHERE ≠ where ≠ WHERE ≠ WHERe

note: In SQL Also:

If (AB) is P.K then

① A,B (any of them) can Not be null

② NO Duplicate tuple in (AB) (so in Relation)

ex: R(A,B) & S(C,D)

```
Select A,D
from R,S
where (A>10) ∧ (C ≤ 5)
```

$\equiv \pi_{A,D}(\sigma_C(R \times S))$

A	D
1	2
1	2

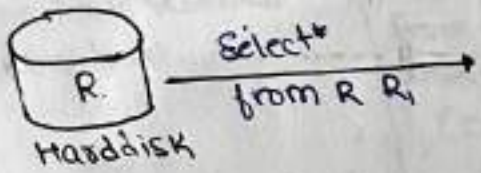
A	B	C	D
1	3	5	2
1	3	6	2

```
Select A,C
from R,S
where ....
```

$\equiv \pi_{A,C}(\sigma_{cond'}(R \times S))$

SQL Data Retrieval Query:

Original Database Doesn't Change



Actual story of how it work

$Q_1 \equiv Q_2 \equiv Q_3$

SQL Query (Q_1)

↓ conversion

Rel. Algo. Query (Q_2)

↓ optimized.

Rel. Algo. (Q_3)

↓

O/P

• Renaming/Alias: {As} optional

① Readable, Compact Query

② mandatory, only when Same table taken more than two time

Conceptual Evaluation → tell's the O/p of Query

VS

Actual Evaluation → provides Efficiency

(this is not how actual evaluation happens)

In SQL:

union
Except / minus
Intersect

- Result in set
- standard
- oracle

In SQL, Same as Rel. Alg. { Result never Contain Duplicate }

① union compatible Table
② Result in a SET of tuple

ex. R union S → invalid SQL Query

(Select * From R) union (Select * From S) — Valid

---- Intersect ----

---- set Difference (Except) ----

UNION ALL
except ALL
intersect All

- not Remove Duplicate
- Result in multiset
- union all : frequency union
- intersect all : frequency intersection

ex:

R
A
a_1
a_2
a_2
a_3

intersect All

S
A
a_1
a_2
a_4
a_5

$1a_1 \cap 1a_1 = 1a_1$
$2a_2 \cap 1a_2 = 1a_2$

a_1	a_2

ex:

R
a
b
b
c

intersect All

S
b
b
b
c

b	b	c

union
Except
Intersect

① 1st remove duplicate from I/p &
② then remove duplicate from o/p.
only for union
(Except, Intersect will not have any Duplicates)

when done on non-key attribute, might lead to wrong ans. (But not always)

note

- By Default SQL Allows Duplicate (SQL I/p & o/p are multisets)
But for the 3 keyword (unions, Intersect, Except)
- I/p as multiset & o/p as set.

- Select Clause can use Arithmatic Expression $(x, -, +, /)$

ex. (find)
Select all sid of sailors who have reserved a red or a green boat

① Select sid
From R,B
where (R.bid = B.bid) ∧
(Color = Red or
Color = green)

≡ equivalent (duplicate tuple will be there)

② Select Distinct sid
. . . . . .
. . . . . .
. . . . . .
||| duplicate will be removed

③
Select sid
from R,B
where (R.Bid = B.bid)
∧ (color = Red)
Union All
color = green

$Q_1 \equiv Q_3$
$Q_2 \equiv Q_4$
In term of Duplicates
(Assuming no null values)

④
Union

ex: Sid of Sailors who have reserved both a red & a green boat.

wrong ✗
Select sid
from R,B
where (R.bid = B.bid) and
(color = Red and
color = Green)
→ o/p empty relation

Q_{Red} ---→ Sname
Intersect
Q_{Green} ---→ Sname
wrong → o/p BabuRao

Color
Red/Green] can't be in single entry

sid
S_1 — BabuRao — Red, not green
S_2 — BabuRao — Green, not red
] no other baburao

Prblm
non-key ---→ Sname
∩
non-key ---→ Sname
Should be taken care

Select sid
From R,B
where :::
(color = Red) and
(color <> Green)
→ give those sid who have booked Some Red boat
(may or may not have booked green boat)

- union, intersect, except on non-key attribute them Take care!!

not equal (<>)
wrong ↳ ≠

R

Sid	bid
a	b_1
a	b_2
b	b_3
c	b_2
d	b_4

B

bid	color
b_1	Red
b_2	green
b_3	Red
b_4	yellow

```
Select Sid
 from R,B
where (R.bid = B.bid) ^
      (color = Red) ^ (color <> Green)
```

not for "Sid who booked Red but not green"

applied single row at a time

O/p: Those Sids who booked Red (may or may not green)

Correct solution

```
Select R1.Sid
From Boat B1, Reserve R1,
     Boat B2, Reserve R2
where R1.sid = R2.sid
 and R1.bid = B1.bid
 and R2.bid = B2.bid
 and (B1.color = 'red' AND
      B2.color = 'green')
```

ex: all sailors sid who have not reserved a red boat

R

Sid	bid
a	b_1
a	b_2
b	b_1
c	b_2
d	b_3

B

bit	color
b_1	Red
b_2	green
b_3	yellow

wrong Query.

```
Select Sid
 From R,B
 where (R.bid = B.bid)
   and (color <> Red)
```

Applies on single tuple at time

Those Sailors sid who Reserved at least one boat of Non-Red color.

O/p a, c, d wrong

Regardless of whether they reserved Red or not.

All sid's in Reserves − Sid's Reserved Red boats

Sid who did not Reserve Red But Booked atleast one Boat

there is blc a → Red, a → green; remove red still a will be in o/p blc of green so will directly remove

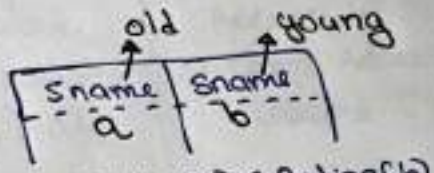
Correct Solution

```
Select S.sid
From Sailors S
EXCEPT
Select R.sid
From Boats B, Reserves R
Where R.bid = B.bid
    And B.color = 'red'
```

Sailors(sid, sname, rating, age)
Boats(bid, bname, color)
Reserves(sid, bid, day)

ex: Find all pair of Sailors in which the older sailor has a lower rating

```
Select S1.sname, S2.sname
From Sailors S1, Sailors S2
where S1.age > S2.age
    and S1.rating < S2.rating
```



Rating(a) < Rating(b)
age(a) > age(b)

- Select can also add a 'const' column (new col will be created and will give illusion of been unit)

ex: Select Salary * 1.1, 'Rs'
From Instructor

Salary * 1.1	'Rs'

---

gate 2000 ex r(w,x) & s(y,z) → can have Duplicate tuple

```
Select distinct w,x
from r,s
```

→ can not have Duplicate tuple

is guaranteed to be same as r. if either r = ϕ or |s| = 1

- Conceptual Evaluation
  1. r × s
  2. projecting w, x
  3. Duplicate Removed.

ex:

r		s	
w	x	y	z
1	2	a	b
2	3	c	d

O/P {1,2, 2,3}

if r ≠ ϕ, s = ϕ

r	
w	x
1	2

s	
y	z

then O/P is ϕ

So, s ≠ ϕ & r can have duplicate

But in O/P every tuple of r But no Duplicate

if `Select w,x from r,s` O/P {1,2, 1,2, 2,3, 2,3}

guarrantee that O/P ⊇ r

C/p contains every Tuple of r But possibly Duplication)

- Dangerous combinations
  1. non-key except nonkey
  2. nonkey intersect non-key
- not dangerous comb^n
  1. nonkey union non-key
  2. key {except, unim, intersect} key

Sub-language of SQL: DDL, DML, DCL, TCL

SQL – Structured Query Language

Data Definition lang (DDL)	Data Manipulation lang (DML)	Data control lang (DCL)
• Used defining modify definition or structure of DB table also used to modify Integrity constraints ⇒ CRETE TABLE... ⇒ DROP TABLE... ⇒ ALTER TABLE... [To add/remove Attribute] ⇒ To define Primary key, UNIQUE, NOTNULL, FOREIGN KEY etc...	used to modify data record of the table. Also used to Access data from DB table ⇒ Insert INTO... ⇒ DELETE FROM... ⇒ UPDATE ... SET ⇒ SELECT * (are mandatory) From table where condition (where is optional) ** data access	used to grant data Access or revoke data Access to users [security] ⇒ GRANT ⇒ REVOKE

Traction Control Lang: [TCL]
used for Concurrency Control to avoid inconsistency
⇒ COMMIT
⇒ ROLL BACK
⇒ CHECK POINT

SQL VS RA Query

Eliminate Duplicate Tuple from final Result

3. SELECT DISTINCT $A_1, A_2, A_3 \cdots A_n$ → Projection op. (π)
4. From $R_1, R_2, R_3 \cdots R_m$ → Cross product ($\times$)
2. where P; → selection op. (σ)

equal

$$\pi_{A_1 \cdots A_n}\left(\sigma_P(R_1 \times R_2 \cdots \times R_m)\right)$$

note
- SQL is Declarative, not procedural
no need to give order of operation

=Same

Catalog (Sid pid) — Parts (Pid Color) ← red

Catalog	
S_1	P_1
S_1	P_2
S_2	P_2
S_3	P_3

Parts	
P_1	Red
P_2	Red
P_3	green

to remove duplicate values use → DISTINCT

Sid's supplied some red parts?

RA: π_{Sid} (catalog ⋈ $\sigma_{Color=Red}$ (parts))

SQL

```
Select S.ids
FROM Catalog, parts
where part.color
    =Red and
Catalog.pid = part.
    pid;
```

Rename can be done { From catalog AS C or part P }

SELECT DISTINCT ⟷ projection(π)

Basic SQL Clauses

6 — work on whole tuple

5 SELECT [DISTINCT] $A_1, A_2, \dots A_n$

1. FROM $R_1, R_2, R_3 \dots R_m$
2. [WHERE CONDITION] ← each record
3. [GROUP By Attribute
4. [HAVING Condition]] ← each group
7. [ORDER By Attribute [DESC]]; → By default ASC

only Conceptual execution flow:- [so that human can understand]

1. FROM : cross product(×)
2. WHERE : selection (σ)
3. *GROUP BY
4. *HAVING
5. SELECT : projection (π)
6. DISTINCT — work on final result
7. ORDER By

note

SELECT, FROM are mandatory Clauses to write SQL Query

⇒ SQL not case Sensitive

⇒ Data is case Sensitive

Aggregate funct^n → Summarize the whole table (Apply on whole table) → produce only one value

count(*): count no. of row in the table (no distinct, no attribute taken)

count(A): will produce same o/p as, if no 'null' value exist

count(Distinct *) ✗ invalid

Min(A), Max(A), Avg(A), Sum(A), Count(A) behave in the same manner (taking attribute) • ignore 'null'

Min(Distinct A), Max(Distinct A)] Same as min(A) & Max(A)

Avg(Distinct A), Sum(Distinct A), Count(Distinct A)] There will Remove Duplicate first & then apply's

ex:

A	B
1	2
3	4
1	2
1	2

Avg(Distinct A) ⇒ $\frac{1+3}{2} = 2$

Avg(A) = $\frac{1+3+1+1}{4} = 1.5$

note

▸ If we use atleast one Aggregate funct^n (in Select clause) then "NO unaggregated attribute allowed" in select clause (unless there is a group by clause)

• Count(A) ≡ count(All A) ↓ Redundant (optional)

↳ Duplicate will be there

Sum(A), Avg(A)] A: must have numeric Domain

Min(A), Max(A)] A can be numeric or string

count(A) — A can have Any Domain

ex: Select ~~sid~~, max(age) ✗ From Sailors;] wrong query

when using Aggregate you can't use other than that (un Aggregate attribute)

note Sum of empty table is null

A	B
null	null

⇓

Select A, Sum(B) From R Group by A;

o/p

A	Sum(B)
null	null

▸ SQL 92 Rule having clause can not have having clause in the Absence of Group by clause

i.e Group By A,B having ~~~ ; Aggregate attribute (Any attribute) ↳ unaggregate attribute must be subset of group By Clause

ex: find the name & age of the oldest sailor.

```
Select S.name, Max(S.age)
From Sailors S
```
} wrong

```
Select sname
from Sailors
where age = max(age)
```
} wrong

where → Applies on one tuple at a time

Independently (Applied on individual tuple)

Sailors

sname	age
a	50
B	40
b	80

→ max(age) = 50

age = 50

So, all tuple will be in o/p

So, 50 will be in o/p

Correct query

```
Select sname
from Sailors
where age = (Select max(age)
             from Sailors)
```

max(age)
50

not a value but a Relation

it will do

40 = | max(age) |
|---|
| 10 |

SQL will convert into value b/c it knows that it will give an single tuple (b/c of Aggregate)

then why is there valid?
b/c SQL know's that it's an Aggregate so it will return single value

Null value

50 + null = Null

unknown value/ not applicable value

Null (any thing) = Null

any arithmatic operation (+, -, ÷, ×)

R(A) — Empty Relation

count(A) = 0

sum(A) = null
AVG(A) = null
min(A) = null
max(A) = null

by definition SQL Define

(2) comparsion with null is always "unknow"

another boolean value

null ≠ unknown

a truth value

many Descriptions

- not exist
- missing
- unknow
- not applicable

3-boolean values → True, false, unknown

new system: T, F, U

note: where clause passes a tuple only when "True"

Tuple fails when 'false or unknown'

R	A	B
U	null	null
T	3	3
F	2	1
U	3	null

Select * from R where A − B ≠ 0

Arithmetic operator, comparison

3 − null ≠ 0

null ≠ 0 fail

So, unknown

null comparison Anything = unknown

Truth value

>, =, ≥, <, ≤, <>

① where clause:

apply to one Row at a time

Row passes iff condition True

Row fails if condition false or unknown

IS null

Select count(*)
from R
where A IS NULL

o/p: no. of null tuple

whether a tuple is null or not

IS not null

where count(*)
from R
where A IS not null

whether a tuple is not null or not

One exception → Definition of Duplicate tuple: from human eyes they look Duplicate.

R	A	B
	null	1
Duplicate	null	1
	null	null
Duplicate	null	null
not Duplicate	2	null

• Select Distinct *
from R;
8 tuple in Ans.

SQL [Standard - 92] → All standard textbooks based on SQL-92 → followed by gate exam

There will be some query which are allowed by SQL but not by SQL-92

↑ Virtual partition

GroupBy

```
Select ..... avg(marks)
From Students
Group By State
```

State should be in select

→ Allowed by all pratical DBMS s/w

Invalid Query (as per SQL-92)
SQL-92 Standard not allow

Having → filtering groups.

where ↔ having (do same thing)

where done on tuple

having done on Group

- usefullness of groupby is Aggregation funct^n
  - b/c of these we use group by
- Apart from aggregation, you can use Group by But it is almost useless without aggregation

note: Aggregation in select Clause happens only within Group (when we have groupby clause)

ex. R

A	B
1	2
1	3
5	7
1	3
6	7
null	null
6	null
null	null

⇒ select A, Sum(B) from R Group by A;

R

A	B
1	2
1	3
1	3
5	7
6	7
6	null
null	null
null	null

⇒

R

A	Sum(B)	
1	8	G_1
5	7	G_2
6	7	G_3
Null	null	G_4

- what happens w/o Aggregation

```
Select A
From R
Group by A;
```

O/P

A
1
6
null
5

note: If groupby is used then for Every Group, atmost / Exactly one tuple will be in O/P

- atmost: b/c of having clause (other tuple will be eliminated)
- Exactly: b/c of not having, having Clause

- Select clause in presence of Group By clause
  - ① aggregated attribute
  - ② unaggregated attribute

i.e Select A,B, Sum(C), count(A) Group By A,B (exactly same like these)

ex: Select A, Group By A,B; — here SQL 92 has prblm

X Invalid

Select 'X', Aggregate(...)... Group by: X — X is a attribute or set of attribute (both X must be same)

note: if there is no group By clause

↳ if the group By clause is omitted the entire table is regarded as a single group

↳ that's why Aggregation without Group by clause give single tuple in the o/p

↳ 1 Group gives 1 tuple.

[when no having clause then for every group, one tuple in o/p

so, no. of tuple in o/p = no. of group]

- group by w/o Aggregation is useless

Select A, From R, Group By A; ≡ same as ≡ Select Distinct A from R;

ex: no. of tuple in o/p = no. of Group (after where clause Removes tuples)

R(A,B) S(B,C)

Select A,C, count(B)
From R,S ①
where A<>S.B ②
Group By A,C; ③

R

A	B
1	2
1	2
1	3
null	null
null	5
7	2

X S

B	C
1	2
1	4
4	3
null	null
2	1

group by A, R.B, S.B, C

result show only true tuple

result tuple

A	C	count(B)
1	3	3
1	1	3
7	2	1
7	3	1
7	4	1
7	1	1

⇒ R X S

A	R.B	S.B	C
1	2	4	3
1	2	null	null
1	2	2	1
1	2	4	3
1	2	2	1
1	2	3	3
1	3	4	2
1	3	2	1
1	null	2	4
1	null	3	3
7	2	2	1
7	2	4	3

1 o null unknown (fail condition) — fail condition so will not be in tuple

One more tuple

7221

- Find the age of the youngest sailor for each rating level above 7.

```
Select Rating, min(age)
From Sailors
Group by Rating
having Rating>7
```

✓ correct query

```
Select rating, min(age)
From Sailors
where rating>7
Group by rating;
```

✓ correct query

Forget About Query
Think "can a sailor (with rating ≤7) affect our calculation?" → NO
So, can use where clause to eliminate them.

ex: Suppose that we want to count the total no. of employees whose Salaries > 40,000, in each department, but only for departments where more than 5 employees

these will effect these count ↑

```
Select Dno, Count(*)
From employee
where Salary>40000
Group by Dno.
having Count(*)>5
```

→ [Eliminates those who have Salary ≤ 40000] → we have only Rich (calling them Rich people)

→ we have only Rich People groupwise

↳ those group where Rich people are >5

---

null < 0 : null ✗
null < 0 : unknown ✓
↳ a truth value
Comparison

Null + 50 : unknown ✗
Null + 50 : null ✓
↳ Arithm operat^

e

A	B
null	null

```
Select A, count(B), count(*)
From R
Group By A;
```

O/P

A	count(B)	count(*)
null	0	1

Keyword

Aggregate fun → either used in select or having clause
Supported by SQL

note
Having compute for all value of an Attribute

1. COUNT()
2. SUM()
3. AVG()
4. MIN()
5. MAX()

it should be not null or nonnull value

R

A	B
20	5
-	10
40	-
20	-
80	20
60	40
-	5

#f record

Select Count(*)
FROM R;

Count(*)
7

#f non null value of A

ex: Select SUM(A) A_1,
SUM(DISTINCT A) A_2,
AVG(A) A_3, AVG(DISTINCT A) A_4
MIN(A) A_5, MAX(A) A_6
FROM R;

Select Count(*) A_1, Count(A) A_2,
Count(DISTINCT A) A_3
FROM R;

A_1	A_2	A_3
7	5	4

A_1	A_2	A_3	A_4	A_5	A_6
220	200	44	50	20	80

AVG(A) = SUM(A) / COUNT(A)
⇒ 220/5 = 44

AVG(DISTINCT A)
⇒ Sum(DISTINCT A) / Count(DISTINCT A)
⇒ 200/4 = 50

(generally not used in where clause)
It is not invalid but aggregate value is to produce single value

1. Select AVG(A), B
FROM R;

aggregate value

AVG(A)	B
44	5
	10
	-
	⋮

X Invalid

Selection of aggregate value with other attribute is not allowed

2. Select *
FROM R
where A = MAX(A); Each record

20 = max(20) ✓
null = max() X
40 = max(40) ✓
20 = max((20) ✓

O/P

A	B
20	5
40	-
20	-
80	20
60	40

GROUP By Clause — Used to group records of the relation based on value of group by attributes. (order doesn't matter)

R

A	B	C
a_1	b_1	80
-	b_2	40
a_2	b_5	90
-	b_2	60
a_3	b_6	60
a_1	b_1	90
a_2	b_5	85
a_3	b_7	40
a_1	b_2	-

Groupby(A) → [4 group's]

GROUP BY(A,B) → both are same (6 groups)

A	B	C
a_1	b_1	80
a_1	b_1	90
a_1	b_2	-
a_2	b_5	90
a_2	b_5	85
-	b_2	40
-	b_2	60
a_3	b_6	90
a_3	b_7	40

∴ there are same restriction for selection

```
SELECT x., AVG(y)
  FROM R
  GROUP By x;
```

- If groupBy clause used in query then
  - ↳ attribute of groupBy clause are allowed to select in select clause
  - ↳ not allowed to select any other attribute which doesn't belong to group By
  - ↳ Allowed to used Aggregate funct^n in select clause & aggregate funct^n compute aggregate for each group

i.e
```
Select A
From R
group by(A);
```

A
a_1
a_2
null
a_3

# of record in result = 3

# of groups

note

only for Aggregate funct^n we eliminate null but for all other funct^n we count them

① Select A, AVG(C) o/p
FROM R (for each group)
group by A;

A	AVG(C)
a_1	85
a_2	87.5
—	50
a_3	65

② Select Avg(C)
From R
group by A;

Avg(C)
85
87.5
50
65

③ Select B
From
group by A; ✗ Invalid

④ Select A,B
from R
group by A; ✗ Invalid

⑤ Select A,B
From B
group by (A, B);

A	B
a_1	b_1
a_1	b_2
a_2	b_5
—	b_2
a_3	b_6
a_3	b_7

⑥ Select (X) must follow Group by (X) Aggregation(...)...
Allowed by SQL-92
Set of attribute

HAVING Clause

- used to select group based on having clause condition
- having clause condition test for each group.
- not allowed to apply on direct record
- where clause condition applied for each record
- having clause condition must be over Aggregate fun^n & some special defined funct^n like some, every.. etc. But not allowed direct attribute comparision.

① Select A
From R
group by A
having C>75; } Invalid usage

② Select A
from R
group by A
having AVG(C)>75;

A
a_1
a_2

③ select A
from R
group by A
having some(C)>75;

A
a_1
a_2
a_3

④ Select A
From R
group by A
having every(C)>75

A
a_2

```
Select A
from R
having AVG(C) > 75;
```

O/p ⇒ 5 tuple

If having clause used without group By
Then having clause condition tested for each record
[default : each record is 1 group]

ORDER BY used to sort record ASC/DESC based on specified attributes

```
Select A,B
From R
order by A;
```
default: ASC

A	B	C
a_1	b_1	90
a_1	b_1	80
a_1	b_2	-
a_2	b_5	90
a_2	b_5	85
a_3	b_9	90
a_3	b_7	40
-	b_2	40
-	b_2	60

ASC

if order by A, B ;
1st 2nd

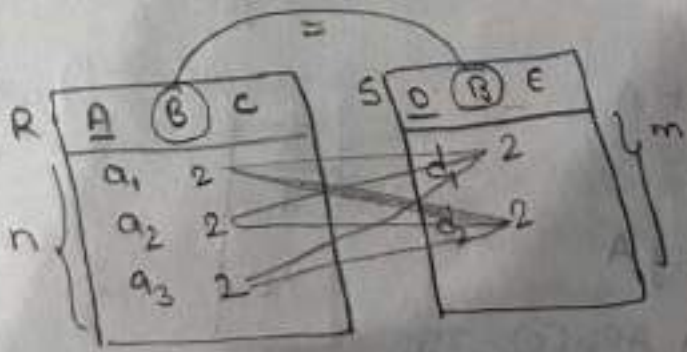
model 8

assume every attribute of R is not null attribute

Ques] R(A B C) S(B D E)

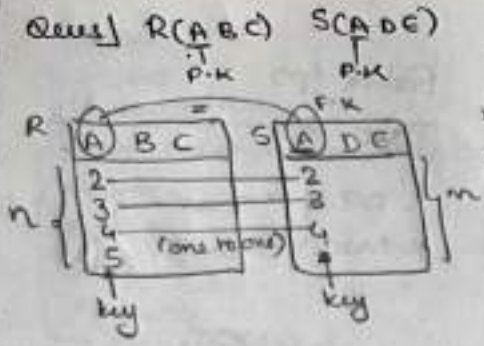
R⋈S result
max: n tuple
min: { n tuple if F.K exist
0 if no F.K

Ques] R(A B C) S(D B E)
P.K P.K



R⋈S
max: m*n tuple
min : 0 tuple

Ques] R(A B C) S(A D E)



RDBS result
max: min(m,n)
min: min(m,n) with F.K
0 if no F.K

Note
- Foreign keys is the deciding authority

Comparision with NULL:-

- NULL is not zero & not empty
- As per RDBMS guideline null set of ASCII value assign by DBMS such that value of 2 null may not be ~~zero~~ equal & comparsion with null the result is unknow which is niether True or false

R	A	B	
✓	a_1	90	T
×	a_2	-	U
×	a_3	40	F
×	a_4	-	U

Select *
FROM R
where B > 50

1 tuple in result

NULL > 50 unknow (U)
NULL ≤ 50 unknow (U)

note
where clause discard record if result of the condition is false or unknow

IS/IS NOT Access based on Null value.

Emp

eid	ppno
e_1	P_5
e_2	-
e_3	P_1
e_4	-
e_5	P_2

Retrieve eid's who have no password?

eid
e_2
e_4

Select eid
from emp
where ppno IS NULL;

we don't use (ppno = null) "=" b/c every compared with null will give unknow & o/p would be zero

SQL use 3-VL truth table :- {T, F, U} (RDBMS Guideline)

OR	T	F	U
T	T	T	T
F	T	F	U
U	T	U	U

AND	T	F	U
T	T	F	U
F	F	F	F
U	U	F	U

False = 0, unknown = 1/2
True = 1

x OR y = max(x,y)
x AND y = min(x,y)

NOT	
T	F
F	T
U	U

Q1] R

	A	B	C
①	20	20	-
②	20	-	10
③	5	25	10
④	25	15	-
⑤	-	25	5

How many tuple in Result?

```
Select *
From R
where A>5 And (B>C or C>5);
```

O/p = 1

① T and (U OR U) = U
② T and (U OR T) = T
③ F and (T OR T) = F
④ T and (T OR U) = U
⑤ U and (T OR F) = U

If Q_1 executed before Q_2, what is Avg of Q_2?

gate Q] R

A	B
a_1	10
a_2	20
a_3	30
a_4	null

Q_1 : update R set B : B+5

A	B
a_1	15
a_2	25
a_3	35
a_4	null

Q_2 : select Avg(B) from R;

O/p : 25

null + 5 = null
null − 5 = null
null * 5 = null
null ÷ 5 = null

note

Arithmetic operation with null & aggregate funct^n discard the null value

- Nested Queries

Select
From
where
Group By
having

main Query
(outer Query)

when can you put Subquery into Select clause
only if q gives single Tuple with single column
Subquery(q) (select ...)
use single aggreate funct$ w/o Group by

- Two type of sub Queries
① Simple subquery
Subquery w/o co-relation with outer query
② Correlated Subquery
Subquery using attribute(s) of outer query.
- bunct$ used for nested Query
- IN, NOT IN (Set membership keyword)

ex: S- Tuple
R — Subquery → Relation which is Result of a Subquery

S in Sailors ✗ Invalid

S IN (R) = True iff Tuple S is in R
S NOT IN (R) = True iff tuple S is not in R
R must be a Subquery
(can't be Relation directly from Database)

IN (Subquery)
✗ IN (Relation name) Invalid

unique R
IN R
NOT IN R
Exists R
All R
Any R
→ Invalid if Relation name used directly
must be Subquery (query)

ex: find the name of sailors who have reserved some boat.

```
Select Sname
From Sailors
where Sid IN Reserves
```

not Compatible

Sid bid Day

✗ wrong

single Column value

```
Select Sname
from Sailor
where Sid IN (select Sid
              from Reserves)
```

Note

- even if there is a compatible, we can't use Relation name
- an attribute belongs to closest surrouding Relation which has this attribute

ex: find the name of sailors who have not reserved boat 103

- Select S.Sname
  from Sailors S
  where S.Sid NOT IN (Select Rsid
  from Reserves R
  where R.bid = 103)

Correct query

wrong query ✗

- Select S.sname
  From Sailors S
  where S.Sid IN (Select R.Sid
  from Reserve R
  where
  R.bid<>103)

it need only atleast one bid which is not 103 but it can be the case were bid is 103 & also not 103 for a sid

ex: find the name of sailors who have reserved a red boad

```
Select S.Sname
From Sailors S
where S.Sid IN (Select R.Sid
                From ReserveR
                where R.bid IN (Select B.bid
                                From Boats B
                                where B.color='red'))
```

- can have multiply level of Subquery.

- which SQL Query is more efficient
  - can not determine by looking at SQL Query → doesn't tell us order of evaluation
  - Can determine Efficiency by looking at Rel. Alg. Query
    - ↳ procedural

ex: Find the name of sailors who have not reserved a red boat ⇒

```
Select ...
... not IN( ...
... IN( ... )
```

ex: Select ... IN(...) ... NOT IN(...)

there query is telling "Sid Reserving some non-Red Boat".

ex Break the o/p of Query

```
Select S.sname
From Sailor S
where S.sid NOT IN(select R.sid
                   From Reserves R
                   where R.bid NOT IN(select B.bid
                                      From Boat B
                                      where B.color='red'))
```

- S.sname: name of those Sailors who did not Reserve any non-Red boat
- R.sid: Sid who Reserved Some non-Red Boat
- B.bid: Bid of Red Boats

o/p
① either not reserved any boat at all
(OR)
② whatever Boat they Reserved was Red

ex query ... = (subquery ...)

✗ Invalid as subquery can have multiple tuple but it should be single

ex query ... = (subquery ...) using aggregate

where will execute tuple by tuple

✓ Valid

ex: Select *
from R
where A > (Select count(*)
from S
where R.B < S.B)

↓ tuple by tuple

A	B
1	2
3	4
5	6

B	C	D
2	4	6
4	6	8
4	7	9

for (1,2)

R.B < S.B
2   2, ~~4~~, ~~4~~] count(*) ①

A → 1>1 ✗ false

for (3,4)

R.B < S.B
4   ~~2~~, 4, 4] count(*) ②

3>2 pass (A)

for (5,6)

R.B < S.B
6   ~~2~~, ~~4~~, ~~4~~] 0

5>0 pass

final o/p:

A	B
3	4
5	6

- Exists / not Exists

ex
```
Select *
From R
where Exists(Select *
             fromS
             where B<C)
```

R

A	B
1	2
3	4
5	6
7	8

S

C	D
2	10
4	12
6	14
8	16

For (1,2) will be in o/p if subquery's true

B=2 2<2 ✗, 2<4 ✓, 2<6 ✓, 2<8 ✓ } True

For (3,4)
B=4 & 4>2 ✗, 4>4 ✗, 4>6 ✗, 4>8 ✗ } True

For (7,8)
B=8 & 8>2 ✗, 8>4 ✗, 8>6 ✗, 8>8 ✗ } false

will not be present in o/p

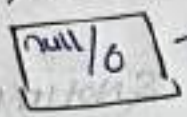
For (5,6)
B=6 & 6>2 ✗, 6>4 ✗, 6>6 ✗, 6>8 ✗ } True

Final o/p =

A	B
1	2
3	4
5	6

R → Empty

Sum(a)/Count(*)



→ non-empty (∃ a tuple)

- there is a 1 tuple which is null (or) 0

ex: Select *
From R
where Exist(select count(*)
From S
where B<C)

o/p

A	B
1	2
3	4
5	6
7	8

→ will be present when inner query is True

ex: Select *
From R
where not Exist(select *
From S
where B<C)

o/p

A	B
7	8

will be present when inner query is false

replace with count(*) then

o/p

A	B

- unique Keyword

Unique (Subquery) = True iff no Duplicate in the result of Subquery

- not unique keword.

not unique (Subquery) = True iff Some Duplicate in the Result of Subquery.

- Any/All Keyword
* Some ≡ ANY.

S ≤ Any (Subquery)

True iff.
S ≤ Some value in the Result of subquery

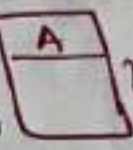
- S ≤ Any → return false

Yempty

S ≤ ALL (Subquery)

- True iff S ≤ foreveny value in the Result of Subquery

- S ≤ ALL → return true



Yempty

any Comparsion operator.

note

① S In R ≡ S = Any R & S Not In R ≡ S <> ALL R

② Exist (aggregate functn)

always satisfy

Always non-empty.

Find the name of Sailors who have reserved all boats

```
Select sname
from Sailor
where not Exists [(Select B.bid
                   from Boats B)
                  Except
                  (select R.bid
                   From Reserves
                   where S.Sid=R.sid)]
```

⇒ write a correlated w/o Except

```
Select sname
from sailor
where not exist
  [ Those (bids) of Boats
                  not
                  Exists
    [(S.sid=R.sid)
     and
     (R.bid=B.bid)] ]
```

these now doesn't exist then that boat is not booked not reserved)

Idea: Those sids of Sailors for which not Exist a Bid such that not exists Record in Reserves table with S_i, b in same Record

↓ Proper query

```
Select sname
from Sailors
where not Exist [Select B.bid
                 from Boat B
                 where not
                 Exists [Select *
                         From Reserve R
                         where R.bid=B.bid
                           and R.sid=Si.sid]]
```

Nested Queries

conceptual (for us(human) to understand) : In Reality SQL optimizers can execute in some other Efficient way.
Execution flow

① non-Corelated nested Query:-

- Inner query is independent of outer query

SELECT Attrs
② FROM TABLE
WHERE Cond.
Top
group By Attr.
HAVING Cond.
ORDER By Attr.

Select ...
From ...
where ...
① Bottom

i.e Select eid
from emp
where age > (Select Avg(age)
from emp
where gen=female)

Bottom

- execution flow bottom to top
- Inner query compute only once

② Corelated nested Query:-

- Inner query not Independent
- where clause of inner query uses attribute of outer query table
- where, HAVING clause of outer query allowed for corelated nested Query

i.e select T_1.eid
Top② from emp T_1
Top③ where T_1.age > (
Top②
select Avg(T_2.age)
from Emp T_2
where T_2.gen=female
and T_2.dno=T_1.dno);

related to Outer query

- if corelation in where clause. Inner query recompute for each record of outer query
- If corelation in having Clause inner query recompute for each group of outer query

Q] R

A
80
60
40
90
50
30

S

B
20
50
80
40
90
60

How many tuple in result of Query? O/P: 3

```
Select *
From R
where (Select count(*)
       From S
       where R.A>S.B)<3;
```

} Value of A which should be greater for only 2 value

ex: emp(eid, sal)

(a) Retrieve eid's who get highest salary?

$$\pi_{eid}(emp) - \pi_{eid}\left(Emp \bowtie_{S_1.S} \rho_{S}(Emp)\right)$$

(1)
```
Select eid
From emp e1
where sal > All(select sal
                From emp e2
                where e1.sal ≠ e2.sal)
```

(2)
```
Select eid
From emp
EXCEPT
SELECT T1.eid
From Emp T1, Emp T2
where T1.sal < T2.sal
```

(b) SQL Query using Max() fun:-

```
Select eid
From Emp
    where sal = (select max(sal) As msal
                 from emp);
```

OR

```
Select eid
From Emp, (select max(sal) As msal
           From Emp) Temp
where sal=msal;
```

emp

eid	sal
e_1	60
e_2	90
e_3	60
e_4	70
e_5	90

Temp

msal
90

(c) Retrieve eid's of 2nd highest salary

RA: $\rho(Temp, \pi_{eid,sid}(Emp \bowtie_{Sal<S} \rho(Emp)))$

↑ All emp's except who get maxsal

$$\pi_{eid}(Temp) - \pi_{ed}\left(Temp \bowtie_{Sal<S} \rho(Temp)\right)$$

SQL:
```
with Temp(eid, sal) As
    (Select distinct T1.eid, T1.sal
     From Emp T1, Emp T2
     where T1.sal < T2.sal)
Select eid
from Temp
Except
Select T1.eid
from Temp T1, Temp T2
where T1.sal < T2.sal;
```

- with clause avoid recomputation, subquery result of with clause can be used many time for query computation

(b) SQL Query for 2nd highest using max():-

```
Select eid          ← Eid of 2nd max
From Emp
where sal = (select max(sal)        ← 2nd max
             from Emp
             where sal < (select max(sal)   ← 1st max
                          from Emp));
```

(90)


```
Select *
From Emp
where sal < (select sal
             from emp
             where dno=5);
```

Error

i.e sal < {60, 70} set of value

So we can't compare

If its single scalar value than only comparision is done

Functions used for nested Q.

- IN / NOT IN
- ANY
- ALL

} preferred for non corelated queries

- EXISTS / NOT EXISTS

} preferred is co-related Queries.

↓

If inner query result more than one record to compare with outer query where / having clause condition required to use funct^n

(binary operator) used for membership test

[Equi join ($\bowtie_=$) Queries can be expressed using IN function]

IN funct^n (X)

X IN | Y |
| 2 |
| 4 |

return true if x ∈ y
[x value is (= Some) value of y set]

$\bowtie_=$

{ IN funct^n used equijoin ($\bowtie_=$) Queries }

Ques1 | R(AB) S(BC)

=some

① $\pi_{AB}(R \bowtie S)$

② Select distinct R.A, R.B
from R, S
where R.B = S.B;

③ Select A,B
from R
where R.B IN (Select S.B
from S);

= some (atleast)

Ques2 | R(AB) S(CD)

$\pi_{A,B}(R \bowtie_{R.A=S.C \wedge R.B=S.B} S)$: Select A,B
from R
where (A,B) IN (Select C,D
from S);

{IN} ⟷ {= some}

{NOT IN} ⟷ {<> every}

Ques3 | Emp(eid ename supID)

Retrieve ename of supervisors?

$\pi_{ename}(Emp \bowtie_{eid=E,N,S} \rho_S(Emp))$

Emp

eid	ename	supID
e_1	A	–
e_2	B	e_1
e_3	C	e_2
e_4	D	e_2

Select ename
from Emp
where eid IN (Select supID
from Emp);

= some

(eid = null) or
(eid = e_1) or
(eid = e_2)

supID
null
e_1
e_2
e_2

IN funct^n behaves like "OR" connectivity with = Comparision of Inner query values

⇓

{=some}

ex: ename's of the emp's who are not supervisors?

RA: $\pi_{ename}(\pi_{eid,ename}(Emp) - \pi_{eid,ename}(Emp \bowtie_{eid=S} \rho_{S,N,S}(Emp)))$

SQL:
```
Select ename
FROM emp
where eid NOT IN (select supID
                  from emp
                  );
```
<> every

{ eid<>null and eid<>e_1 and eid <>e_2 }

where supID is not null

supID
~~null~~
e_1
e_2
e_2

we should eliminate the null value in the table

if Inner value consist null value than NOT IN funct^n will return true (NOT IN result either false or unknow

Note

NOT IN funct^n behaves like ... AND connectivity with Comparision of Inner Query Val's

IN funct^n the "null" will not create any problem

NOT IN funct^n will create problem if null value exist

ANY/ALL funct^n preceded by Compansion operator {<, <=, >, >=, =, <>}

Any fun can be used for conditional join ($\bowtie_c$) queries

X (> ANY) Y

True if value of x is > some value of y set

X (> ALL) Y

TRUE: If x value > every value of y set

{ANY} is {OR connectivity}

{ALL} is {AND connectivity}

{IN} ⟺ {= ANY}

{NOT IN} ⟺ {<> ALL}

- check if the Comparsion attribute is containing null value or not
- for any funct^n it is not an issue but for ALL funct^n it's an issue

(Unary)
EXISTS funct^{n} :
[Used to test result of inner query empty or non-empty]
behave as [same/any/atleast one]

EXISTS (Inner Query)
↓
True if result of inner query is not empty.
[Atleast one record in result of Inner Query]

NOT EXISTS (Inner Query)
Return TRUE if result of Inner Q is empty

Ques) Emp(eid sal dno)
Retrieve eid's whose salary less than any emp salary of dep 5.

RA: $\pi_{eid}(Emp \bowtie_{Sal<S, D=5} \rho_{(E,S,D)}(Emp))$

SQL:
(I) Select eid
From Emp
where sal<ANY(select sal
From Emp
where Dno=5);

(II) Select eid
(3) From Emp T_1
(1) where EXISTS (select *
From Emp T_2
(2) where T_2.dno=5
& T_1.sal< T_2.sal);

(empty)

(II) Select distinct T_1.eid
From Emp T_1, Emp T_2
where T_1.sal< T_2.sal
& T_2.dno=5

emp

eid	sal	dno
e_1	60	4
e_2	65	5
e_3	70	4
e_4	75	5
e_5	80	3

emp

eid	sal	dno
e_1	60	4
e_2	65	5
e_3	70	4
e_4	75	5
e_5	80	3

Ques] Emp(eid sal dno)

Retrieve eid's whose salary less than every emp salary of dept 5.

RA:

$\pi_{eid}(Emp) - \pi_{eid}(emp \bowtie_{sal \geq S \wedge D=5} \rho_{(S,D)}(Emp))$

SQL:

```
Select eid
From Emp
Except
Select distinct T1.eid
From Emp T1, Emp T2
where T1.sal >= T2.sal
   and T2.dno = 5;
```

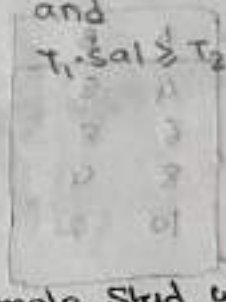
(ii)

```
Select eid
From emp
where sal < ALL (select sal
                 From Emp
                 where dno=5);
```

(iv)

```
Select eid
From emp
where EXISTS (Select *
              From Emp T2
              where T2.dno=5
              and T1.sal >= T2.sal);
```

eid's sal >= Some emp sal of dept 5



- Check emptiness of subquery
- EXISTS: Some/any/atleast one

$\bowtie_C$

- NOT EXISTS: every/All

H.W Stud(sid, gen, age)

1. sid's of female stud whose age more than some male student age

RA:

$\pi_{sid}(Stud \bowtie_{gen=female \wedge G=male \wedge age > A} \rho_{(Stud)_{S,G,A}}(Stud))$

(i)

SQL

```
Select Stud.sid
FROM Stud
where age > ANY (select S.sal
                 FROM Stud AS S
                 where Stud.gen = female
                 and S.gen = male);
```

2. sid's of female stud whose age more than every male stud age

R.A

$\pi_{sid}(Stud) - \pi_{sid}(Stud \bowtie_{gen=female \wedge G=male \wedge age < A} \rho_{(Stud)_{S,G,A}}(Stud))$

SQL

```
Select Stud.sid
FROM Stud
where age > ALL (Select S.sal
                 FROM Stud AS S
                 where Stud.gen=female
                 and S.gen=male);
```

(2) Select Stud.sid
From Stud, Stud AS S,
Where Stud.gen=female
and S.gen=male
and Stud.age > S.age

(2) Select sid
From Stud AS T1
where T1.gen=female
and NOT EXISTS(
Select *
From Stud T2
where T2.gen=male
and T1.age ≤ T2.age)

(3) Select sid
From Stud where Stud.gen=female
and EXISTS (select sal
FROM Stud as S1
where Stud.age>
S1.age

and S1.gen=male);

Que] R

A	B
4	5
6	8
8	4
10	2

S

C	D
5	4
4	7
7	8
8	9

How many tuple in result of given SQL Queries

$\pi_{AB}(R \bowtie_{R.B>S.C} (\sigma_{D>10}(S)))$ empty

(a) Select *
From R
where B > ANY (select C
From S
where D>10);

false

B > ANY	C
empty ϕ	

O/P 0

(c) select A
from R
where EXIST(select count(*)
From S
where S.D>10
and R.B>S.C);

O/p : 0

i.e

count(*)
0

not empty

(b) Select *
From R
where B > All(select C
From S
where D>10);

True

O/P 4

$\pi_{(R)} - \pi_{AB}(R \bowtie_{R.B \not> S.C} (\sigma_{D>10}(S)))$
AB AB empty

empty

$x=\{\} \,||\, \text{empty}$
$\exists x\,(x>10)$: false
$\forall x\,(x>10)$: True

R(A...) S(B...) ① Retrieve A value of R there are more than every 'B' value of S

R÷S: $\pi_A(R) - \pi_A(R \bowtie_{R.A \le S.B} S)$

② Select A
from R
where NOT EXISTS(
select *
from S
where R.A<=S.B);

③ SQL: Select
From R
Except
Select distinct R.A
From R,S
where R.A <=S.B)

③ Select A
from R
where A> All(select B
from S);

(ii) Retrieve A value of R thos are more than some B value of S

$\bowtie_C$

R·A: $\pi_A(R \bowtie_{R.A>S.B} S)$

① SQL: select distinct R.A
from R,S
where R.A> S.B;

② Select A
from R
where Exists (select *
from S
where R.A>S.B);

③ Select A
FROM R
WHERE A> Any(select B
From S);

NOTE

x operator: ANY

NOTE

x op ALL

- count(*), count(A) funcn never result empty

R

A	B
a_1	-
a_2	-

① Select count(*) from R;

count(*)
2

② Select count(B) From R;

count(B)
0

③ Select sum(B) from R;

④ Select min(B) from R;

sum(B)	min(B)

} empty

⑤ Select B from R

B
null
null

} not empty

ex: Emp(eid dno. sal) Retrieve eid who get max salary for each dept.

R.A : $\pi_{eid}(emp) - \pi_{eid}(Emp \underset{sal<S \wedge dno.=D}{\bowtie} \rho_{E,D,S}(emp))$

SQL:

```
Select eid
From Emp T1
where NOT EXISTS (Select *
                  From Emp T2
                  where T1.sal < T2.sal
                    and T1.dno. = T2.Dno);
```

→ if these empty then it will return true

ii) By using group by:

```
Select eid
From Emp, (Select dno. , max(sal) msal
           from emp.
           Group by dno.) temp
where Emp.dno. = temp.dno.
  and sal = msal;
```

OR

```
Select eid
FROM Emp
where (dno, sal) IN
  (select dno, max(sal)
   From emp
   group by dno.);
```

Emp(eid	dno	sal)	⨯	Temp(dno	maxsal)
✗ e_1	2	90		2	95
✓ e_2	2	95		3	90
✗ e_3	2	40		4	60
✗ e_4	3	60			
✓ e_5	3	90			
✓ e_6	4	60			

Assignment

1) Enroll (sid cid fee)

Retrieve sids who paid min fee for each course

$$\pi_{sid}\left(\pi_{(Enroll)} - \pi_{(Enroll \bowtie_{\substack{cid=c \wedge fee \geq f}} \rho_{(C(sid,c,f))})}\right)$$

2) family (parent, child, Child DOB)

A	B	1980
A	C	1982
B	D	2010
B	E	2012

Retrieve Youngest child of each parent

→ max child DOB

Ques| Enroll(Sid cid) Retrieve Sids enrolled only one course

R.A : $\pi_{sid}(Enroll) - \pi_{Sid}(Enroll \bowtie_{Sid=S \wedge cid \neq C} \rho_{S,C}(Enroll))$

SQL: Select sid
From Enroll T_1
where NOT EXISTS(Select *
From Enroll T_2
where $T_1.sid = T_2.sid$
and $T_1.cid <> T_2.C$)

ii) using Group by & having clause:-

Enroll(sid cid)

S_1 C_1
S_1 C_2
S_1 C_3
S_2 C_1
S_2 C_2
S_3 C_1
S_4 C_2

Select sid
from Enroll
Group by sid
having count(*)=1;
aggregate fun^{n}

⇓

Select sid
from (Select sid, count(*)C
from Enroll Group by sid)
where C=1

Ques| works (eid, pid) project (pid, pname)

P_1	DB
P_2	DB
P_3	NW

Retrieve eids who works for only 2 DB project

ex: what is result of give query? Emp(eid ename, sal gen dno)

```
Select dno
1. From emp
2. where gen=female
3 Group by dno.
4. having Avg(sal) > (Select Avg(sal)
                      from emp
                      where gen=male)
```

Avg sal of all male emp

Avg sal of female emp of each dept

Avg sal. of female emp of dept more than Avg sal. of all male emp of company.

ii)
```
Select dno.
From emp T1
where gen=female
Group by dno.
having Avg(sal) > (Select Avg(sal)
                   from emp T2
                   where T2.gen=male and
                   T1.gen=T2.dno);
```

(1 Top) (3 Top) (2) Top

Avg sal male emp of same dept

Avg sal of female emp of each dept

Avg sal. of female emp of dept more than Avg. sale of male of the same dept.

SQL Query equalent for DIVISIONS of RA:

Sid's enrolled every course taught by Korth

•Korth

R.A : Enroll(sid, cid) Course(cid, Instructor)

$$\pi_{sid,cid}(Enroll) \;/\; \pi_{cid}(\sigma_{Inst.=Korth}(Course))$$

SQL: Co-related Query :-

Enroll(sid cid) T_1

sid	cid
S_1	C_1
S_1	C_2
S_1	C_3
S_1	C_4
S_2	C_1
S_2	C_2
S_2	C_5
C_3	C_4

Course(cid Inst.)

cid	Inst.
C_1	K
C_2	K
C_3	K
C_4	N
C_5	U

$\{C_1, C_2, C_3\}$

$-$

Enroll(sid cid) T_2

sid	cid
S_1	C_1
S_1	C_2
S_1	C_3
S_1	C_4
S_2	C_1
S_2	C_2
S_2	C_5
C_3	C_4

$= \phi$ empty

Retrieve T_1.sid of enroll(T_1) if {all cid of Korth} − {cid's of enroll(C_2') enrolled by T_1.sid}

$\{C_1, C_2, C_3\} - \{C_1, C_2, C_3, C_4\} = \phi$ (empty)

$S_1 S_2$

Select distinct T_1.sid

(1) From enroll T_1

(3) where NOT EXISTS (

(2)
```
Select cid
from course
where Inst. = Korth
Except
Select cid
from enroll T2
where T2.sid = T1.sid
```
);

$S_1 = \{C_1C_2C_3\} - \{C_1C_2C_3C_4\} \cdot \{\phi\}$

$S_2 = \{C_1C_2C_3\} - \{C_1C_2C_1\} \cdot \{C_3\}$

$S_3 = \{C_1C_2C_3\} - \{C_4\} = \{C_1C_2C_3\}$

SQL: Non Corelated Query:-

$$\pi_{sid,cid}(Enroll) \;/\; \pi_{cid}(\sigma_{Inst=Korth}(course)) =$$

$$\pi_{Sid}(Enroll) - \pi_{Sid}\Big(\pi_{Sid}(Enroll) \times \pi_{cid}(\sigma_{Inst=Korth}(course)) - \pi_{Sid,cid}(Enroll)\Big)$$

$$\pi_{Sid,Course.cid}\left(\sigma_{Inst=Korth}(Enroll \times course)\right)$$

[Sid's enrolled proper subset of all Korth course]
(Sid's not enrolled every Korth course].

```
Select Sid
From Enroll
Except
Select Sid
From( Select Sid, Course.cid
      From Enroll, Course
      where Inst.=Korth
      Except
      Select Sid Cid
      From Enroll);
```

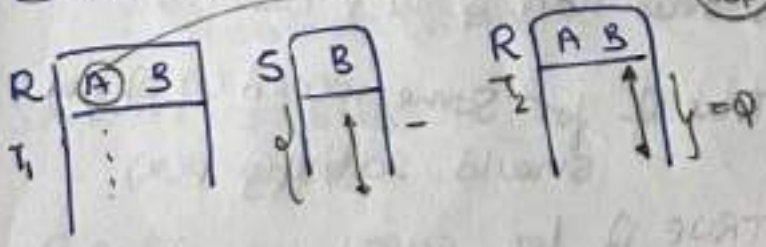
ex: Retrieve all A values such that for every 'B' value there must tuple (a,b) in R.

R.A: $\pi_{AB}(R) \mid \pi_{B}(S)$

```
Select distinct A
From R T1
where NOT EXISTS(
      Select B
      from S
      Except
      Select B
      from R T2
      where T2.A = T1.A));
```

(1) top
(3) top
(2) bottom

(a) co-related Q:



H.W

$$\pi_{cid,sid}(E) \;/\; \pi_{sid}(\sigma_{gen=female}(Stud)) \qquad \pi(E) - \pi\left(\left(\pi_{cid}(E) \times \pi_{sid}(\sigma_{gen=female}(Stud))\right) - \pi_{cid,sid}(E)\right)$$

Declarative Query language

Tuple Relational Calculus [TRC]

- predicate : simple statement which can be true or false
- variable :- x is variable which can be anything

- It is an non-procedural query lang. it uses first order logic & predicate calculus formulas.

Basic formula used in TRC :-

P, Q are predicate stmts

x is variable

$x \in$ Relation

$P \vee Q$: TRUE either P or Q

$P \wedge Q$: TRUE if both P & Q must be true

$\neg P$: TRUE if P is false

$P \rightarrow Q$: TRUE if P is true then Q must be true

Qualifier

- $\exists x \in$ Relation$(P(x))$: True if for some record (x) of Rel should satisfy P(x)
- $\forall x \in$ Relation$(P(x))$: TRUE if for every record(x) of Rel should satisfy P(x)

⇒ Tuple variable

① Free tuple variable

T ∈ Stud

↑ free tuple variable

T : Stud | ⟷ universe of tuple

• scope is global

② bounded tuple variable

Variable that are bounded by the quantifier [∃,∀]

(there exist, for all)

∃T₁ ∈ Stud (P(T₁))

∀ T₂ ∈ course (P(T₂))

T₁ T₂ bounded

$\exists_x (x \in stud \wedge x.age > 20) \wedge$ } not correct

$\exists_x (x \in Enroll \wedge x.fee > 5000)$

$\exists_x (x \in Stud \wedge x.age > 20 \wedge (x = y)) \wedge$ — free variable

Correct Version $\exists_x (x \in Stud \wedge x.fee > 5000 \wedge (x = y))$

① ② ③ ④ ⑤

hard to read

Format of TRC Query

$\{ T / P(T) \}$ T: tuple variable, P(T): formula over T

Retrieve set of tuples T those are satisfied P(T)

T: used for result must be free tuple variable

ex: $\{ T / T \in Stud \wedge T.age > 20 \}$

Retrieves students whose age more than 20

In R.A: $\sigma_{age>20}(Stud)$

ex; $\{T / \exists T_1 \in Stud\ (T_1.age > 20)\}$ ✗

$\{T / \exists T_1 \in Stud\ (T_1.age > 20 \wedge T = T_1.sid)\}$

↑ Set of IDs

$\pi_{sid}(\sigma_{age>20}(Stud))$

ex:

Supplier (Sid, Sname, rating)

Sid	Sname	rating
S_1	–	15
S_2	–	18
S_3	–	12
S_4	–	9

Parts (Pid, Pname, Color)

Pid	Pname	Color
P_1	–	Red
P_2	–	Red
P_3	–	green
P_4	–	green
P_5	–	blue

Catalog (Sid, Pid, cost)

Sid	Pid	cost
S_1	P_1	50
S_1	P_2	40
S_1	P_3	50
S_2	P_1	30
S_2	P_5	40
S_4	P_4	20

Parts(T) ∞
T ∈ parts
T is a tuple variable in Relation parts

for specific access T[pid] or T.Pid

i) Retrive sids whose rating more than 6 supplied some part

⋈ : ∃

① $\exists T_1 \in Supplier\ \exists T_2 \in Catalog$
$(T_1.Sid = T_2.sid \wedge T_1.rating > 10 \wedge (T = T_1.sid))$

② $\{T / \exists T_1 \in Supplier\ (T_1.rating > 10 \wedge T = T_1.sid) \wedge \exists T_2 \in Catalog\ (T = T_2.sid)\}$

R.A $\pi_{sid}(Catalog \bowtie \sigma_{rating>10}(Suppliers))$

R ⋈ S
R ⋈_C S
$\sigma_C(R \times S)$
↔ $\exists R \exists S\ (P(R,S))$

some / any / atleast one

TRC

- Rel. Alg. : $\sigma_{A=B \wedge D>5}(r)$

  ⇓

  TRC: $\{t \mid t \in r \wedge t.A = t.B \wedge t.D > 5\}$

- Rel. Alg. : $\pi_A(r)$

  ⇓

  TRC: $\{t.A \mid t \in r\}$ or $\{t.A \mid r(t)\}$

- Rel. Alg. $\pi_{AC}(r)$

  ⇓

  TRC: $\{t.A, t.C \mid t \in r\}$

r table

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

- TRC Doesn't tell us orders of attributes in Result Scheme

Result

A

A
α
β
γ ✗

$\Rightarrow \{t \mid \exists_{x \in r}(x.A = t.A)\}$

↓ another tuple variable

- who is going to tell that 't' belong to where?

b/c t is only used once with (t.A) that means $t \in A$.

A	C
α	1
β	1
β	2

if you want these result

$\Rightarrow \{t \mid \exists_{x \in r}(t.A = x.A \wedge t.C = x.C)\}$

b/c t is using A & C

So, in result it will show A & C

Original table

R

A	B
16	17
16	18
17	17

Result ⟶ logic

A	B
16	17
17	17
20	17 ✗

$\{t \mid \exists_{x \in R}(t.A = x.A \wedge t.B = x.B \wedge x.B = 17)\}$

- Rel. Alg. $\pi_A(\sigma_{B=10}(r))$

  ⇓

  TRC: $\{t.A \mid t \in r \wedge t.B = 10\}$

Result ⟶ logic

t

A
α
β ✗

$\{t \mid \exists_{x \in r}(x.B = 10 \wedge t.A = x.A)\}$

↑

O/p schema

- if you want to union then it must be union compatible

Relations in TRC

R∪S

Result

t

A	B

method 2:

$\{t / (\exists \ldots R \ldots) \vee (\exists \ldots S \ldots)\}$

Independent — have nothing to do with each other

$\{t / (\exists_{x \in R}(t.A = x.A \wedge t.B = x.B) \vee \exists_{x \in S}(t.A = x.A \wedge t.B = x.B))\}$

R

A	B
α	1
α	2
β	1

S

A	B
α	2
β	3

R∪S

A	B
α	1
α	2
β	1
β	3

method 1:

$\{t / t \in R \vee t \in S\}$

R∩S

$\{t / (\exists_{x \in R}(t.A = x.A \wedge t.B = x.B)) \wedge (\exists_{x \in S}(t.A = x.A \wedge t.B = x.B))\}$

R-S

↳ ∃same tuple in r & ¬∃same tuple in s

$\neg \exists_x \alpha(x) \equiv \forall_x \neg \alpha(x)$

$\{t / (\exists_{x \in R}(x.A = t.A \wedge x.B = t.B)) \wedge (\forall x \in S(x.A \neq t.A \vee x.B \neq t.B))\}$

$\neg \exists_x (\alpha \wedge \beta) \equiv \forall_x (\overline{\alpha \wedge \beta})$

$\equiv \forall_x (\bar{\alpha} \vee \bar{\beta})$

R×S method 1:

$\{t_1, t_2 / t_1 \in A \wedge t_2 \in B\}$

method 2:

$\{t / (\exists_{x \in A}(x.a_1 = t.a_1 \wedge x.a_2 = t.a_2) \wedge \exists_{x \in B}(t.b_1 = x.b_1)\}$

↳ $\{t / \exists_{t_1 \in R} \exists_{t_2 \in S} (t.A_1 = t_1.A_1 \wedge t.B_1 = t_1.B_2 \wedge t.A_2 = t_2.A_2 \wedge t.B_2 = t_2.B_2)\}$

R÷S method 1:

$R(A,B) \div S(B) = Q(A)$

- informal.

$\{t \mid \forall_{x \in S} \exists_{y \in R} (t.x = y.y)\}$

- formal.

$\{t \mid \forall_{x \in S} [\exists_{y \in R} (t.A = y.A \wedge x.B = y.B)]\}$

method 2:

$\{t.A \mid t \in R \wedge (\forall_{x \in S} \exists_{y \in R} (t[A] = y[A] \wedge y[B] = x[B]))\}$

R(A,B)

A	B
1	a
1	b
2	d
2	a
3	a
3	b

S(B)

B
a
b

Q

A
1
3

note: In TRC, $\pi, \sigma, \bowtie, \times$ tools of Rel Alg. are not allowed

But "if Given" then just focus on understanding the query

R Natural join S Method 1:

R(A,B) S(A,D)

Rel. Alg. $R \bowtie S \equiv \pi_{R.A,B,D}(\sigma_{R.A=S.A}(R \times S))$

⇓

TRC: $\{x, y.B \mid x \in R \wedge y \in S \wedge (x.A = y.A)\}$

Complete tuple.

logic → Result

Method 1

Method 2

Result → logic

Method 2:

$\{t \mid (\exists_{x \in R} x.(AB) = t.(AB)) \wedge (\exists_{x \in S} x.(AD) = t.(AD))\}$

Benefit of I/S is (only one free variable)

note

- $\forall_{x \in Human(x)} (mortal(x))$

$\equiv \forall_x (Human(x) \rightarrow mortal(x))$

$\equiv \forall_x (\overline{Human(x)} \vee mortal(x))$

to find max salary-amount

Salary(employee, salary-amount)

$\{t / \forall p \in s (t.sal\_am \geq p.sal\_am)\}$

or

$\{t / \forall p (p \notin s \vee p.sal\_am \leq t.sal\_am)\}$

note $\{ \text{---} / \text{---} \}$

only use free variable (x) — no bound variable

All remaining variable must be bounded (i.e except x)

- The o/p tuple variable must be free & All remaining variable must be Quantified / bounded

ex. Sailors(sid, sname, rating, age)
Boats(bid, bname, colour)
Reserves(sid, bid, day)

① name of sailor who reserved boat 103

$\{t.Sname \mid t \in Sailors \wedge \exists x \in Reserve (x.sid = t.sid \wedge x.bid = 103)\}$

② sid of sailors who have reserved a red boat.

TRC1:

$\{t.Sid \mid t \in Reserves \wedge \exists x \in Boats (t.bit = x.bid \wedge x.colour = Red)\}$

TRC2:

$\{t / \exists x \in Reserve\, \exists x \in Boat (t.sid = x.sid \wedge x.bid = y.bid \wedge y.colour = Red)\}$

ex: name of sailors who have rescued a red boat

TRC1:

$$\{t.sname \mid t \in Sailors \wedge \exists x \in Res.\ \exists y \in Boats \left(\begin{array}{l} t.sid = x.sid \wedge \\ x.bid = y.bid \wedge \\ y.color = Red \end{array}\right)\}$$

TRC2:

$$\{t \mid \exists x \in sailor\ \exists y \in Reserve\ \exists z \in Boats \left(\begin{array}{l} (t.sname = x.sname) \wedge \\ (x.sid = y.sid) \wedge \\ (y.bid = z.bid) \wedge (z.color = Red) \end{array}\right)\}$$

ex: name of sailors who have reserved atleast two boats

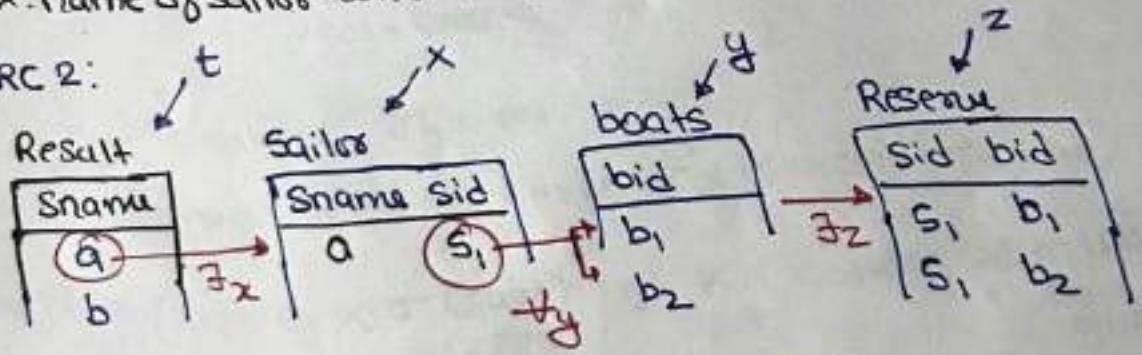
TRC1: $\{t.sname \mid t \in Sailors \wedge \exists x \in Res\ \exists y \in Res \left(\begin{array}{l} t.sid = x.sid = y.sid \\ \wedge\ x.bid \neq y.bid \end{array}\right)\}$

can also be written as $\exists x,y \in Reserve$

TRC2: $\{t \mid \exists x \in Sailor\ \exists y,z \in Reserve \left(\begin{array}{l} t.sname = x.sname \wedge \\ x.sid = y.sid = z.sid \wedge \\ y.bid \neq z.bid \end{array}\right)\}$

ex: name of sailor who have reserved all boats.

TRC 2:



$$\{t \mid \exists x \in Sailor\ ((t.Sname = x.sname) \wedge \forall y \in Boats\ \exists z \in Reserves \left(\begin{array}{l} z.sid = x.sid \wedge \\ z.bid = y.bid \end{array}\right)\}$$

(OR)

$$\{t \mid \exists x \in Sailor\ \forall y \in Boats\ \exists z \in Reserves \left(\begin{array}{l} (t.sname = x.sname) \wedge \\ (x.sid = z.sid) \wedge \\ (y.bid = z.bid) \end{array}\right)\}$$

ex: Sailor who have reserved all red boats

means: t.sid = x.sid, t.sname = x.sname, t.rating = x.rating

TRC Q:

$\{t \mid \exists x \in Sailor\ ((t = x) \wedge \forall z \in Boat((z.color = Red) \Rightarrow \exists y \in Reserve(y.sid = x.sid \wedge y.bid = z.bid)))\}$

ex: equivalent to $\forall t \in r(P(t))$?

$\neg \exists t \in r(\neg P(t))$

no Row of which doesn't satisfy P

$\Downarrow$

[every Row of r satisfies P]

$\nexists$ $\boxed{\exists t \notin r(\neg P(t))}$ → means a row not in r does not satisfy P

false

(Doesn't tell us anything whether Row of r satisfy P or not)

2) Retrieve Sid's who supplied some red part.

$\{T / \exists T_1 \in Catalog\ \exists T_2 \in Parts\ (T_1.pid = T_2.pid \wedge T_2.color = Red \wedge T = T_1.sid)\}$

R.A $\pi_{sid}(Catalog \bowtie \sigma_{col=Red}(Parts))$

$\pi_{sid}(\sigma_{age>10}(Supplier)) \cap \pi_{sid}(Catalog)$

Set operations:

1. $R \cup S = \{x / x \in R \underset{(OR)}{\vee} x \in S\}$
2. $R \cap S = \{x / x \in R \underset{(AND)}{\wedge} x \in S\}$
3. $R - S = \{x / x \in R \wedge \neg(x \in S)\}$ — But not

Ques] Retrieve Sid's who supplied some red part or some green part

Query (1)

R.A: $\pi_{Sid}(Catalog \bowtie \sigma_{col=RED \vee col=GREEN}(part))$

T.C:

$\{T / \exists T_1 \in catalog\ \exists T_2 \in part\ (T_1.pid = T_2.pid \wedge (col=RED \vee col=GREEN) \wedge T = T.Sid)\}$

Query (2)

R.A: $\pi_{Sid}(Catalog \bowtie \sigma_{col=RED}(part)) \cup \pi_{Sid}(Catalog \bowtie \sigma_{col=green}(part))$

T.C: $\{T /$ $\exists T_1 \in catalog\ \exists T_2 \in part$

$(T_1.pid = T_2.pid \wedge col=RED \wedge T = T_1.sid) \vee$

$\exists T_2 \in Catalog\ \exists T_1 \in part$

$(T_1.pid = T_2.pid \wedge Col=GREEN \wedge T = T_1.sid)\}$

Ques] Retrieve sid who have supplied some red and some green parts

Query ①

(wrong) R.A: π_{Sid} (catalog $\bowtie$ $\sigma_{col=RED \wedge col=green}$ (part))

(empty record)

C P x
C1 P1 P2 x
C2 C1 P1 x

T.C: $\{T \mid \exists T_1 \in catalog\ \exists T_2 \in parts$
$(T_1.pid = T_2.pid \wedge (T_2.col = RED \wedge T_2.col = green) \wedge T = T_1.sid)\}$

(empty record in result)

need 4 instance of the table 2 catalog 2 parts

Query ②

correct R.A: $\pi_{Sid}(catalog \bowtie \sigma_{col=red}(part)) \cap \pi_{Sid}(catalog \bowtie \sigma_{col=green}(part))$

T.C: $\{T \mid \exists T_1 catalog\ \exists T_2 part$
$(T_1.pid = T_2.pid \wedge col = RED \wedge T = T_1.sid) \wedge$
$\exists T_1 Catalog\ \exists T_2 catalog$
$(T_1.pid = T_2.pid \wedge col = green \wedge T = T_1.sid)\}$

Ques] Retrieve sids whose rating max.

Supplier (sid sname rating)

Query ① RA: $\pi_{sid}(Supplier) - \pi_{sid}(Supplier \bowtie_{rating<R} \rho_{(S,SN,R)}(Supplier))$

T.C: $\{T \mid \exists T_1 \in Supplier (T = T_1.sid)$ (belong to the Supplier set) $\wedge$
$\neg(\exists T_1 \in Supplier\ \exists T_2 \in supplier$
$(T_1.rating < T_2.rating \wedge T = T_1.sid)\}$

$P - S = P \wedge \bar{Q}$

Query ②: Sid's of supplier rating ≥ every rating of supplier

$\{T / \exists T_1 \in Supplier \ \exists T_2 \in Supplier (T_1.rating \geq T_2.rating \wedge T = T_1.sid)\}$

```
Select sid
from supplier
where rating ≥ ALL (select rating
                    from supplier);
```

Unsafe TRC Query

unsafe Query → ∞ result (not necessary, but mostly it happens)

① TRC Query result infinite record set

$\{T / \neg(T \in Supplier)\}$ unsafe TRC Query

(¬ : not operation)

T: Set of all tuple those are not belongs Supplier relation [Infinite set of tuple]

② expressive power comp :-

$$\left\{\begin{array}{l}\text{basic RA}\\ \text{Queries}\\ \text{expressive}\\ \text{power}\end{array}\right\} = \left\{\begin{array}{l}\text{Safe TRC Queries}\\ \text{expressive}\\ \text{power}\end{array}\right\} = \left\{\begin{array}{l}\text{Safe DRC}\\ \text{Queries}\\ \text{expressive}\\ \text{power}\end{array}\right\}$$

③ Basic RA Queries:-

Queries which can formulate using $\{\pi, \sigma, \times, \rho, -, \cup, \cap, \bowtie, \bowtie_c, \bowtie, \bowtie, \div, \cap\}$ operation

④ Queries failed to express using basic RA:-

- ↳ count of record / count of attr. value.
- ↳ sum & Avg of Attr. values
- ↳ Ordering of record based on Attr. value in ASC/DESC.
- ↳ To prove duplicate records in table / result.
- ↳ Insert / delete / updation records. etc

Qus] family (parent, child, childDOB) which of the query is possible to verify using R.A

(i) parent with only one child → $\pi_{parent}(family) - \pi_{parent}(family \bowtie_{parent=parent \wedge child \neq C}^{P,C,D} \rho_{P,C,D}(family))$

(ii) youngest child of each parent

(iii) All greate grand parents of "Amy"

$\pi_{child}(family) - \pi_{child}(family \bowtie_{childDOB < D}^{P,C,D} \rho_{P,C,D}(family))$

$\wedge parent = P$

family

parent	child
H	G
G	C
F	D
C	B
B	A
A	Amy

$\pi_{parent}(\sigma_{child=Amy}(T_1))$ — T_1

$\pi_{T_2.parent}(\sigma(T_1 \times T_2)) \cup$

$T_1.child = Amy$

$T_1.parent = T_2.child$

T_1 = family

T_2 = family

T_3 ⋮

$\pi_{T_3.P}(\sigma(T_1 \times T_2 \times T_3)) \cup$

$T_1.C = Amy \wedge$

$T_2.P = T_1.C \wedge$

$T_2.P = T_3.C$

⋮

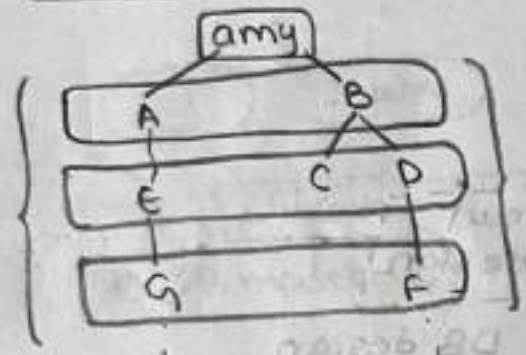
$\pi(\sigma(T_1 \times T_2 \times T_3 \cdots\cdots T_n))$

not constant length RA Query failed to express by using basic RA (length is not given)

(iv) All descendents of 'amy'

family(~~parent~~ P, ~~child~~ C)

P	C
amy	A
amy	B
B	C
B	D
A	E
D	F
E	G
H	I



$\pi_C(\sigma_{P=Amy}(family)) \cup$

$\pi_{T_2.C}(\sigma_{T_1.P=amy \wedge T_1.C=T_2.P}(T_1 \times T_2)) \cup$

$\pi_{T_3.C}(\sigma_{T_1.P=amy \wedge T_1.C=T_2.P \wedge T_2.C=T_3.P}(T_1 \times T_2 \times T_3)) \cup$

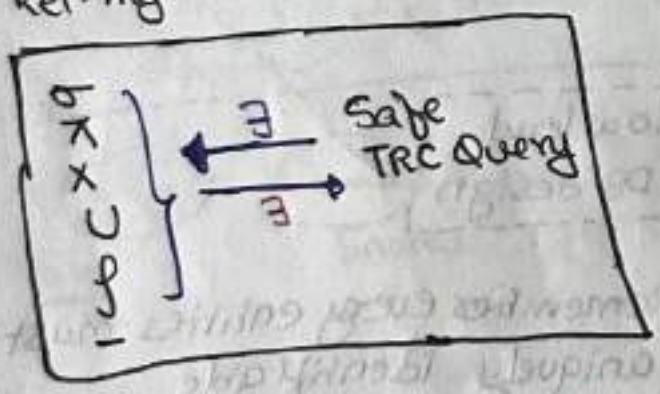
$\vdots$

$\pi_{T_n.C}(\sigma_{T_1.P=amy \wedge T_1.C=T_2.P \wedge \ldots \wedge T_{n-1}.C=T_n.P}(T_1 \times T_2 \times T_3 \cdots T_n))$

∴ not constant length RA query

failed to express using RA

Rel. Alg.



• Rel. Alg. ≡ Safe TRC Queries.

ER model (Entity-Relationship Diagram)

- ER model is used for high level database design
- Diggramatic representation of database design

1. Requirements
what type of data, operation, users, security ... etc.

2. Conceptual DB design (ER Diagram)
[Identify DB tables (Entity sets) & Relationship sets]

3. Logical DB design
[Convert ERD to RDBMS tables which satisfy 1NF]

1, 2, 3 Step are High level DB design

4. Schema Refinement
[Normalization]

5. physical DB design
[Design Indexi to reduce I/O cost]

6. User Access Right (security) & Application design } not in gate

4, 5, 6 Step are low level DB design

- DBMS
  - a s/w
- Database
  - a collection of Related Data

- Remember every entities must be uniquely identifiable

Reason: by the definition of Entity
- a Distinguishable thing.

- Don't confuse ER model with Relational model; they are different
  - Rules
  - way of Representing data

• ERmodel (VS) Relational model

① In Rel. model

Database → Relation → set of tuple

Relation: Data Structure of Relational model

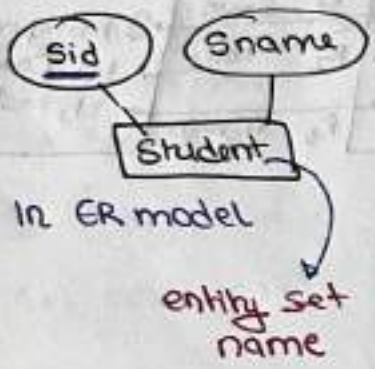
set of tuple: no two tuple can be same

Database → made of Represented as → Table (Visualized as)

But in ER model, two entities may be same (in case of weak entities)

② In Rel. model: Attribute Domain → Set of Atomic value

ER mode: many different type of attribute

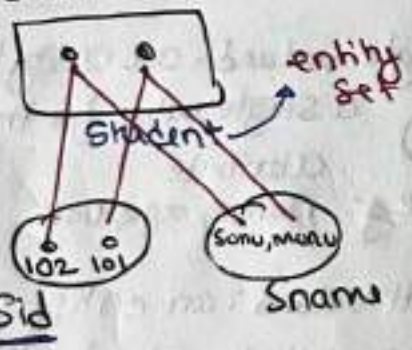


Student → relation name

Sid	Sname
101	manu
102	sonu

Table → Visualization in relational model

• Visualization in ER model



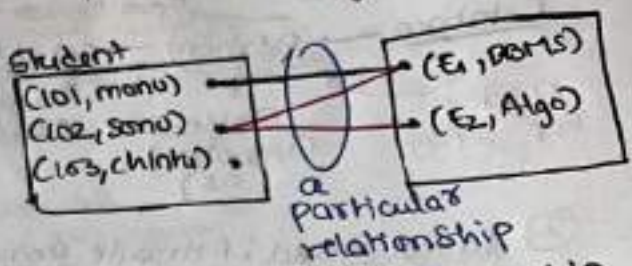
ER model	Relational model
① Entity	tuple / Row / Record
② Entity type (Entity set)	Relation table
③ PK, SK, CK	PK, SK, CK
④ attribute, Domain	attribute, Domain
⑤ cardinality of entity type	cardinality of Relation instance
no. of entities	no. of rows

Corresponding value

Relationship — Association blw entities

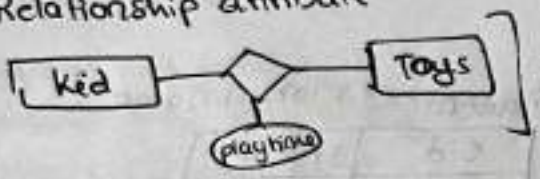
Relationship Set: Set of all similar relationship → blw same entity types

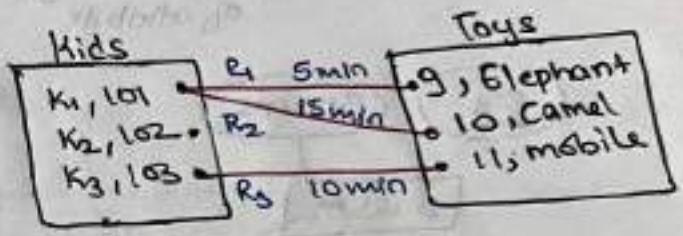
visualization of (Association blw entities)



visualization of Relationship

- Relationship attribute





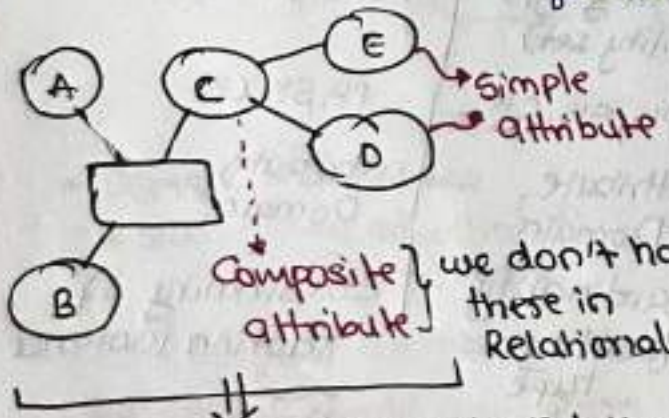
- Degree of Relationship set

Relationship is in blw how many entities is called Degree of Relationship set

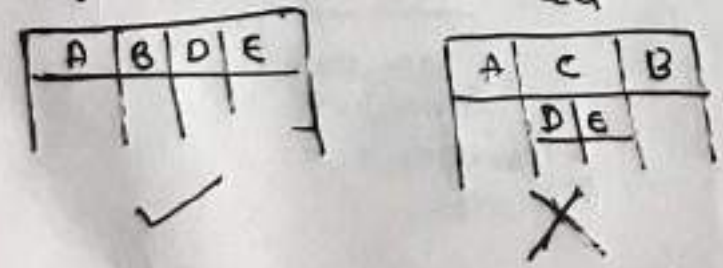
- Type of attribute in ER model.

① Simple attribute: no subpart

② Composite attribute: can be divided further.



after converting into Relational model



① Single valued: an entity has a single value for that attribute

eg: Name, age, DOB.

② Multivalued: an entity can have multiple value for that attribute

ex: phone no., hobbies.

Main Components in ERD

1. Attributes : property of entity
2. Entity sets
3. Relationship sets. association b/w entities.

1 Attribute :

Key Attribute :

Multivalued Attribute :

Derived Attribute : they are not stored while convert into Relation i.e

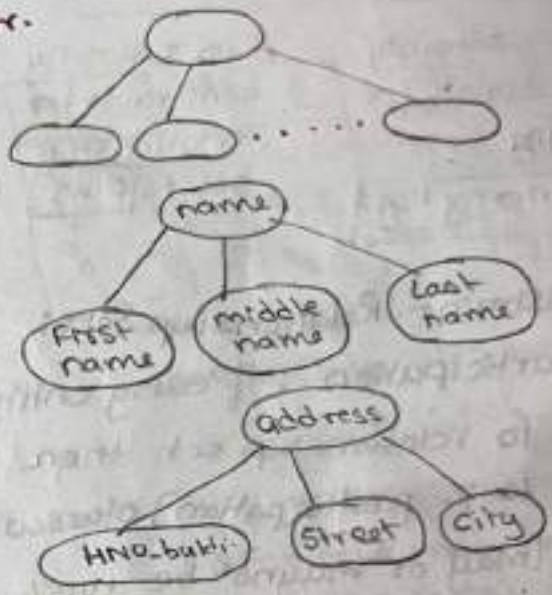
[Value of Attribute derived from other stored Attribute]

DOB age

age = (Sysdate - DOB)

Composite Attribute:

Attr. can represent as two or more Attr.



i.e

2 Entity Set :

Set of similar entities (object/tuples)

i.e

we depict in the diagram but we don't put in database, it is been derived from DOB

3 Relationship set :-

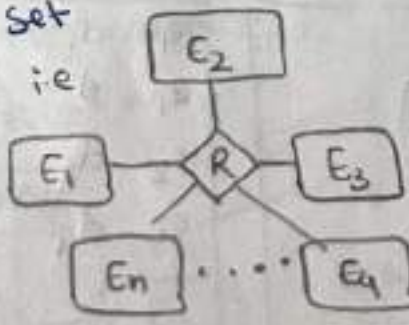
used to Related two or more entity set

(Association b/w entities)

i.e

Binary Rel. sets

i.e



n-ary Rel. set

i.e

Ternary Rel.

Stud(Sid...) Enroll(Sidcid fee) course(cid...)

F.K F.K

main Integrity Constraints for entity set is Primary Key

main Integrity constraint for relationship set is F.K's

Sidcid → fee

Descripte Attr. which describe the relation attribute set

Constraint Relationship Sets:

1. participation : if every entity of the entity set must be related to relationship set then total participation [must be 100% participation] otherwise partial participation [may or maynot be 100% participation].

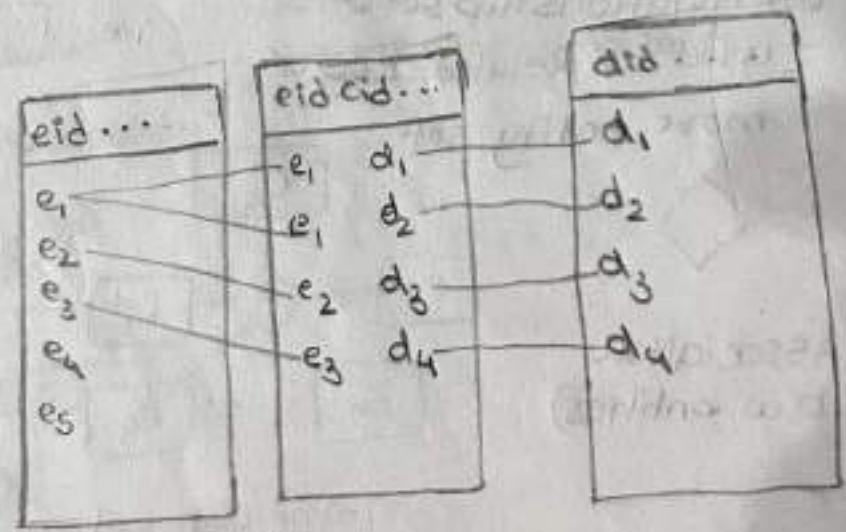
emp, dept entity set manages rel. set such that each dept there must be manager.

(note) By deafult Funct^n = Total funct^n

- every (Total) funct^n is partial funct^n But not vice versa.

↓

Every element of Domain is mapped to atmost one element of CoDomain.



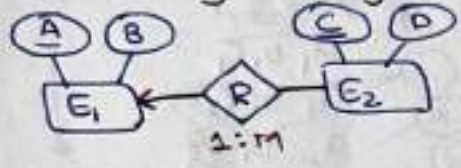
Mapping [cardinality]

one : [atmost one] ⟶
many : [0 or more] —

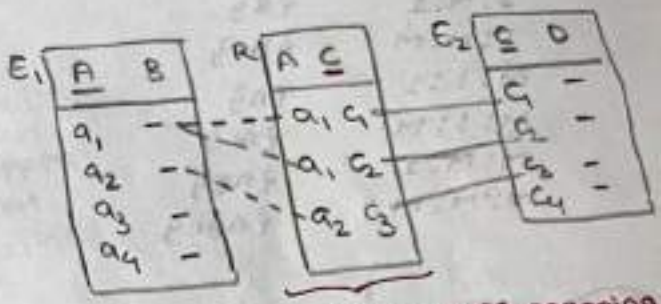
4 possible mapping for binary relationship sets

NOTE: C.K of relationshipset is design based on required mapping

1 one : many mapping



or

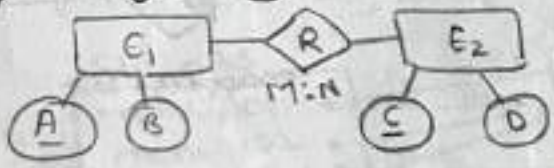


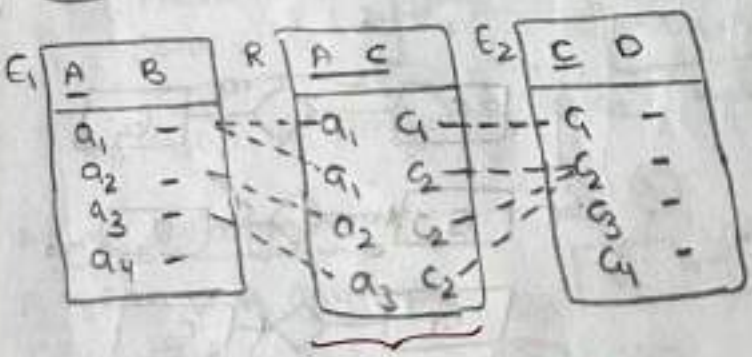
* R(A,C) for 1:M mapping
C.K of R is {C}
↳ from the right hand side

*for M:1 mapping
C.K of R is {A} ⟵
↳ from the L.H.S

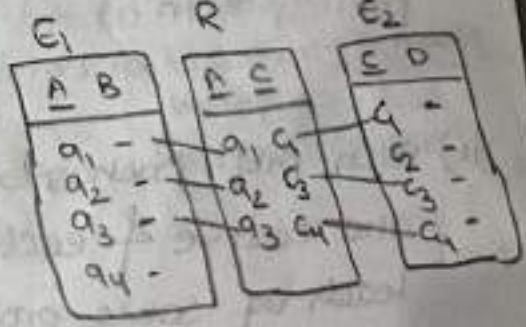
2 many : one mapping

3 many : many mapping

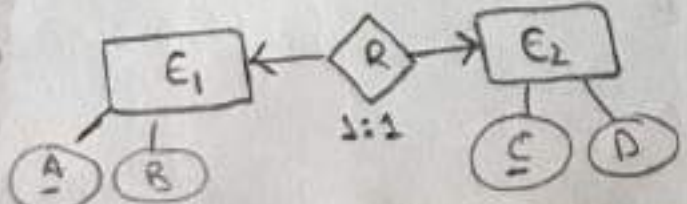




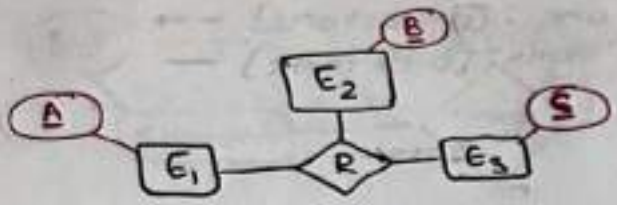
R(A,C) for M:N mapping
C.K of R is {AC}



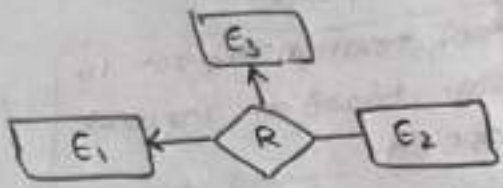
4 one : one mapping



R(A,C) for 1:1 mapping C.K's of R is {A,C}



Relation	R(ABC) C·K
1:1:1	{A, B, C}
1:1:M	{C}
1:M:1	{B}
1:M:M	{BC}
M:1:1	{A}
M:1:M	{AC}
M:M:1	{AB}
M:M:M	{ABC}



R(A B C)

$a_1\ b_1\ c_1$

$a_1\ b_2\ c_2$

$a_2\ b_1\ c_3$

Que] Draw ERD

i) Each kid likes some candy & each candy likes atmost 1 kids.

mapping 1:M

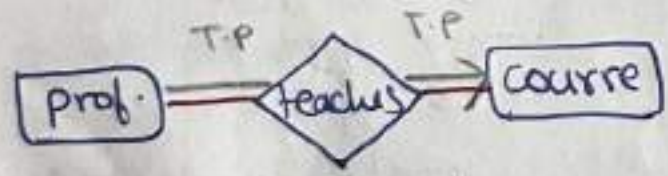
some : 1 or more

atmost : 0 or 1

participation at kid — TP

at candy — PP

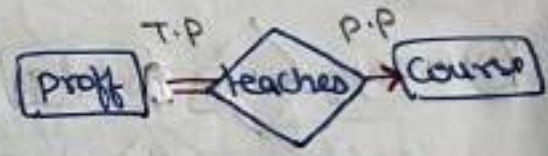
ii) each prof. teaches exactly one course & each course teach by some prof.



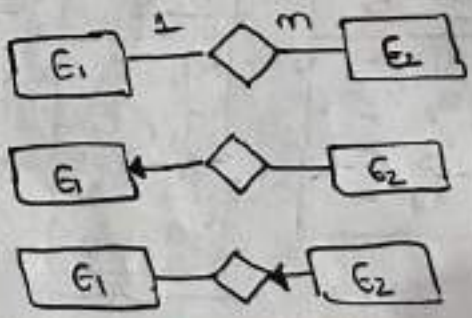
mapping M:1

many : 0 or more

ii) each professor teaches exactly one course

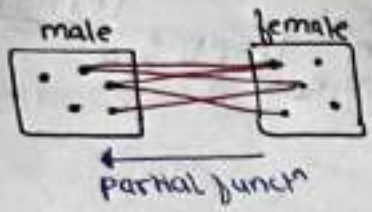


mapping M:1



One to many (from E_1 to E_2)

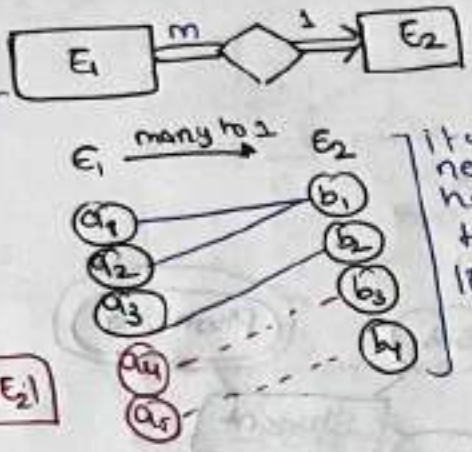
one to many Relationship



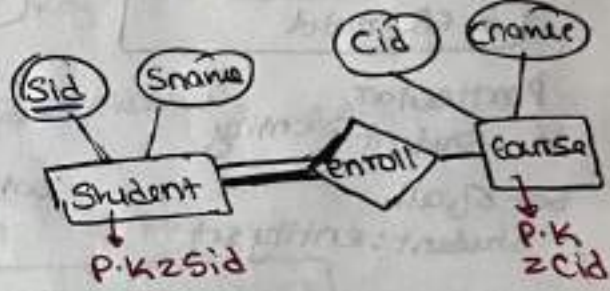
from male users to female users : not a funcn (but a relation)

from female users to male users : partial funcn

ex:



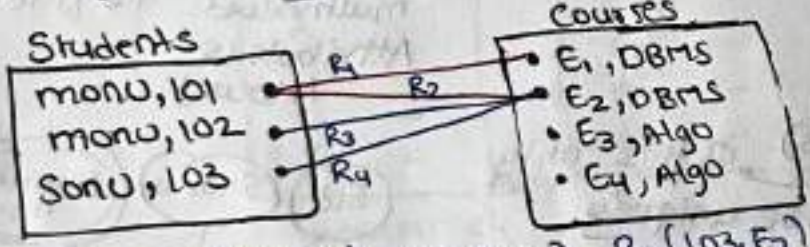
So, $|E_1| \geq |E_2|$



▸primary key / C.K for Relationship

To Desribe a relationship (Association) blw two entities

P.K of both entity sets.



$R_1(101, E_1)$ $R_2(101, E_2)$ $R_3(102, E_2)$ $R_4(103, E_2)$

Relationship set

"Enroll" : (Sid, Cid)

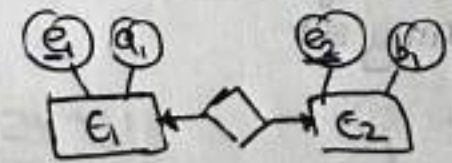
P.K of Enroll

(Sid, Cid)

is a many to many mapping blw Student course.

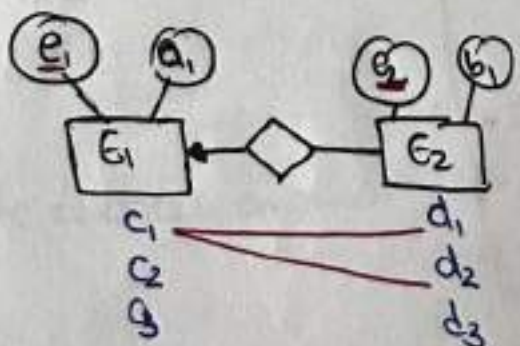
▸1-1 Relationship:

P.K can be of any table either from E_1 or E_2



▸1-M Relationship.

P.K will be of many side

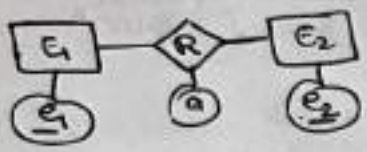


Two relationship

(c_1, d_1)

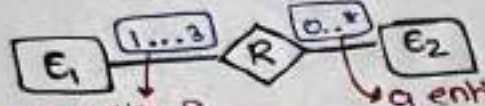
(c_1, d_2)

- Relationship with attribute (single valued)



R(e1, e2, a)

then nothing will change
1-1 : P.K of R : e1 or e2 → if C.K then (e1, e2)
1-M : P.K ---"--- : e2
M-1 : ---"--- : e1
M-N : ---"--- : e1 e2
} same for C.K



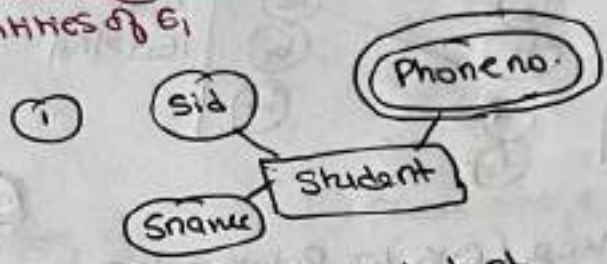
a entity of E1 can associate to "1" to atmost 3 entities of E2

a entity of E2 can associate with 0 or more entities of E1

Some misconceptions in ER model

Particular Student : entity

Set of all Student : entity set



what is the P.K of Student entity set? Sid.

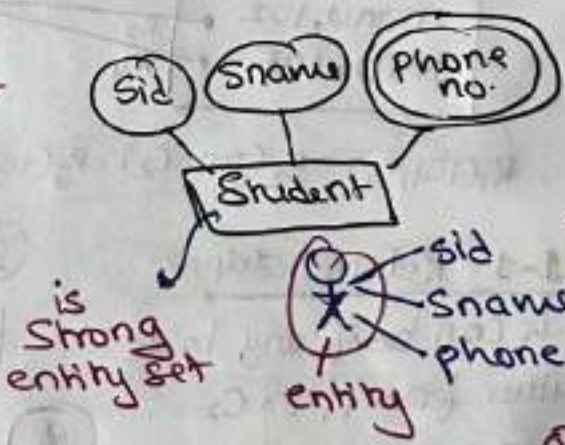
Sid	Sname	Phone no.
101	John	999, 998

} In ER model just this is fine

multivalued Attribute is Allowed

② Strong entity set (SE's)

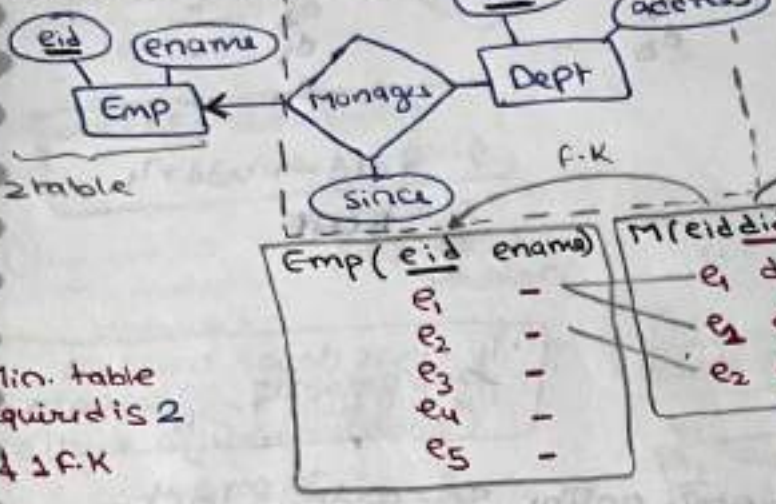
Any two entities of this entity set can be uniquely identified by some set of attribute in this same entity set



Min RDBMS table for given ERD

1 table

1 one : many mapping :-

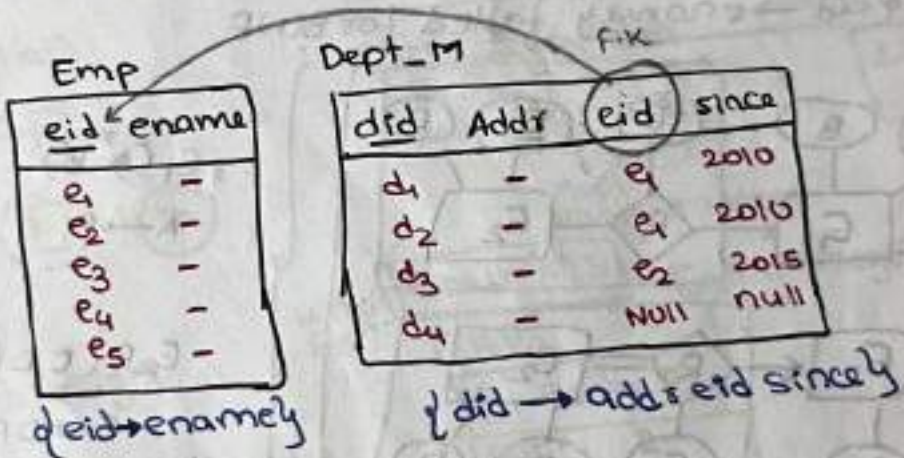


2 table

Min. table required is 2
1 1 f.K

{eid → ename} {did → eid since} {did → addr}

RDBMS design :-



{eid → ename} {did → addr eid since}

```
Create table Dept_M
( did varchar(10) primaryKey,
  Addr varchar(10)
  eid varchar(10) NOT NULL for total participation at dept end
  since date,
  foreign key(eid) references emp(eid)
);
```

Note

- if Tp at dept sid FK eid must be NOT NULL Attr.
- if pp at dept sid FK eid allowed Null's

1:M rel set merge with left side

entity set:-

emp-M (eid ename did since) dept (did addr)

e_1	A	d_1	-	d_1	-
e_1	A	d_2	-	d_2	-
e_2	B	d_3	-	d_3	-
				d_4	-

$\{eid \rightarrow ename,$
$did \rightarrow eid since\}$

C.K = did

NOT BCNF

$\{did \rightarrow addr\}$

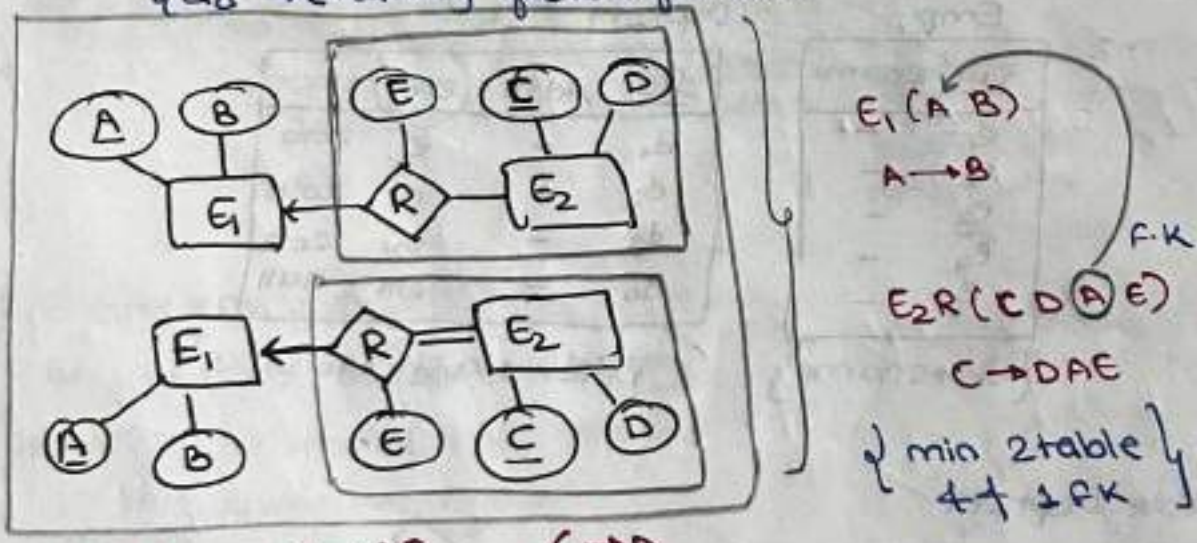
BCNF

- If C.K is design based on the FD then Mapping is not lost

disadvantages

1. partial dependances at emp entity set side is lost
2. It form data redundancy

$\{eid \rightarrow ename\}$ failed for BCNF



$E_1(A\ B)$

$A \rightarrow B$

$E_2R(C\ D\ A\ E)$

F.K

$C \rightarrow DAE$

{ min 2 table + 1 FK }

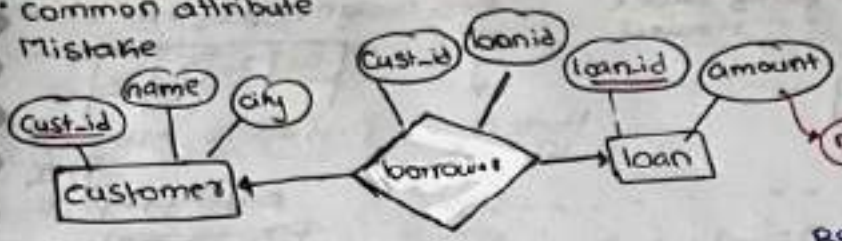
$A \rightarrow B$ $C \rightarrow AE$ $C \rightarrow D$

ER model ⟶ Relational model

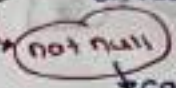
then it should be

- Good design ≥ 3NF, no null problm
- min. no. of table.

- Common attribute Mistake



- Don't include P.K attribute as descriptive attribute on relationship set



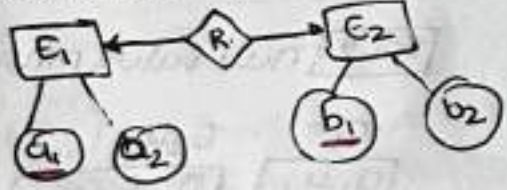
↳ can be put in Relational model (but not in ER model)

weakest Identifying Relationship
↳ create a separate relation
(Similar to Multivalued Attribute)

- mapping cardinalities (type of Relationship) & participation
  - Partial ≡ optional
  - Total ≡ mandatory

- 1-1 mapping

(i) (both sides partial)

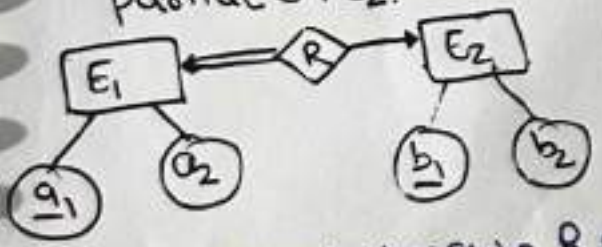


possible solution

(1) E1, R, E2 : Better design

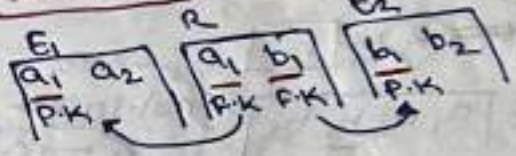
(2) E1R, E2 } problem of null value
(3) E2R, E1 } (better design only if null value allowed)

(ii) total participation on E1, partial on E2.



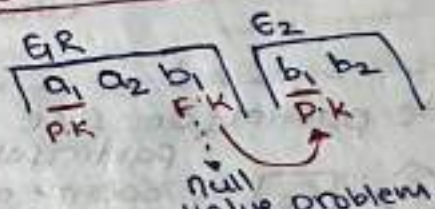
merge relationship R on "Total participation" E1 side

Solution 1 create relation for each.



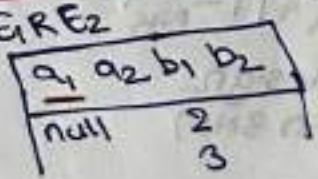
3 table required
Best design

Solution 2 only if null value are allowed



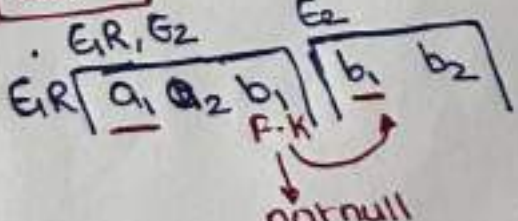
2 table required

~~wrong soln~~ **Solution 3** → Entity Integrity Prblm



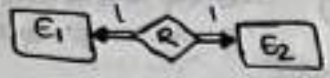
Soln 1 now you will not have null prblm

- E1R, E2



2 table reqn
Best Design

(iii) both side total participation } only 1 table required



Best Design: E_1RE_2

a_1	a_2	b_1	b_2

there is in BCNF

P.K can be either a_1 or b_1

C.K: a_1, b_1

FD. $a_1 \to a_2$, $b_1 \to b_2$

both S.K

▸1-M mapping (Similar to WES)

(i) (Total on many side)

2 table required

Best Design

E_1

a_1	a_2

P.K

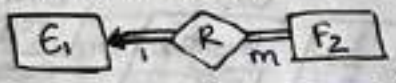
E_2R

b_1	b_2	a_1

P.K F.K

no null problem b/c E_2 has Total participat°

(ii) (Total participat° on Both side)



E_1RE_2

a_1	a_2	b_1	b_2

CK = b_1

not in 3NF

$a_1 \to a_2$

not 3NF (Violation)

not a good design

good design (Total on Both side)

E_1

a_1	a_2

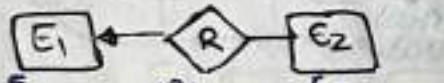
E_2R

b_1	b_2	a_1

So,

2 table required

(iii) many side partial (one side participation doesn't matter)



E_1

a_1	a_2

P.K

R

a_1	b_1

F.K

E_2

b_1	b_2

P.K

3 table req.

good design. (Also in 3NF)

Sol 2 null value prblm

E_1

a_1	a_2

E_2R

b_1	b_2	a_1

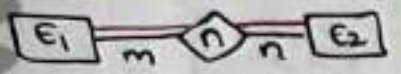
null value prblm

- if null value is allowed then 2 table required

• many to many
(what ever participation you take)

3 table required ⟶ E_1 | R | E_2

E_1: a_1 a_2 (PK) — R: a_1 b_1 (F.K) b_2 (P.K) — E_2: b_1 b_2 (P.K)

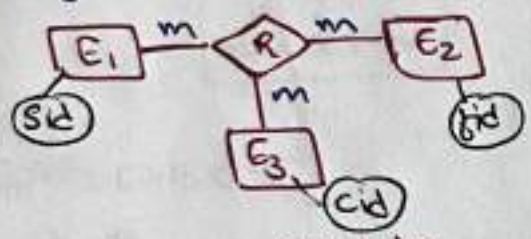


if we merge them

E_1RE_2 (Bad design) $\{ a_1\ b_1\ a_2\ b_2 \}$ (a_1 b_1 P.K)

$a_1 \to a_2$
$b_1 \to b_2$
(partial) not in 2NF (Violation)

• non-binary Relationship:
Degree of Relationship ≥ 3

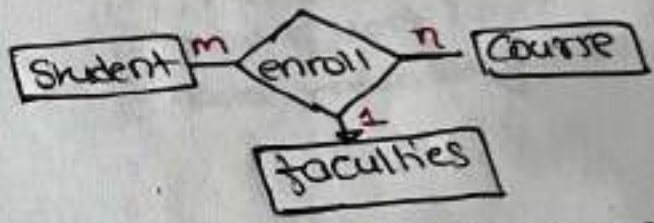


i.e many on all side.

R(Sid, fid, cid)
P.K

(Note) we don't allow more than one Arrow (one side) in non-binary Relationship

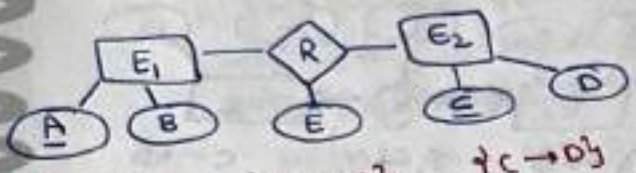
▸ many on all side, except one:
(only one Arrow)



• for a particular (student, course) combination, atmost one faculty

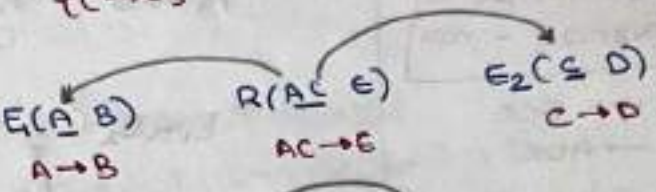
FD: Sid, cid → fid
P.K

[2] Many : Many mapping :



{A → B} {AC → E} {C → D}

RDBMS design :



min table required is 3
4 1 2 F.K

E1R(A B C E) E2(C D)

A	B	C	E
a1	b1	null	null

C → D

A → B
AC → E
{AC}
not 2NF

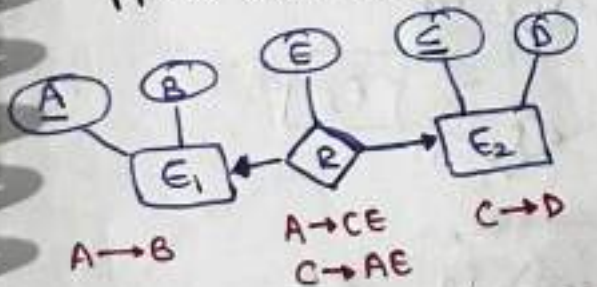
(a) M:N failed False
(b) pp at E1 lost True
(c) pp at E2 lost False
(d) Data redundancy True

E1RE2(A B C D E)

A → B
C → D
AC → E
AC : CK

(a) M:N lost
(b) pp at E1 lost
(c) pp at E2 lost
(d) Data redundancy

[3] 1 : 1 mapping with pp at both side



R(A C E)

E1

A	B
a1	
a2	
a3	
a4	

Key

R

A	C	E
a1	c1	-
a2	c3	-
a3	c4	-

Key Key

E2

C	D
c1	-
c2	-
c3	-
c4	-

Key

F.K (should be unique)

E1

A	B
a1	-
a2	-
a3	-
a4	-

A → B

E2R

C	D	A	E
c1	-	a1	-
c2	-	null	null
c3	-	a2	-
c4	-	a3	-

C → DAE
A → CE
{A C} C.K
A.K C.K

min 2 table
4 4 1 FK

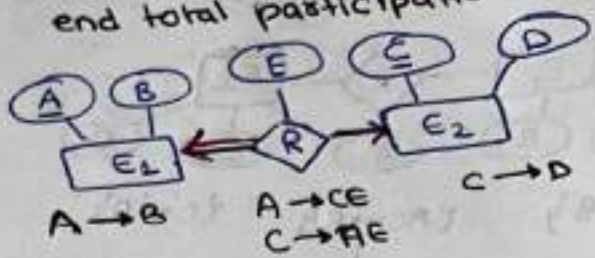
E_1 R E_2

A	B	C	D	E
a_1	–	c_1	–	–
a_2	–	c_3	–	–
a_3	–	c_4	–	–
a_4	–	null	null	null
null	null	c_2	–	null

A → BCE
C → ADE
{A, C} C.K

no.CK with NOT NULL only

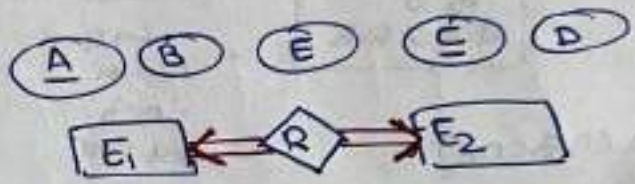
1:1 mapping with atleast one end total participation :-



A → B A → CE C → AE C → D

$E_1 R E_2$ (A B C D E) {A, C}
AK PK

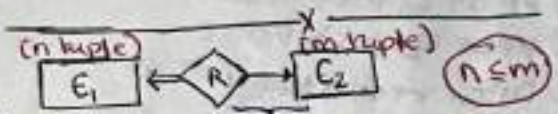
{A → BCE
C → ADE}



$E_1 R E_2$ (A B C D E)
PK UNIQUE NOT NULL

{A → BCE
C → ADE}

min 1 table 4
0 F.K

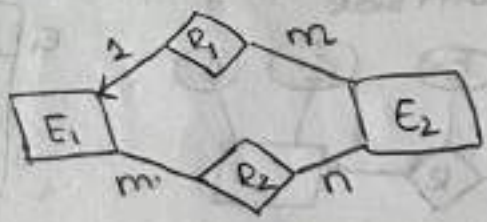


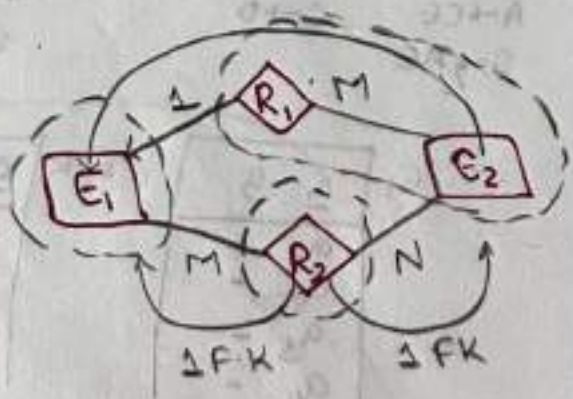
↳ min 1 table is sufficient but practical what we do is.

- if these participation is very less. So better combine $E_1 R$ together & E_2 as Separate.
- if the participation is huge or appear to the other then you can merge both.

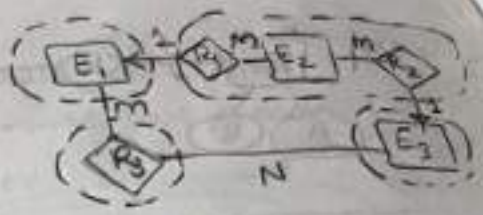
Q] E_1, E_2 are Entity set
R_1, R_2 are Relationshipset related b/w E_1 & E_2 with 1:M & M:N

how many Relationship Required — 3 table, 3 F.K



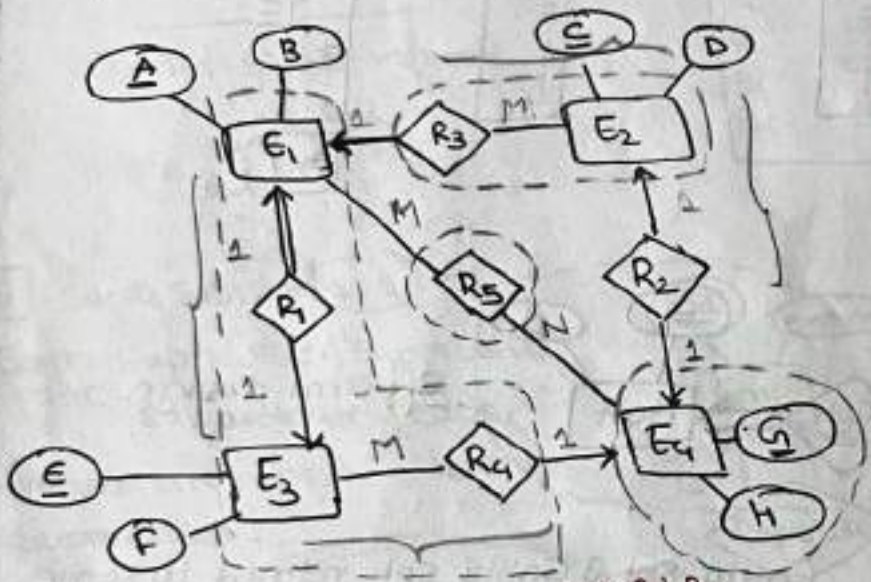


Qus] E_1, E_2, E_3 (Entity set)
R_1 Relationship set b/w E_1 & E_2
with 1:M & R_2 Relation
b/w E_2 & E_3 with M:1
& R_3 Relation b/w E_1 & E_3
with M:N 4 table



Q] How many min. Rel. table ? 4 table

- #f F.K's in minimized RDBMS design? 5 FK
- #f Total Attribute in All minimized RDBMS design? 5+4+2+2 = 13



$R_1(\underline{A}\,\underline{E})$
$R_2(\underline{C}\,\underline{G})$
$R_3(A\,\underline{C})$

M:1 Rel R_4

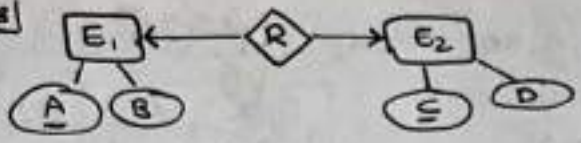
$E_1R_1E_3R_4(\underline{A}\ B\ \underline{E}\ F\ \underline{G})$
AK PK FK

F.K FK
$E_2R_3R_2(\underline{C}\ D\ A\ G)$
{C, G}
P.K A.K
1:M Rel R_3
1:1 R_3

$E_4(\underline{G}\ H)$

FK FK
$R_5(\underline{A}\ \underline{G})$

Ques]



with 30% participation at E_1 & 80% participation with which is best possible design?

(a) E_1E_2 merge in one rel with no FK

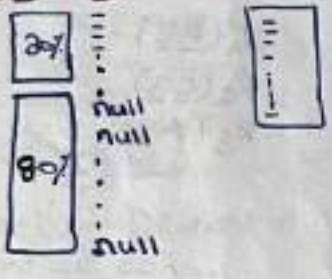
(b) E_1, E_2 kept separate with F.K at E_1

(c) " " " " " " E_2

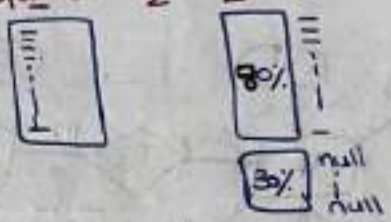
(d) " " " " " " E_1 & E_2

$E_1R(\underline{A}\ B\ \underline{C})$
$E_2R(\underline{C}\ D\ \underline{A})$

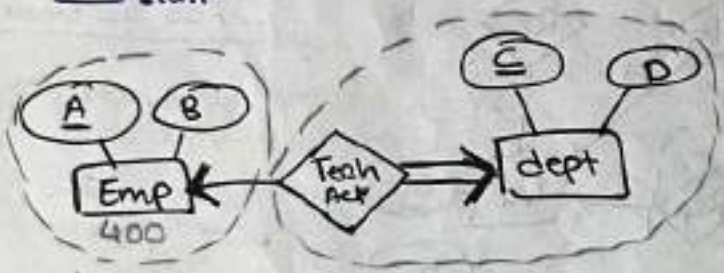
[b] $E_1R(\underline{A}\ B\ \underline{C})$ $E_2(\underline{C}\ D)$



[d] $E_1(\underline{A}\ B)$ $E_2R(\underline{C}\ D\ \underline{A})$



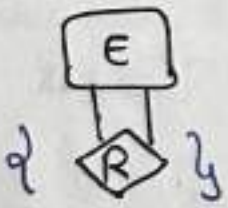
Ques]



efficient design

Self Referencial Relationship set (Recursive entity set)

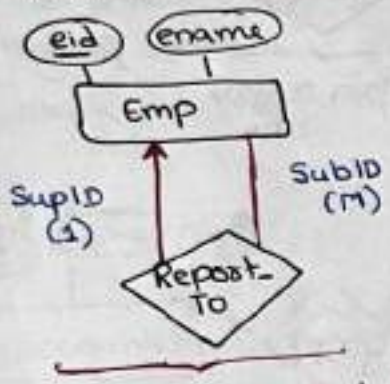
Entity of Entity set related to some other order pair entities of same entity set



mapping 1:M
M:1
1:1
M:N

Emp entity set, Report_to rel set
related b/w Supervisor (SupID) & subordinate (SubID)
{Each supervisor, superwise many subordinate & each subordinate report to one supervisor.

(i) SupID, SubID → 1:M



Min 1 table required
1 F.K

Emp (eid ename)
e_1 –
e_2 –
e_3 –
e_4 –

Rep (1:M)

SupID	SubID
e_1	e_2
e_1	e_3
e_2	e_4

↑ key

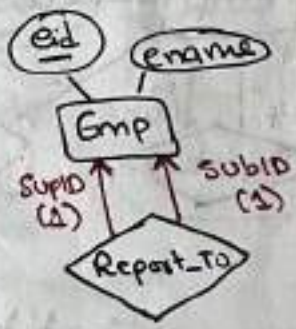
Emp_R

eid	ename	SupID (FK)
e_1	A	null
e_2	B	e_1
e_3	C	e_2
e_4	D	e_3

[RDBMS design]

(ii) SupID, SubID → 1:1

Each supervisor, superwise one subordinate & each subordinate report to one supervisor



min 1 table
1 F.K

Emp (eid ename)
e_1 –
e_2 –
e_3 –
e_4 –

Rep

SupID	SubID
e_1	e_2
e_2	e_3
e_3	e_4

↑ key ↑ key

{SupID, SubID} C.K

Emp_R

eid	ename	SupID (FK)
e_1	A	–
e_2	B	e_1
e_3	C	e_2
e_4	D	e_3

OR

{eid P.K
SupID}
A.K

Emp_R

eid	ename	SubID (FK)
e_1	A	e_2
e_2	B	e_3
e_3	C	e_4
e_4	D	–

B.K
{eid, SubID}
A.K

[iii] SupID, SubID :- M:N

Each Supervisor can superwise many subordinate each.

Subordinate can report to many Supervisors.

min 2 table 2 FK

Emp(eid ename)

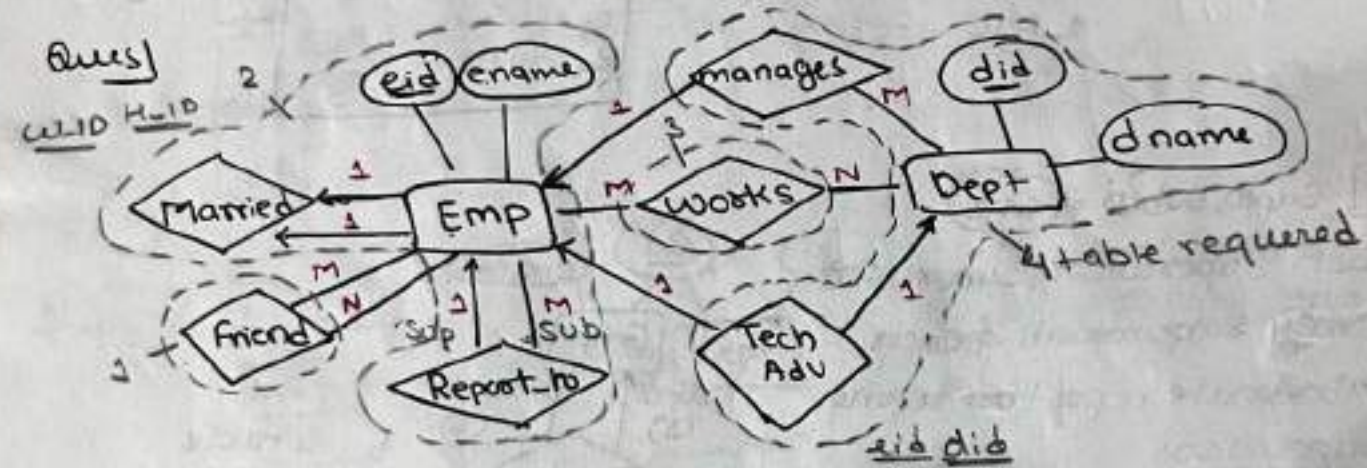
e_1	-
e_2	-
e_3	-
e_4	-

Rep

SupID	SubID
e_1	e_2
e_1	e_2
e_4	e_3

{SupID SubID} CK.

Ques)



1:M 1:1

FK FK

Emp(eid ename Sup WID)

Dept(did dname eid Techeid)

FK FK

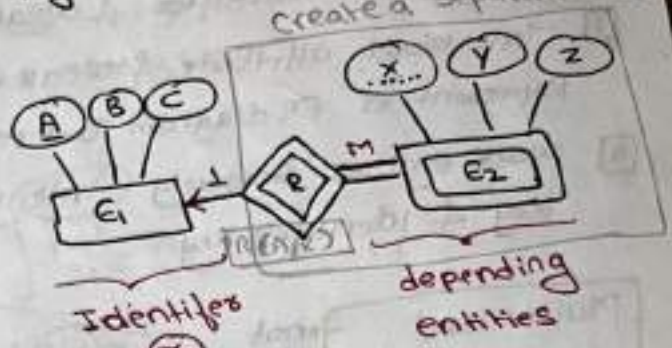
1:M 1:1

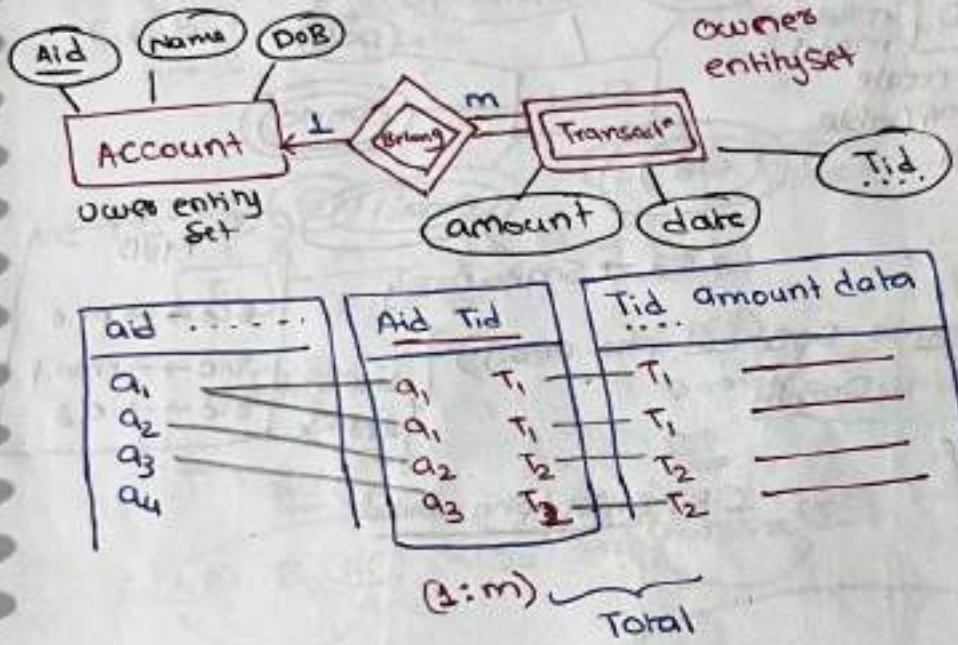
cowork(eid did)

Friends(eid_1 eid_2)

Weak entity Set — denoted by — Partial key — • entity set w/o any key

• entity set with no key (Attrs. of weak entity set is not able to differentiate entity of weak entity of weak entity set uniquely]





• Entities of weak entity set are not independent entities which are depends on other strong entity set which is called Identifier / OWNER entity set.

1:m RDBMS design

F.K

Account(Aid ...) Transbelgs(Aid Tid amount data)

Account	Transbelgs			
a_1	a_1 T_1	5000	20/11	
a_2	a_2 T_1	6000	21/11	
a_3	a_2 T_2	.....		
a_4	a_3 T_1	.....		

weak entity set:
• Different entities but we can't identify them uniquely in weak entities set

ex.

eid	ename	c-name	age	
101	E_1	monu	10	are different entities
		sonu	10	
102	E_2	monu	10	
		chintu	10	

1. participation at weak entity set side must be total
2. Mapping btw identifier & weak entity set must be one to many [1:M]
3. Multivalued attribute & weak entity set is allowed to Represent in ER diagram but not allowed in RDBMS
4. weak entities uniquely identifies by combining partial key & identifier key

Multivalued Attribute ERD → not allowed in Rel. model

*for each MVA, we create a separate relation/table in relational model

{ sid → SnameDOB }

MVD
sid →→ pho
sid →→ email
sid →→ cid

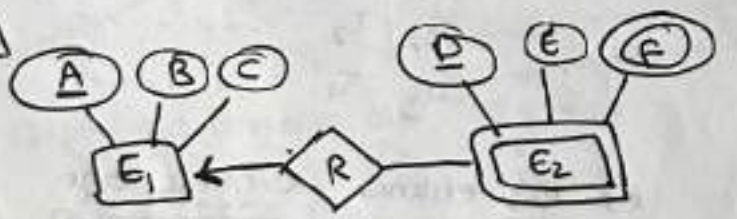
Stud(Sid Sname DOB cid pho email) } 1NF✓ 2NF✗
{sid → SnameDOB}
P.A

C.K: Sidcidphoemail

Stud(sid sname DOB)
sid → SnameDOB

Stud2(Sid cid phno email) } BCNF✓
no nontrivials FD

Qus)



(1) How many min rel which satisfy 1NF? 2table
(2) " " " " " 3NF? 3

$E_1(ABC)$
$A \to BC$

$R(AD)$	$E_2(DEF)$
$D \to A$	$D \to E$

1:M Rel Set merge with right side entity set

$E_1(\underline{A}BC)$ $E_2R(DEA)$

$A \to BC$
CK: A

$D \to AE$
CK: DF

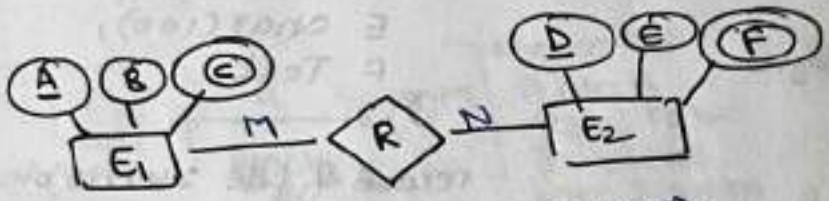
1NF
not 2NF ✗

$E_1(\underline{A}BC)$ $E_{21}R(DA)$ $E_{22}(DF)$

$A \to BC$ $D \to AE$

2NF ✓
3NF ✓
BCNF ✓

i.e

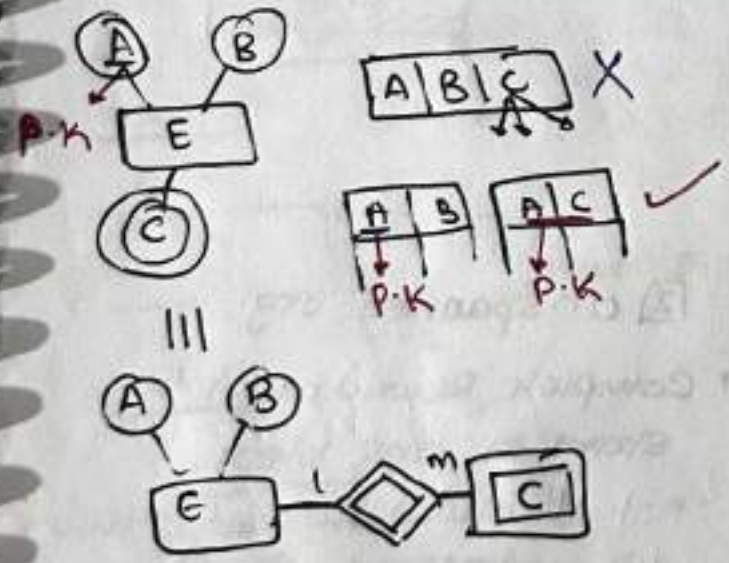


$E_1(ABC)$
$A \to B$
AC

$R(AD)$
1

$E_2(DEF)$
$D \to E$
DF

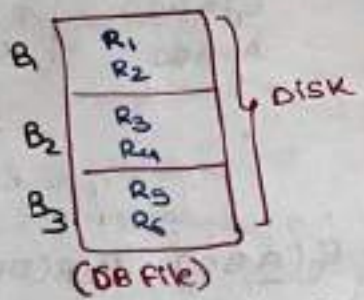
$E_{11}(\underline{A}B)$ $E_{12}(\underline{A}C)$ $R(\underline{AD})$ $E_{21}(\underline{D}E)$ $E_{22}(\underline{DF})$ } 5 table for BCNF



- every MVA can be converted in a weakES

File org. 44 Indexing

database is collection of file or table
file is collection of block or pages
Block is collection of record or tuples



(DB File)

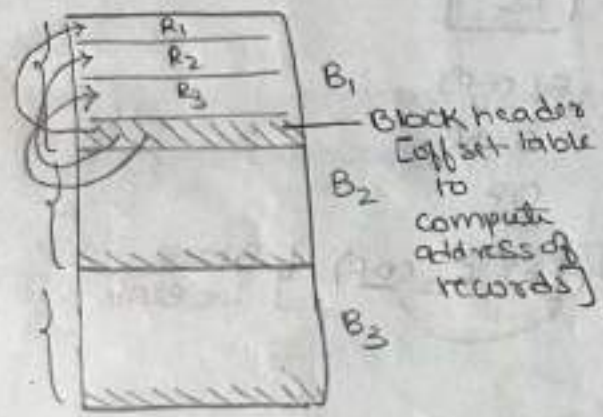
Data access from Disk [DB file] is block by block.

⇒ Record in DB file (DB table)

1. Fixed length records

```
Create table R
( A char(100),
  B char(200),
  C char(100)
);
```

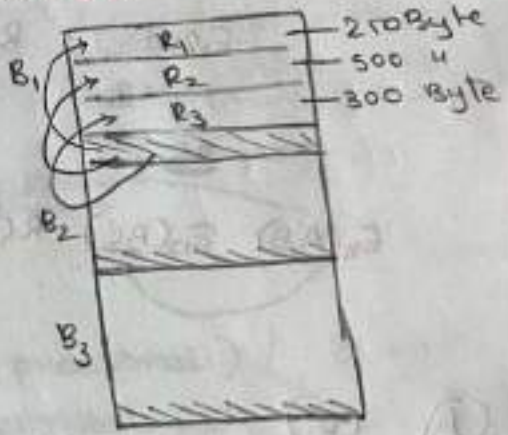
Record Size : 400 Bytes



2. Variable length record

```
Create table S
( D char(100),
  E char(100),
  F Text
);
```

record of file : variable length



Records organization in DB file

1. spanned organization

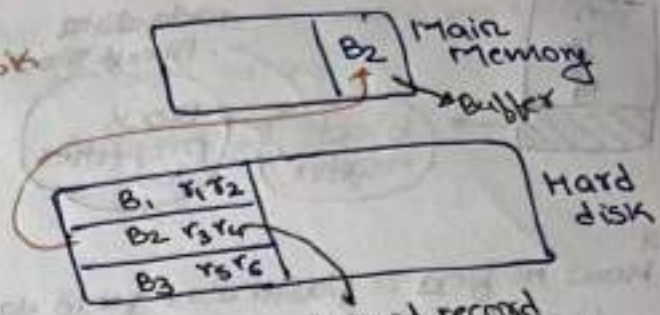
Record allowed to Spanned n more than one block unable to Store in one Block bic of the Space So the rest of the Record is Store in another Block

2. unspanned org.

- Complete Record must be store in one block.
- not allowed to spanned in two different block

• File organisation

Relation R → Store on Disk as a single file



then Block is transferred not sector

Hard disk :

Sector
↳ smallest unit of Data Access

DBMS
def) Block = 3 sector size
↳ smallest unit of Data Access for DBMS

• whenever a certain portion of the data is need, it must be located on disk, copied to M.M for processing & then rewritten to the disk if the data is changed.

a → M.M

Block
↳ a contignous sequence of sectors from a single track

"Access cost" # Block needed to be Transfered from Disk to M.M

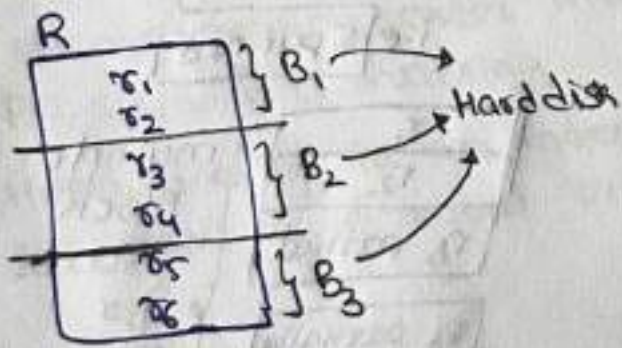
Record ≡ Tuple in relation
file ≡ Relation

Block = page → not the page from O.S paging Concept

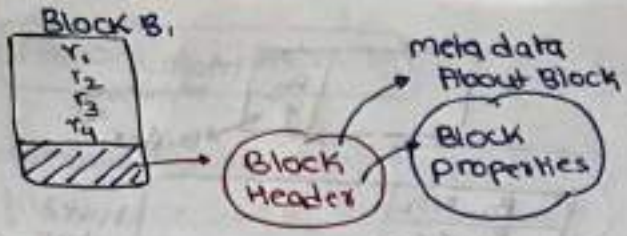
Block are also know as page

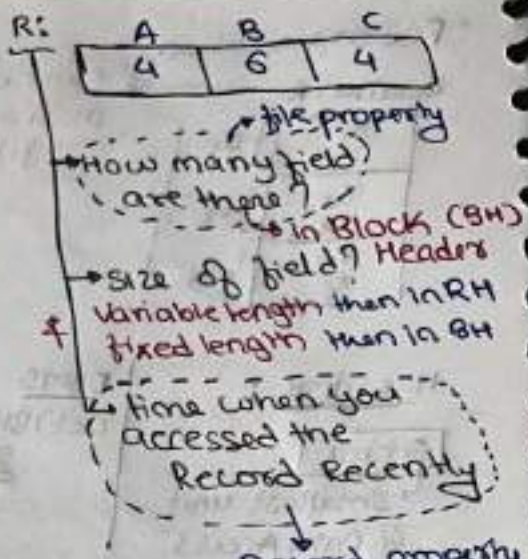
• Record Property : ex: Time stamp of record
↳ Stored in the record

One Relation — stored as — one file ⇒ (some Sequence of Block)



Block size ≥ record size

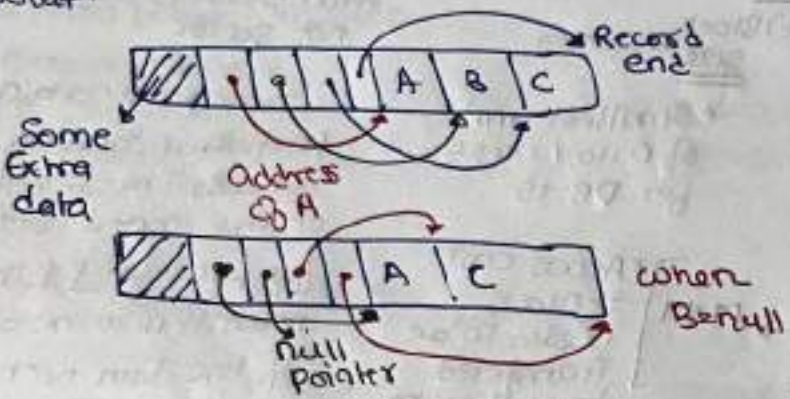




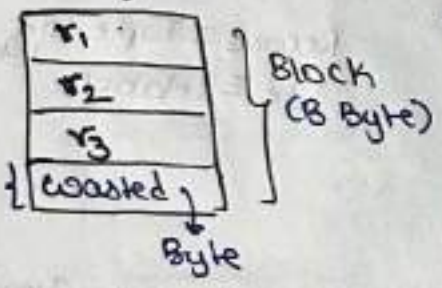
idea

ex: How to find a particular field value inside a variable length Record.

Solutn



Block factor



unspanned

- no. of record in a block $\left\lfloor \frac{B}{R} \right\rfloor$
- unused space in each block

$B - (\text{used space})$

$B - (\text{Block factor} * \text{no. of record})$

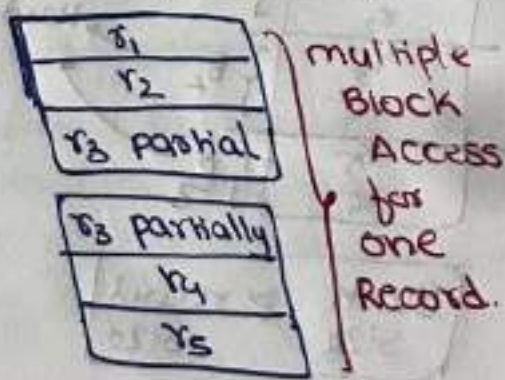
$\Rightarrow B - \left(\left\lfloor \frac{B}{R} \right\rfloor * R\right)$

unspanned organizatn are used for fixed length

i.e unSpanned — By default

unspanned / spanned either can be used for variable length

i.e Spanned



note

For variable-length records using Spanned Organizatn each, block may store a different no. of records. In this case the blocking factor bfr represents the avg no. of record per block for the file

bfr : Avg. no. of Record per block

file of 'R' Records

$$\#\text{Blocks needed} = \left\lceil \frac{R}{bfr} \right\rceil$$

- Block address

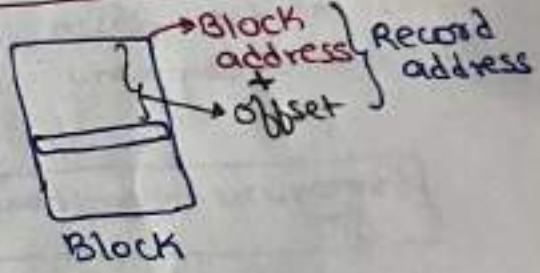
Address of 1st sector of the block

<C,h,S> address (or) LBAaddress

- Record Address.

A record can be identified by giving it's block address & the offset of the 1st byte of the record within the block.

Block address + offset



- Spare Index doesn't mean that for every Block you are creating one entry (that happens in Primary Index)

for every distinct value you create one index entries

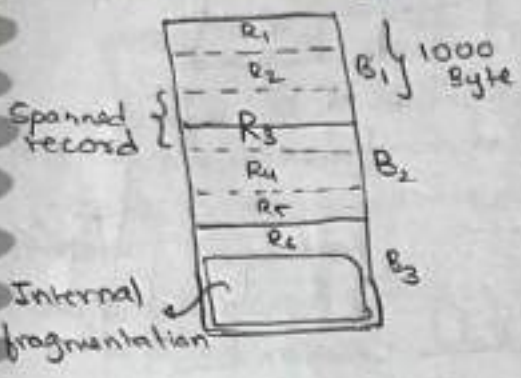
note → Data access always Happens Block by Block.

- we are learning order Indexing P.I, S.I, Cluster Indexing
- Hash Indexing is an indexing were we don't care about order.

① Spanned org.

Block : 1000 Byte

Record : 400 Byte

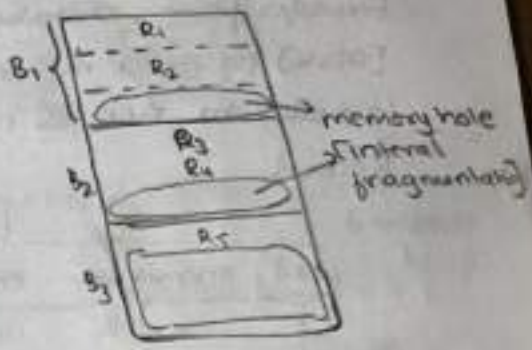


- less possible Internal fragmentation
- more access cost to access Spanned Record

② Unspanned org.

Block : 1000 Byte

Record : 400 Byte



- More possibility of Internal fragmentation
- Less access cost to access unspanned Record

Note:

- to organize record in DB file with fixed length record, unspanned Organisat^n is preferred. → Fixed length + unspanned org.
- to organise record in DB file with Variable length record, Spanned Org. is preferred → Variable length + Spanned org.

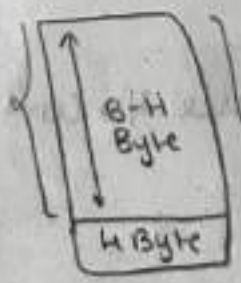
Block factor of DB file

max: possible records per blocks

Block size : B byte

Record size : R bytes [Fixed length]

Block header : H Byte

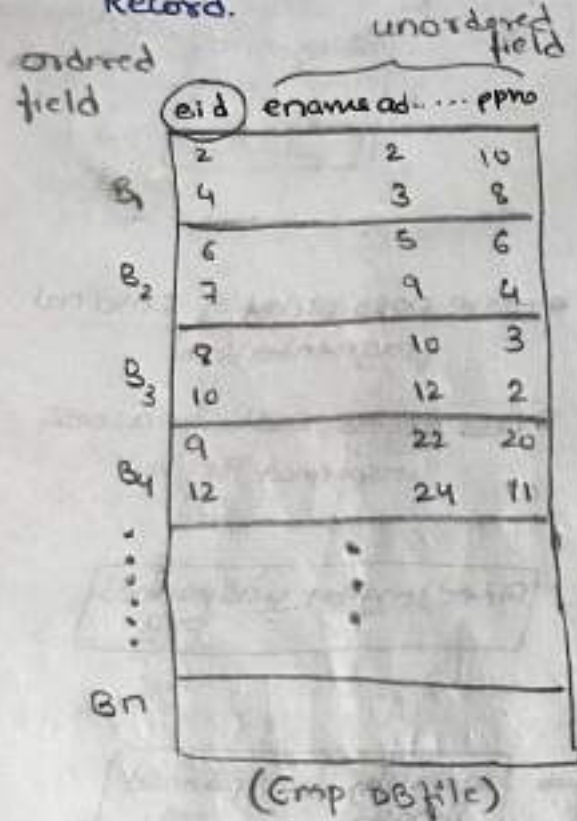


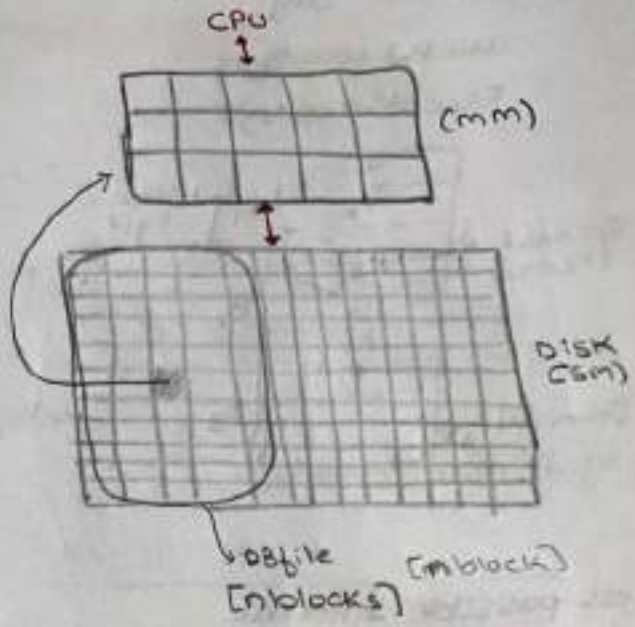
Block factor of DB file $= \left\lfloor \frac{B-H}{R} \right\rfloor$ record/Block (max possible)

i.e $\left\lfloor \frac{1000-0}{400} \right\rfloor = 2$ record/Block

Access cost or I/o cost

- no. of disk block required to transfer from Database file [Disk] to main memory in order to access required Record.





I/o cost to access record without Index :-

(i) over ordered field

```
Select *
From emp
where eid = x
```
ordered file

worst case $\lceil \log_2 n \rceil$ blocks access cost

(ii) over unordered field

```
Select *
From emp
where ppno = y
```
unordered file

worst case nblock access cost

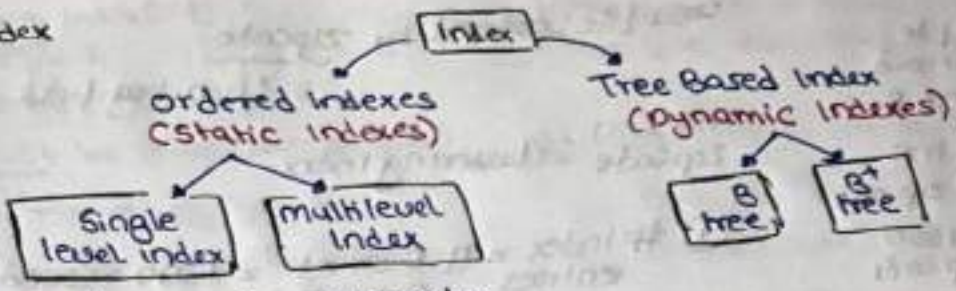
Indexing used to reduce I/o cost

note

Ordered Record (sorted file)

- A binary search for disk file can be done on the block rather than on the record.

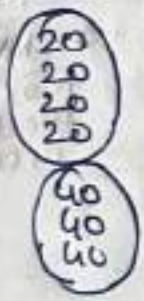
• Index



- To gain fast random access to record in a file, we can use an Index Structure
- Each index structure is associated with a particular search key
- A file may have several indices on different search key

---

2. • Clustered Index → Know as
   - sequentially ordered
   - by a non-key field 'F'
   - then the Index using Search key 'P'



file R(A, B, C, D) — non-key
Key
R physically ordered on (D) ordering field R
ordering non key field = clustering field

• Clustered Index condition
  - physically ordered
  - non-key field

& if you create Index on Clustering field then these is know as clustering I

Clustering Index:

Index entries = # of cluster = # Distinct search Key values

Index Entry < search key value , Block pointer >

for every Distinct Search Key value we make one Index entry

pointer to the (1st address of that cluster) 1st occurrence of Search key value

ex: ordered file
300000 record
B.S = 4096 B
ordered by the attribute zipcode
& there are 1000 zip code
(with avg. 300 record per zipcode)
5B zipcode & 6B block pointer
Record size: 100

Data file: ordered by zipcode
* non-key field

Index on zipcode : Clustering index

① # Index entries = # Distinct zipcode value = 1000 record in clusters index

② Index entry (Record) size: (zipcode size) Search key + Block address size
⇒ 5+6 = (11B)

③ Blocking factor of Index: # index entries per Block
⇒ $\left\lfloor \frac{4096}{11} \right\rfloor$ = 372 index per Block

⑥ with indexing
$\lceil \log_2 I \rceil$ + 8 Data Block
3 + 8 = (11)
max. Block access

④ # Index Block required: Index: 1000 Records
per Block ⇒ 372 Index Records

Index Block: $\left\lceil \frac{1000}{372} \right\rceil$ = 3

⑤ Searching [zipcode = 302017]
max. # Disk Block Access? (w/o Indexed)
file is sorted by zipcode & you are searching on zipcode
we can use binary search

Data Block
$\lceil \log_2 B \rceil$
so, $\lceil \log_2 7500 \rceil$ = 13 Block access

Block for file = $\left\lceil \frac{300,000}{40} \right\rceil$
⇒ 7500 Blocks

Data file:
Record size = 100B
Block size = 4096B
Record per Block = $\left\lfloor \frac{4096}{100} \right\rfloor$ = 40
(Bf of file)

doubt

[zipcode = 302017] Record we want to find

↳ 13 Block Access + 8 = 21 Block access
(1st occurring is counted)
(1st occurrence of this zipcode)

$\left\lceil \frac{299}{40} \right\rceil$ [...] → 1, [ ] → 40, [ ] → 40, [ ] → 40
on avg. 300 record per zipcode

1. Primary Index [P.I]
on a file R
to create P.I on file R
R must be physically ordered
- Key field f
- "Index by f" is called P.I

- almost one P.I can be created on a file

ex. order of Data file is same as order of P.I } True

ex. Index on P.K is called P.I ?? (No)
(1) Data file may not be ordered by P.K)
(2) may be data file is heap file

ex: Data file R(A,B,C)
order (Sort) by (A) → ordering field
if A is (Key) ⇒ file ordered by (A) ordering key field.
& if A is key ⇒ key field

- Primary Index
  - index entry for every Record
  - Dense Implementation possible (But not used)
  - Sparse Implementation (used)
    - for P.I, one index entry for one Block

Sparse Index (By default)
<Search key value of Anchor Record, Block Pointer>
Block Pointer → Can't be Record Pointer

Dense Index (Not used)
<Search key value of a Record, Record (or) Block Pointer>

Data file
ex: no. of block 'b' = 9500
Block size 'B' = 4KB
OKF (ordering key field) length 'V' = 15B
Block pointer 'p' = 6B

Index file on OKF
it's primary Index

① # index entries = # Block in data file
= 9500

② Size of Index entry = OFK size (Search key) + Block pointer
⇒ 15+6 = 21B

③ Blocking factor of Index file
→ # Record per Block
→ # Index entrie per Block

Index entry size = 21B
B = 4KB
$\left\lfloor \frac{4KB}{21B} \right\rfloor = 195$ entries in a Block

④ # Block needed for (Index file)
→ 9500 Index entries

one Block = 195 Index entries

$\left\lceil \frac{9500}{195} \right\rceil = 49$ Index Block

⑤ max. Access cost using P.S
$\lceil \log_2 49 \rceil + 1 = 7$ Block need to access in w.c
(+1 for data Block)

⑥ max Access cost w/o P.S
Binary search still can be applied as it's sorted

$\lceil \log_2 9500 \rceil = 14$

---

ex: ordered file / unspanned
300,000 records
BS = 4096 B
File record are fixed
record length R = 100 B
ordered by key "K" field

find K = 7 record (Data record)
worst case no of disk block we need to access for this query.

Binary search on ordered file

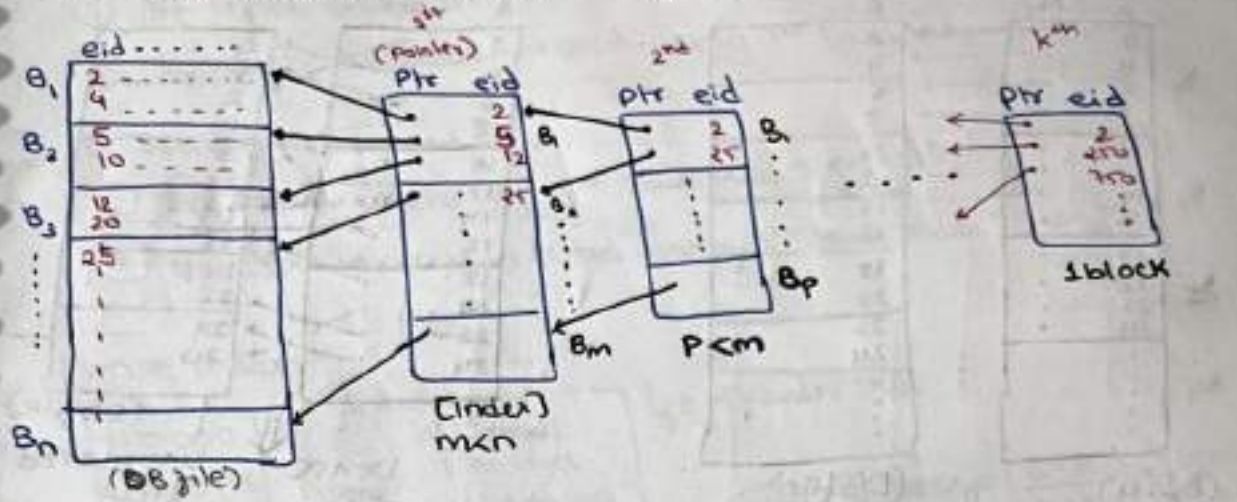
#Record/Block = $\left\lfloor \frac{4096}{100} \right\rfloor = 40$ Record per block

#Block = $\left\lceil \frac{300,000}{40} \right\rceil = 7500$

access cost = $\lceil \log_2 7500 \rceil = 13$ Blocks needed

- if ordered by non-key field then access cost will be 7500 Blocks

→ I/O Cost to access record using Index:-



* I/O Cost $\lceil \log_2 m \rceil + 1$

* I/O Cost $\lceil \log_2 p \rceil + 2$

* I/O cost : $k+2$ blocks

1. I/O cost to access record using 1st level index = $\lceil \log m \rceil + 1$
2. I/O cost to access record using Multilevel index : $k+1$

Multilevel Index : Index to index until last level 1 Block index

- Entry of Index is two field <search key, pointer>
- Block factor of index file :- max # of index entries per block

Search Key : K Byte
pointer : P Byte

Bf of Index: $\left\lfloor \frac{B-H}{1K+P} \right\rfloor$ (entries/block)

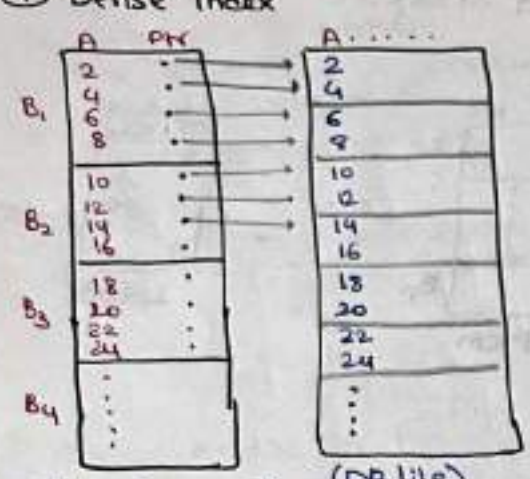
[also known as address of the disk block]

Bf is used to find num of block req. to store index

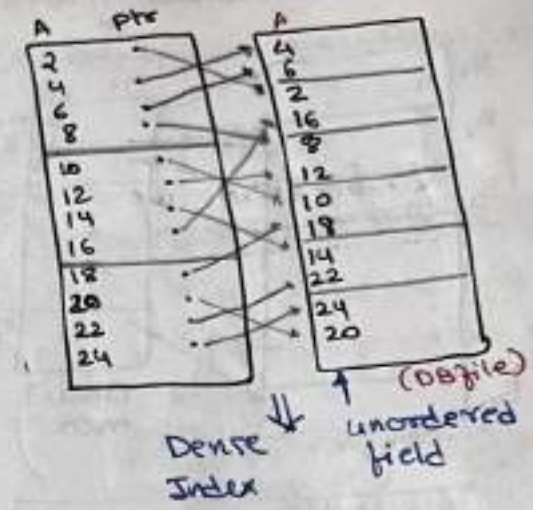
Categories of Index:

1. Dense Index
   - for each record of DB file there must be index entry in index file
2. Spare Index
   - For set of record of Database file there exists an entry in index file

① Dense Index

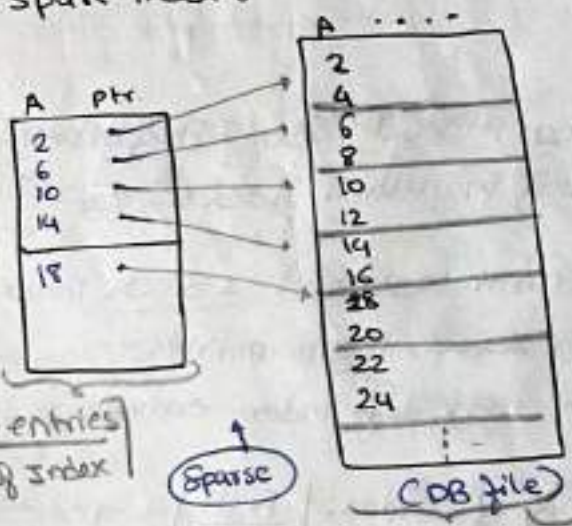


[Dense]



- For DenseIndex the no. Index enter = no. record

② Spare Index :-



- If Sparse Index over key no. of Index enter of Index file = no. of block of DB files

$\left\lceil \frac{\#\text{ of entries}}{\text{By of Index}} \right\rceil$ $\left\lceil \frac{\#\text{ record}}{\text{By of DB}} \right\rceil$

Qus] DB file consist 1lakh record with record size 100 bytes stored Disk with block size 1024 bytes. search key size 12byte pointer size: 10

① No Cost to access record without index

(i) using unordered field? n: 10000 blocks (no. of block)

(ii) using ordered field?

$\lceil \log_2 n \rceil = \log_2 10000$

$\Rightarrow 14$ blocks

(2) if Dense index used

(i) #f index block at 1st level index? m=2174

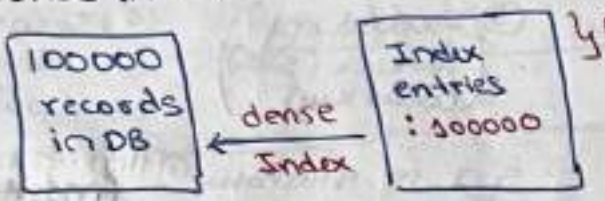
(ii) I/O cost to access record using 1st level index $\lceil \log_2 m \rceil + 1$ → (15)

(3) if sparse index used

(i) #f index blocks at 1st level index? m=218

(ii) I/O cost to access record using 1st level index? $\lceil \log_2 m \rceil + 1$ → (9)

Dense Index:-



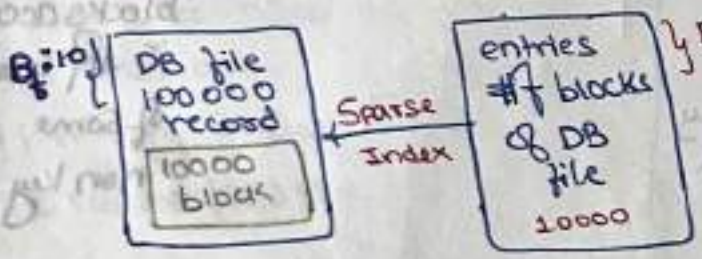
Bf of Index: 46

$Bf_{index} = \left\lfloor \frac{B-H}{K+P} \right\rfloor = \left\lfloor \frac{1024-0}{12+10} \right\rfloor$

= 46 entries/block

$\#f_{dense\ block} = \left\lceil \frac{100000}{46} \right\rceil$ = 2174 block

Sparse Index:-



$Bf_{DB} = \left\lfloor \frac{B-H}{R} \right\rfloor = \left\lfloor \frac{1024-0}{100} \right\rfloor$ = 10 R/block

$\#f_{DB\ blocks} = \left\lceil \frac{100000}{10} \right\rceil$ = 10000 blocks of DB

$\#f_{sparse\ blocks} = \left\lceil \frac{10000}{46} \right\rceil$ = 218 block

(4) MLI

(i) #f levels with dense at 1st level?

(ii) #f level with sparse at 1st level?

(iii) I/O cost using MLI with 1st level dense?

(iv) I/O cost using MLI with 1st level sparse?

Type of Indexing

Search key : Field used for Indexing

- Ordered field
  - KEY [PK/AK]
    - ① Primary Index (P.I)
  - NON Key
    - ② Clustering Index (C.I)
- Unordered field (cheap file)
  - Key [PK/AK]
    - ③ Secondary Index (S.I) (over key) (nonclustered Index)
  - non key
    - ④ Secondary Index (over non-key) (mostly Sparse)

Block access

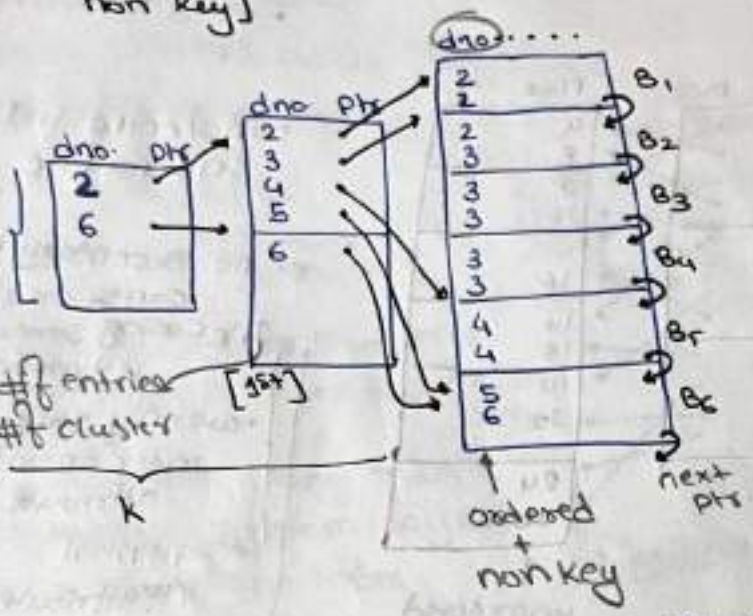
Dense/**Sparse	mostly Sparse	(Always dense)	(mostly Sparse)
ML1 : (K+1) blocks cost	ML I	ML1 : (K+1)	ML1 :
1st level : $\lceil \log_2 n \rceil + 1$	K + one or more blocks of DB	1st level : $\lceil \log_2 n \rceil + 1$	K+1+ # of DB blocks access is # records of same non key.

$\lceil \log_2 n \rceil$: Binary search on Index Block

+1 : 1 more Access for Data Block

(I^th) Index Block

- Clustering Index [ordered field + non key]



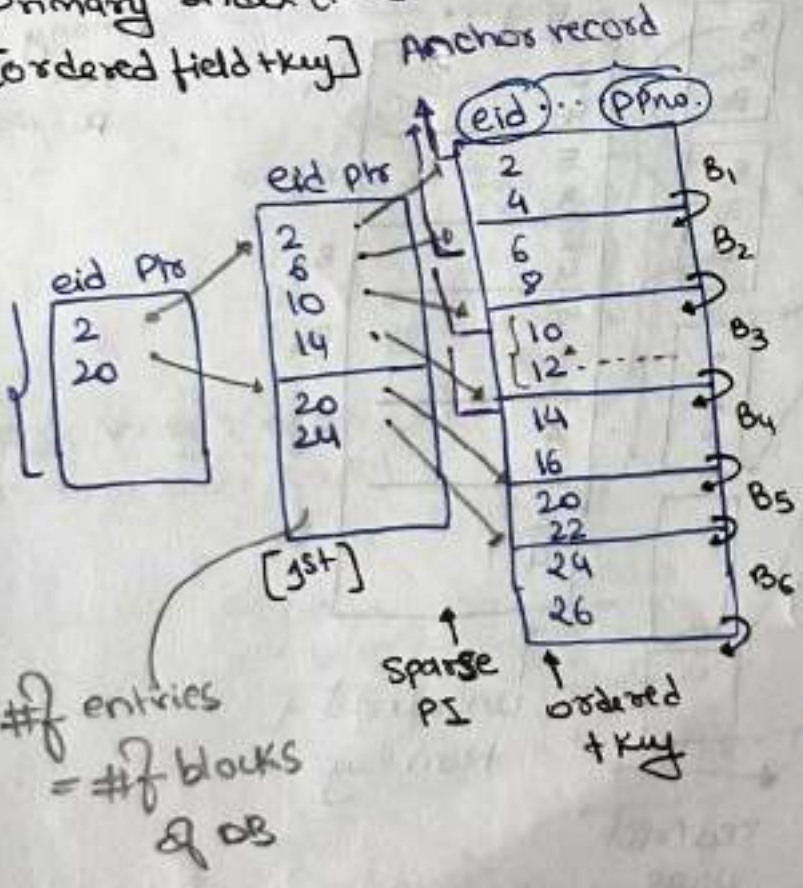
Select * from emp where dno = 5; } No cost using MLI with 1st level CI

K + one or more blocks until nxt cluster begins
∀ DB

- Atmost one clustering indexing is possible bc search key must be ordered field

- Clustering index mostly sparse index [but dense also possible, if each cluster with one record]
- for either relation, either primary index or clustering index possible but not both.

- Primary Index [P.I]
[ordered field + key]



- It can be Dense or Sparse [Sparse primary Index is preferred]

- atmost 1 P.K is possible for any RDBMS table bc search key must be ordered field.

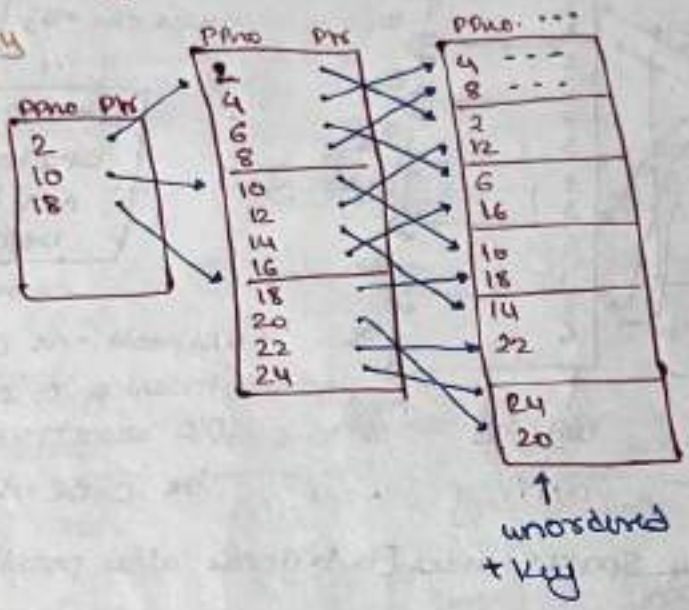
- these Index is know as P.I
- A Index on field (F)
C.K of Data file & Data file sorted by F

- Secondary Index [6·1]: SI is Alternative Index even PI & CI already exist
  [unordered field + key/ non key]

(8) SI over keys:

- many SI over key possible
- unordered field in Data file

Secondary Index will be ordered



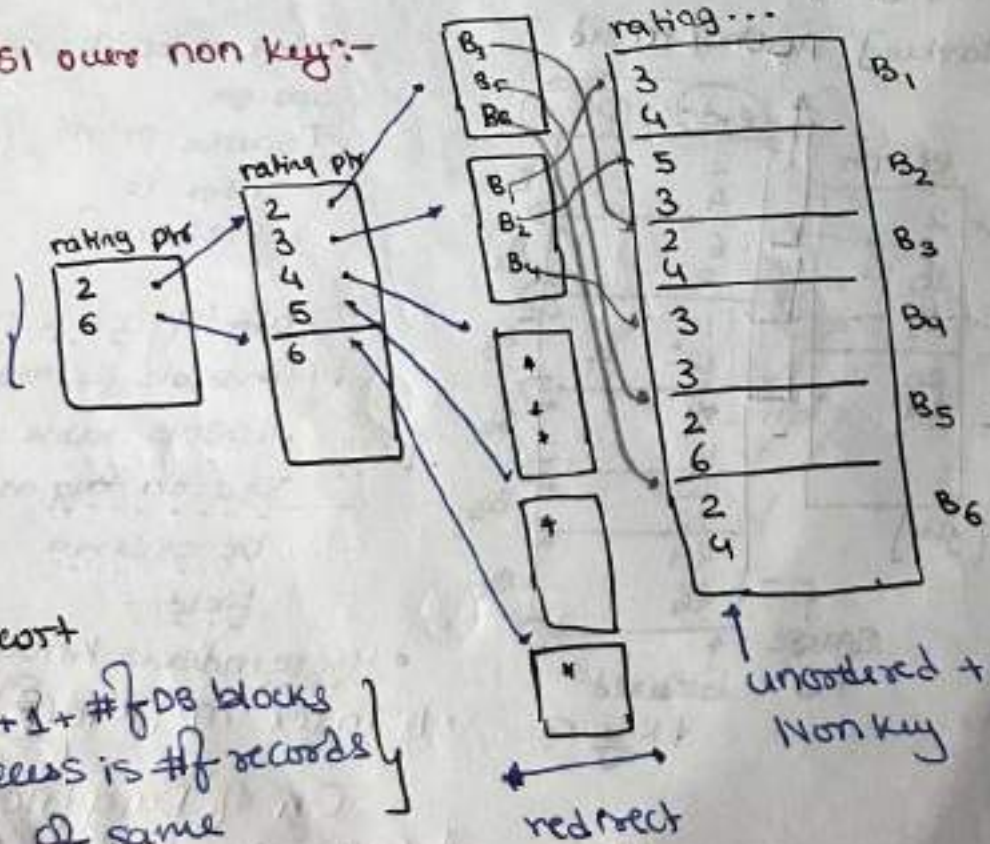
↑ unordered + key

er. no. Secondary index can be created on a file (using single attribute)

- we only study index on single attribute.
- sequential (#attribute - 1)
- heap (#attribute)

(4) SI over non key:-

- many SI over nonkey possible.



WC I/O cost

⇒ { k+1+ # of DB blocks access is # of records of same non key }

redirect page

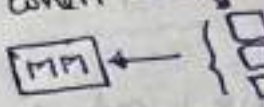
↑ unordered + Non key

- So far,
  - Single level ordered Indexed { P.I, C.I, S.I } < on key / on non key
    - Index is ordered
  - I (Index Block), B (Data Block)
  - Access Cost (I/O cost): $\lceil \log_2 I \rceil + 1$ (for data block)
    - Binary Search on Index

I/O cost: $\lceil \log_2 I \rceil + 1$
- binary search

To Reduce I/O cost:

① Put Index in M·M
- good only when index size is very small → almost 2-3 Block



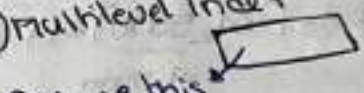
- may be okay for Primary Index

Prblm ① Index larger than M·M

Prblm ② Even if Index Size < m·m Size

there M·M has many things to accomodate so m·m may not be able to take Index

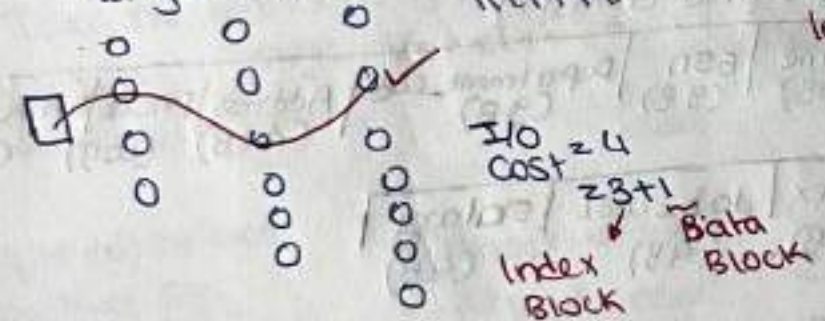
② Multilevel Index



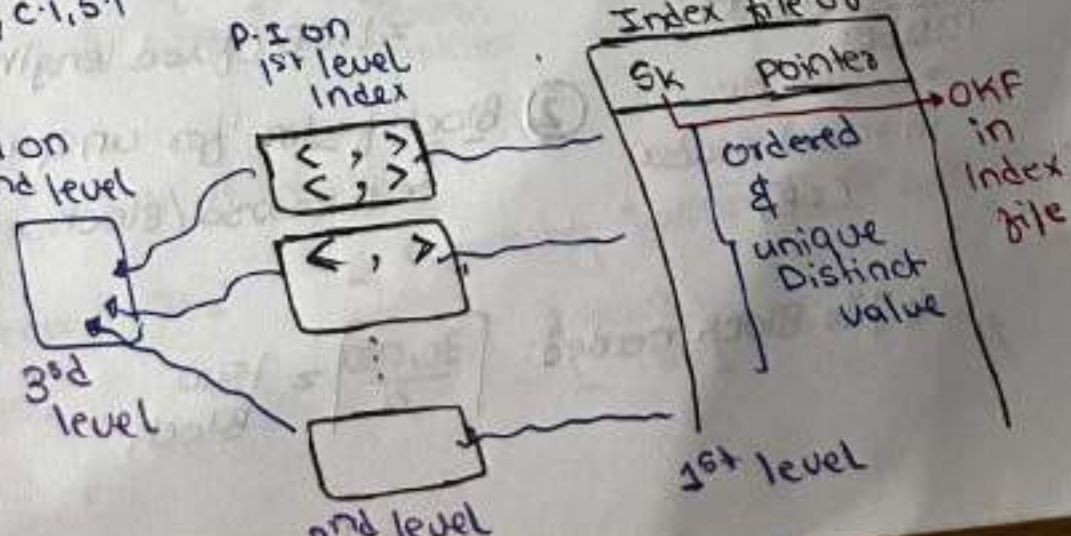
ex: we go for multilevel Index b/c Index doesn't fit in one block?

when no. of Index Block is larger & you are unable to put in M·M (NO)

- once we Decide to go for multilevel Index then by default outermost level has one Block



- Single level Indexes
  - Primary Index, C·I, S·I

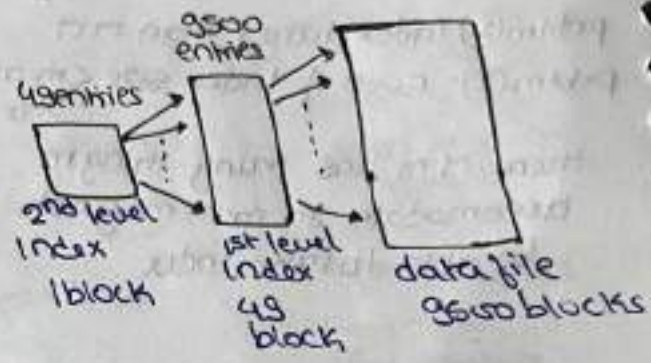


Note

- 2nd level & greater are always sparse
- Index level "K" is P.I (sparse) on Index level "K-1"
- Index level "1" is P.I / CI / SI
  Inner most level
- Last (outermost) level is called Top level

ex Assume 1st level is PI on Data file & 2nd level is Dense Index on 1st level then Problem
↳ All Index level copy on each other

- MLI is better choice than Binary search on single level



0: If Data file is ordered

① Approach 1: w/o Index

I/O cost = $\lceil \log_2 9500 \rceil = 14$
Data block

② Approach 2: with Single level Index

I/O cost = $\lceil \log_2 I \rceil + 1 = 6 + 1 = 7$

③ Approach 3: MLI

2+1 = ③

X

record field consist of

Name (30B)	SSN (9B)	Department_Code (9B)	Address (40B)	Phone (10B)	Birth_date (8B)	Sex (1B)	Job_Code (4B)	Salary (4B)

ex: B = 512B
Block pointer (P) = 6B
record pointer (P_R) = 7B
30,000 record of fixed length
Additional 1 byte is used as a delet^n marker
↳ it will be in record header (1B)

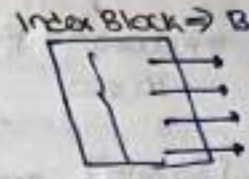
④ Record size = 30+9+9+40+10+8+1+4+4+1
= 116B (fixed length)

② Blocking factor for unspanned organizat^n
↳ # Record / Block

$\left\lfloor \frac{512B}{116B} \right\rfloor = 4$ in one Block

file Block needed: $\left\lceil \frac{30,000}{4} \right\rceil = 7500$ Block

③ Suppose that ordered by Ssn & Primary Index on ssn



Second level

Index Record size

⇒ 9 + 6 = 15 B

- 9 B: SK size (SSN)
- 6 B: BP size

Blocking factor of IL_2 = 34

1st level Index

Index Record size = 9+6 = 15B

P.I on Data file

IL_1 Blocking factor

$\left\lfloor \frac{512\,B}{15\,B} \right\rfloor = 34$

3rd level

IL_3 entries = 7

Blocking factor = 34 of IL_3

2nd level

IL_2 entries = 221

IL_2 Blocks

$\lceil \frac{221}{34} \rceil = 7$

1st level

IL_1 entries = 7500

IL_1 Blocks

$\lceil \frac{7500}{34} \rceil = 221$

- Total no. of Block's you need : 1+7+221 ⇒ 229
- Search on SSN: No cost: 3+1 (3 index level, 1 data block)

on SSN, we have PILI

④ file not ordered by key SSN want secondary index on these

① Index entry size: IL_1 = SI on key

1st level Index

9+7 = 16B

- 9: SSN
- 7: Record pointer

Record or Block pointer (By default)

2nd & higher levels

↳ Index Record size = 9+6 = 15

② Blocking factor of IL_1:

$\left\lfloor \frac{512\,B}{16\,B} \right\rfloor = 32$

& Blocking factor of IL_2 & higher level

$\left\lfloor \frac{512\,B}{15\,B} \right\rfloor = 34$

③ no. of 1st level index entries: #Record in file 30,000

S.I on key

IL_1 Block = $\lceil \frac{30,000}{32} \rceil = 938$

2nd level index entries = # Blocks in IL_1

= 938

IL_2 Block = $\lceil \frac{938}{34} \rceil = 28$

3rd level index entries = # Block on 2nd level.

= 28 single Block

level in MLI = 3

Index Blocks (total) = 1 + 28 + 938

= 967

④ Access Cost Search on SSN : 3 + 1 → Data Block (Index levels)

⇒ ④

⑤ file not ordered by the non-key Department_code → too much complicate will not be asked

S.I on there

Solution in file Org. & Indexing Lecture 9 : Multilevel Index & ISAM

ISAM → By Default: multilevel PI + sequential file ordered by key

(Indexed Sequential access method) ordered file

also → multilevel C-I + sequential file ordered by Non key

multilevel SI : 1st Level S.I

Remaining level : Sparse

↳ Kth level is P.I on K-1 level

PYQ Indexing → By go classes

ex: ISAM file - - - - → multilevel Primary Index

record size: 64B
Key size: 14B
address of disk block: 2B
block size: 512B
& there are 16K records

Blocking factor of file = #Records/Block $\Rightarrow \left\lfloor \frac{512}{64} \right\rfloor = 8$

#file Block $\Rightarrow \left\lceil \frac{16,000}{8} \right\rceil = 2000$ file Block

Index → multilevel primary index

Index entry size (whatever level of index) : 14+2B = 16B
(Key field size)

Index Block factor $\left\lfloor \frac{512B}{16B} \right\rfloor = 32$ index entries per Block

1st level Index:

no. of Index entries = # Block in file

= 2000

no. of Index level 1 (IL_1) Block = $\left\lceil \frac{2000}{32} \right\rceil = 63$ Index level 1 Block

2nd level Index

IL_2 entries = # Block in IL_1 = 63

IL_2 Block = $\left\lceil \frac{63}{32} \right\rceil = 2$

3rd level Index: 1 Block; 2 entries

Size of Data file: 2000 Blocks

Size of Index file: 1+2+63 = 66 Blocks

ex: 1GB data organized using Indexed-Sequential file → file with multilevel primary index

with Block factor: 8
B.S = 1KB
pointer: 32bit

- no. of level
- size of index at each level
- referencing capability per block of Index Storage may be considered to be 32] # Index entries = 32 (fan out per block)

Block factor : 8 Record/Block
Block size : 1KB

using these we can find Record size b/c it's exactly divisible else we could not find do

Block factor = $\left\lfloor \frac{\text{Block size}}{\text{Record size}} \right\rfloor$

b/c of the floor

Blocking factor : 8 records / Block → Record size = $\frac{1KB}{8} = 128B$

Block size : 1KB

(a) #Blocks = $\frac{1GB}{1KB} = 1M$B

1st level Index:

#Entries = 2^{20} = #file Blocks

Blocking factor of index = 32

#IL_1 Block = $\left\lceil \frac{2^{20}}{2^5} \right\rceil = 2^{15}$ Blocks

2nd level Index

#entries = 2^{15} = #IL_1 Blocks

#IL_2 Block = $\left\lceil \frac{2^{15}}{2^5} \right\rceil = 2^{10}$ Block

3rd level Index:

#entries = 2^{10} = #IL_2 Block

#IL_3 Block = $\left\lceil \frac{2^{10}}{2^5} \right\rceil = 2^5$ Block

in 4th level Index:

#entries = 2^5 = #IL_3 Block

#IL_4 Block = $\left\lceil \frac{2^5}{2^5} \right\rceil = 1$ Block

1st level	2nd level	3rd level	4th level
2^{15} Block	2^{10} Block	2^5 Block	1 Block

Total Index Size : $2^{15} + 2^{10} + 2^5 + 1$ Block

4 level

Gate 2008

ex. 16384 records

Record Size : 32B

key Size : 6B

B.S: 1024B

Bp: 10B

file is ordered on non-key & unspanned

& Secondary Index is built on the key field)

& multilevel index scheme is used to store secondary index

Index on Key field: SI on key

↓

Search key of Index

SI on key:

Index entry size (every level) : 6B+10B = 16B

Index Blocking factor (every level) = $\left\lfloor \frac{1024B}{16B} \right\rfloor = 64$

1st level Index

#IL_1 entries = 16384

↓

#Block IL_1 = $\left\lceil \frac{16384}{64} \right\rceil$ = 256 Block on level 1

2nd level index:

#IL_2 entries = 256

↓

#Block in IL_2 = $\left\lceil \frac{256}{64} \right\rceil$ = 4 Block on level 2

3rd level Index

#IL_3 entries = 4

↓

#IL_3 Block = $\left\lceil \frac{4}{64} \right\rceil$ = 1 Block on Level 3

- Ordered Indexes → we use them when data is not changing (also known as Static Indexes)
  - problem (Disadvantage)
    - modification is very time consuming (very hard)
      - Insertion
      - Deletion
      - modification

- for Dynamically changing Data file we need some "modification friendly" Index's (Dynamic Index)
  - (Tree based Indexing)
    - B+ tree Index (Better)
    - B tree Index

} they are disk structure (Hard drive)

ex: the in-memory Data structure are not suitable for Disk Data structure

In M·M, a node was a structure, having few Byte only

So, suitable for memory as we access M·M in word.

But Disk Access Block by Block } & Block Size: very large than M·M word size which will use a lot of space

- Disk search tree Data structure
  - B-tree
  - B+-tree

} Put a lot of Index entries in a single node (Block)

↑ Block

} Similarly to BST & AVL tree but diff. is that they use or take advantage of node size B/c they know node size is very huge

- Search tree
  - In-memory: BST, AVL, ~~Heap~~ doesn't tell where to go.

DISK: B tree
B+ tree

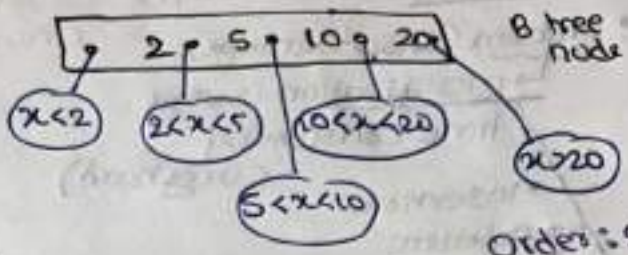
B tree
B+ tree

} A parameter → order of B/B+ tree should be taken care

max. no. of child pointer → for BST it will be 2

Order of B tree
max no of Tree
(Block & node)
Pointers per node

ex:



Order: 5 ptr.

Pointers: { B_p (Block pointer)
R_p (Record pointer)
$\Rightarrow R_p = B_p + offset$

Block ≡ Node

Block pointer ≡ node pointer
≡ Tree pointer
≡ Child pointer
≡ Index pointer

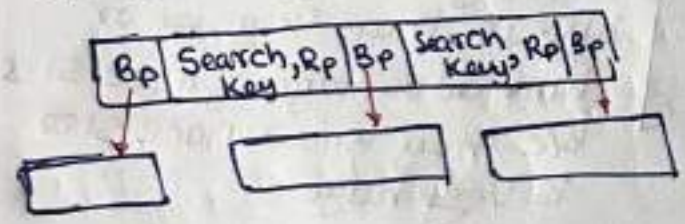
• B tree of order P
↳ max no of Block pointer in each node in any B tree

1. all leaves must be on same level

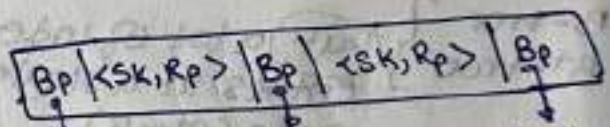
2. Space utilization Rule (≥50% utilization)
Root node: "2" B_p to "P" B_p
non-Root node: $\lceil \frac{P}{2} \rceil B_p$ to "P" B_p

• Every node have same structure

3. non-leaf node structure



4. Leaf node structure



null pointer (inactive) space will be wasted

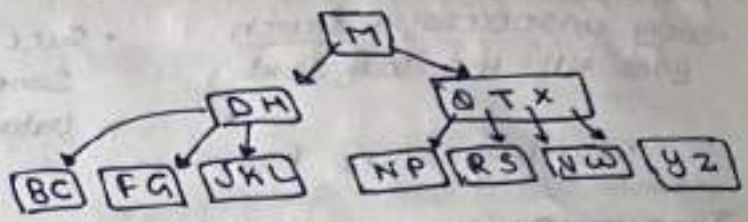
null pointer (but space will be there)

null pointer (inactive)

• B tree of order P
Root node: $2 B_p$ to 'P' B_p & 2-1 key to 'P-1' key
other node: $\lceil \frac{P}{2} \rceil B_p$ to 'P' B_p & $\lceil \frac{P}{2} \rceil - 1$ key to 'P-1' key

• within each node we have sorted

Ex: Order P of B tree
max. no of tree pointer/Node
then possible value of P



Root:
1 key to (P-1) key
max

$1 \le P-1 \Rightarrow (P \ge 2)$ no use

non-Root:
$\lceil \frac{P}{2} \rceil - 1$ key $\le \sim\sim \le (P-1)$ key

$\lceil \frac{P}{2} \rceil - 1 \le 2$ (min. no of node in non Root)

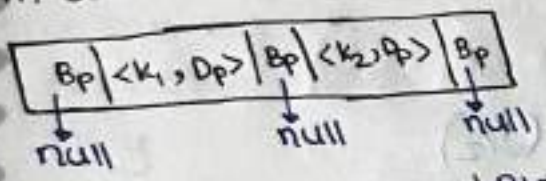
$\lceil \frac{P}{2} \rceil \le 3 \Rightarrow (\lceil P \rceil \le 6)$

$4 \le P \le 6$

$3 \le P-1 \Rightarrow (P \ge 4)$
max. no of nodes a non-root node

I/o cost to search "J" is (5)

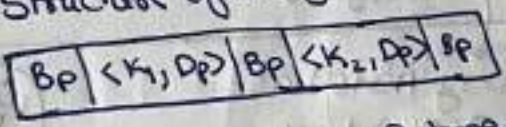
I/o cost for Data Record having Search key value "J" is 3+1=(4)
In Btree



Dp (Data pointer) or pointer { Block pointer (or) Record pointer }

Ex: B.S: 4096B
Key: 4B
Bp: 8B
no header info.
find order of Btree
max. no of tree pointer
↳ P

Structure of Every node



node: In one B-tree node
{ max. Bp: P
max <key, Dp> pairs = P-1 }

$\Rightarrow P(8) + (P-1)(4+8) \le 4096$

$\Rightarrow 8P + 12P - 12 \le 4096$

$P \le \frac{4108}{20} \Rightarrow P \le 205.4$

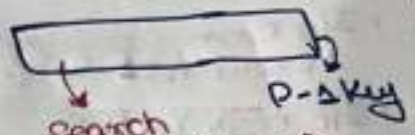
So, $\boxed{P = 205}$

In B tree

- every unsuccessful search goes till the leaf level
- Successful search can stop as soon as we find search key value in any node of B tree

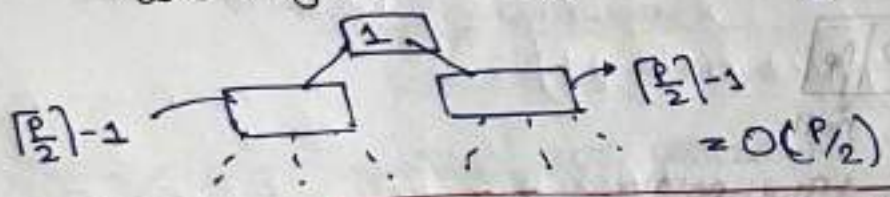
Searching is like BST

- Time complexity of search: worst case ⇒ O(height * ( ))
  - worst case → go till leaf level
  - ( ) → within node → O(P-1) = O(P)
- search within Node → can do Binary search



B tree ⇒ 'n' key

↳ w.c height (max height): $O(\log_{P/2} n)$



So, Time compln :- $O(\log_{P/2} n * \log_2 P/2)$

- $\log_{P/2} n$ → max. height
- $\log_2 P/2$ → with node Binary search

rewritten as

$O(\log_{P/2} n * \log_2 P/2)$

$O\left(\frac{\log_2 n}{\log_2 P/2} * \log_2 P/2\right) \Rightarrow O(\log_2 n)$

every node has O(P/2) key

'n' key → to get max. height

↳ put min. no. of keys in every node

⇒ O(P/2)

Search time comp.

'n' keys

- AVL tree ---- $O(\log_2 n)$ --- more no. of levels
- B tree ---- $O(\log_2 n)$ → no. of level's less But within node search take more time

• Insertion is always done at leaf node in B tree

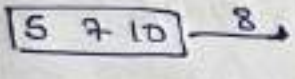
• Insertion in Btree: always start at leaf

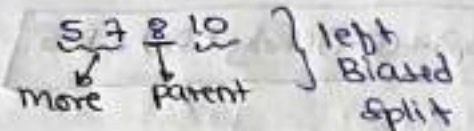
① leaf not full: just put it in sorted ordered

② leaf full But parent not full
↳ Split leaf & put middle element in parent

③ Leaf full & parent also full
Split leaf & put middle element in parent
Split parent also

Btree order is Even:
Order 4: #key = 1 to 3



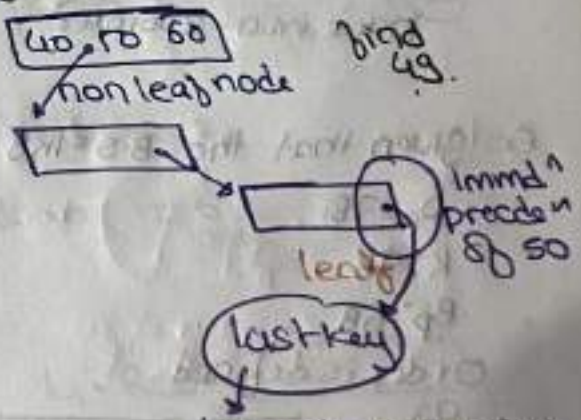


• Always stick to left or right Biased
But only one of them consistently

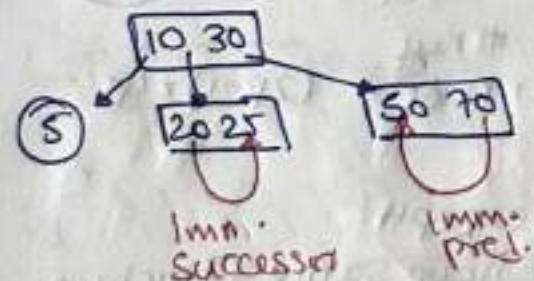
• we can get different B tree of different height for the same set of key, But using different order of insertion of key

ex: A Btree has single node (only the root node)
is this Alone root node a leaf node (or) Internal?
b/c All Bp are null

ex: Immediate predecessor (or) successor of every key of non-leaf is always in a leaf
True



ex: Immediate successor of every key leaf is always in a non-key
false



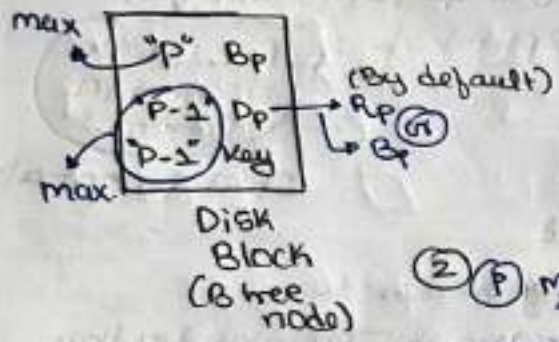
ex Relationship blw no. of leaf node & no. of key in non-leaf node
} Proof by finding immediate predessor

no. of leaf node = 1 + no. of key in non-leaf node

- for every key of non-leaf node we have unique immediate predecessor leaf node

#leaf node = # key in non-leaf nodes + 1 (Right most leaf node)

Find the order: ① (By default) (P) max. no. of Bp in a node (block)



$P(B_p) + (P-1)(Key + D_p) \leq B \cdot S$

② (P) max. no. of key in a node

$P(K + D_p) + (P+1)B_p \leq B \cdot S$

③ each B-tree node, except root has atleast P & atmost 2p tree pointers
If tree is non-empty then root can have min. 2 pointer.

$2p(B_p) + (2p-1)(K + D_p) \leq B \cdot S$

ex: given that the B.S = 1KB

D_p = 7B
K = 8B
B_p = 6B
Order is defined as 3rd point

① $2P(6) + (2P-1)(9+7) \leq 1024$

$12P + 32P - 16 \leq 1024$

$P \leq \frac{1040}{44} = 23$ (P=23)

max #key per node = $(2P-1) = 45$

② in these quest^n if leaf node Bp null value are removed for use?

- all the node except root node have same structure
- all the tree pointer in leaf are null

Variant of B tree

then find order.

So, now

max. #key per leaf node > max. # key in internal node

- Since leaf node required no pointer to children, they could conceivably store a different (larger) no. of key than internal node for the same disk page size

↳ here order of a leaf node of B tree in the max. no. of key that can be sorted in the leaf node

leaf node structure in prev. variant of B tree

leaf node

$P(9+7) \leq 1024$

$P \leq \frac{1024}{16} = 64$

order of leaf node

ex: 2KB ⇒ B·S

12 B ⇒ pointer

56 B ⇒ block header

Key: 8B

Build a index on key

max. no. of record we can index with a 3level B tree.

max. #key per block : y

$y(8+12) + (y+1)(12) \leq 2048 - 56$

$y = 61$; order = 62

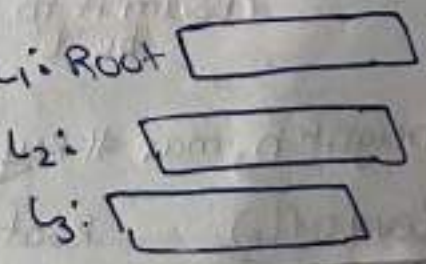
min. #key = $\lceil \frac{p}{2} \rceil - 1 \Rightarrow \lceil \frac{62}{2} \rceil - 1 = 30$

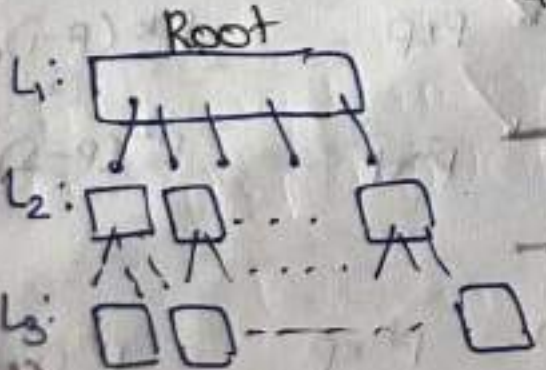
Root node : 1 key to 61 keys

non-Root node : 30 key to 61 key

max. no. of key we can store in B tree of 3level

fill every node as much as possible





#nodes	#Keys	#Bp
1	61	62
62	62*61	62*62
62*62	62*62*61	62*62*62

All will be null pointer

(leaf)

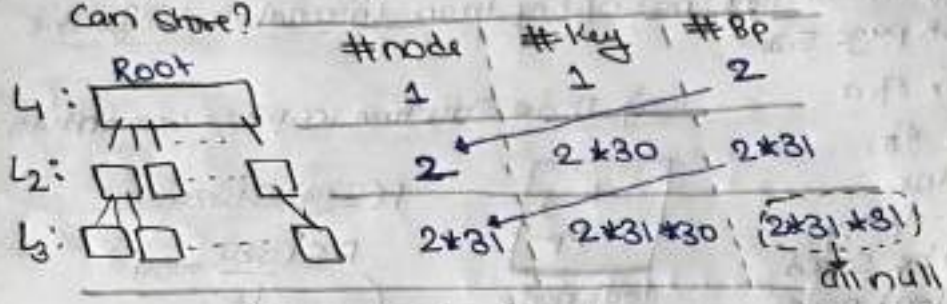
max no of keys in 3 levels:

$61 + (62*61) + (62*62*61) = 238328$ key

ex: Order of B tree : 62

3 level ; min # key use

Can store?



min #key in 3-level

$1 + 60 + 60 \times 31 \rightarrow (1921)$

Question type

ex. given height

max no of keys

nodes etc - that

can be stored

Height of B tree : #levels - 1

↳ # edge from Root to any leaf

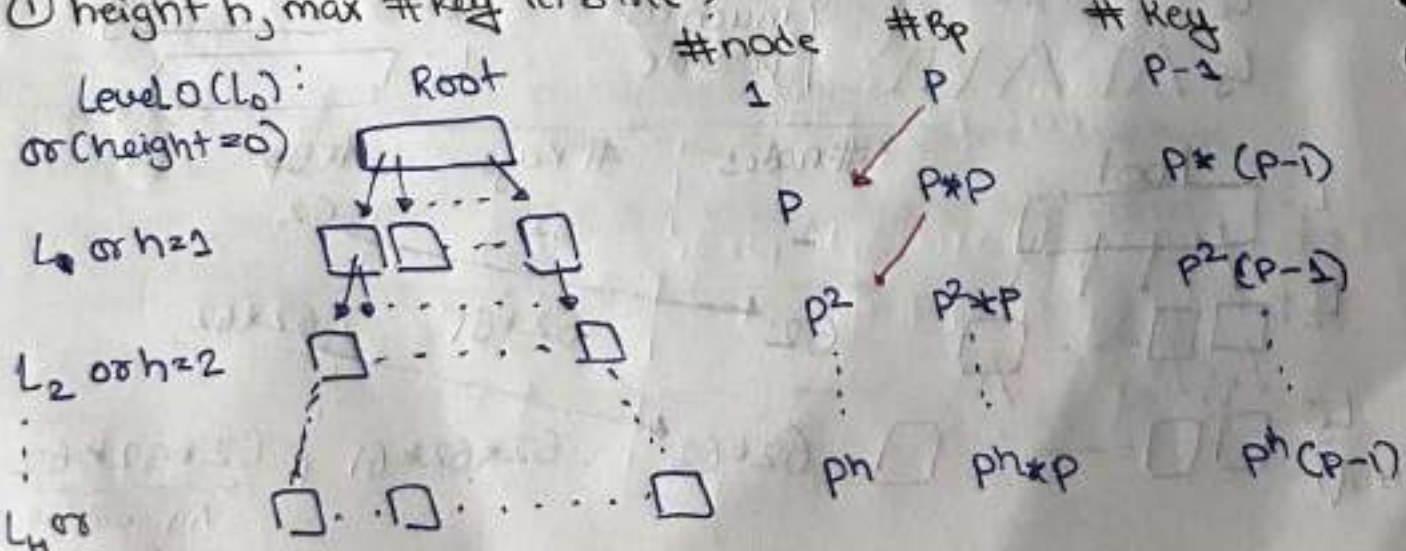
B tree order p:

Root: 2 Bp to 'p' Bp | 1 key to "p-1" key

non-Root: $\lceil \frac{p}{2} \rceil$ Bp to 'p' Bp | $\lceil \frac{p}{2} \rceil - 1$ key to "p-1" key

assum it to be "t"

① height h, max # key in B tree?



① height h, max # key in B tree?

$\Rightarrow (p-1) + (p-1)p + (p-1)p^2 + \dots + (p-1)p^h$

$\Rightarrow (p-1)[1 + p + p^2 + \dots + p^h]$

$\Rightarrow (p-1)\left[\frac{p^{h+1}-1}{p-1}\right] \Rightarrow \boxed{p^{h+1}-1}$

max. no of key for height 'h'

② max. no. of node

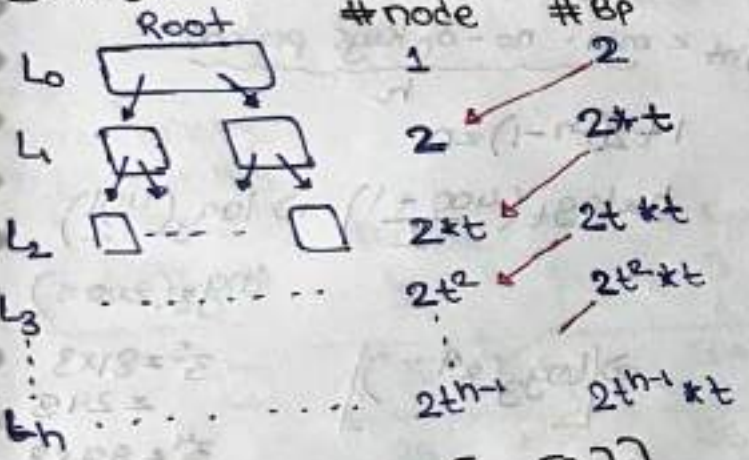
$= 1 + p + p^2 + \dots + p^h$

$= \boxed{\frac{p^{h+1}-1}{p-1}}$

ex: max. data entries B tree of order 5 with height of 3

using formula $\Rightarrow 5^{3+1}-1$

② height h, min # key in B tree.



	# node	# BP	# Keys
L_0	1	2	1
L_1	2	$2*t$	$2(t-1)$
L_2	$2*t$	$2t*t$	$2t*(t-1)$
L_3	$2t^2$	$2t^2*t$	$2t^2*(t-1)$
L_h	$2t^{h-1}$	$2t^{h-1}*t$	$2t^{h-1}*(t-1)$

① height h, order p $\left[t = \lceil \frac{p}{2} \rceil\right]$

min # key

$\Rightarrow 1 + 2(t-1) + 2t(t-1) + \dots + 2t^{h-1}(t-1)$

$\Rightarrow 1 + 2(t-1)\left[1 + t + \dots + t^{h-1}\right] \Rightarrow 1 + 2(t-1) * \frac{t^h - 1}{t-1}$

$\Rightarrow \boxed{1 + 2(t^h - 1)}$

ii) min. no. of node

$\Rightarrow 1 + 2 + 2t + 2t^2 + \dots + 2t^{h-1}$

$\Rightarrow 1 + 2\left[1 + t + t^2 + \dots + t^{h-1}\right]$

$\Rightarrow 1 + 2\left(\frac{t^h - 1}{t-1}\right)$

Question type: no. of search key (Given) & min. & max. height that B tree can have [Reverse Quest^n]

#keys = n ; order p

max. height ≡ min. no of keys per node. (height = h)

$1 + 2(t^h - 1) = n$

$t^h = \left(\frac{n-1}{2} + 1\right)$

$h = \left\lfloor \log_t\left(\frac{n-1}{2} + 1\right) \right\rfloor$

min. height ≡ max. no. of key per node (h)

$p^{h+1} - 1 = n$

$\Rightarrow h = \lceil \log_p(n+1) \rceil - 1$

ex: B tree with order p (max. no. of child pointer) the max. level of index required to store 400 distinct key in order p=5

max height = min. no. of keys per node (h)

$1 + 2(t^h - 1) = n$

$h = \log_t\left(\frac{400+1}{2}\right) \Rightarrow \log_3\left(\frac{401}{2}\right)$

$\log_3(200.5)$

$3^5 = 81 \times 3 = 243$

$3^4 = 27 \times 3$

$\Rightarrow \lfloor \log_3(3^{4\cdots}) \rfloor$

$\Rightarrow 4\cdots$

max. height = h = 4

& level = 5

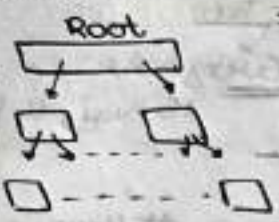
#keys = 400 ; p = 5 Root: 1key to 4keys

non-Root: 2key to 4keys

level = height + 1

$t = \lceil p/2 \rceil = \lceil 5/2 \rceil = 3$

• max. level



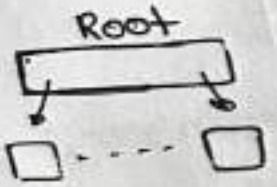
#Node	#Bp	#key
1	2	1
2	2*3 = 6	2*2=4
6	6*3 = 18	6*2=12
18	18*3=54	18*2=36
54	54*3=162	54*2=108
162	162*3	162*2=324

till level 5: #key : 108+36+12+4+1 = 161 (min. no.) node to be inserted

till level 6: #key: 161+324 = 485 (min. no.)

#keys we have : 400 ⇒ { 5 level (or) 6 level

• min. level than we will try to fill max. key in a node



#node	#Bp	#Keys
1	5	4
5	5*5=25	5*4
25	25*5=125	25*4
125	125*5	125*4

till level 3: #key = 124 (max. node that you can insert)

till level 4: #key = 624

#key we have 400 ⇒ level 3 ✗ (or) level 4 ✓

ex. Search field is a non-ordering key field

$p = 23$

each node is 69% full

no. of keys stored in Level 4 (height 3)

$\Rightarrow 23 \times 0.69 = (16 B_p)$ per node

of the total we are able to use 69% only

B-Tree

Every node: 15 keys, 16 B_p

	#node	#B_p	#Keys
(Root) L_1	1	16	15
L_2	16	16×16	$16 * 15$
L_3	16^2	$16^2 \times 16$	$16^2 * 15$
L_4	16^3	$16^3 * 16$	$16^3 * 15$

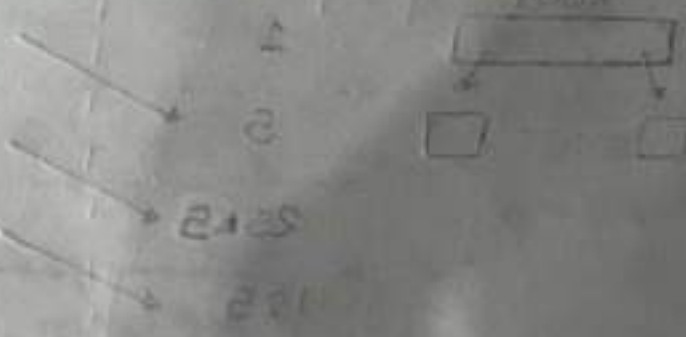
$\Rightarrow 15 + 16*15 + 16^2*15 + 16^3*15$

$\Rightarrow 15(1 + 16 + 16^2 + 16^3)$

$15 \times \frac{16^{3+1} - 1}{16 - 1} \Rightarrow \frac{16^4 - 1}{15}$

$\Rightarrow$ 65,535

Total keys



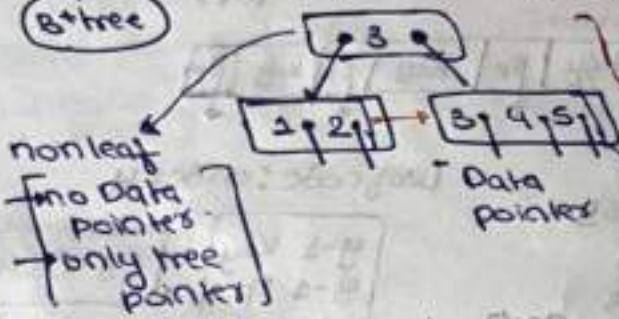
B+ tree
→ every datais present on leaf

Some Data are present in non-leaf
(unique key but still duplication is there)

→ every leaf point to the next leaf node

Search keys: 1,2,3,4,5 order: 5

B+tree



non leaf
→ no Data pointer
→ only tree pointer

Same value can be store if more than one node

• Data ptr
• only one tree ptr that point to next leaf

• In non-leaf node, no duplication will happen

In B+ tree

#Leaf node = 1 + #key in non-leaf nodes
→ for B tree
→ for B+ tree

Order of B+tree → max. no of total ptr. per node

Root → 2 Bp to 'p' Bp

non-Root & non-leaf node → $\lceil p/2 \rceil$ to 'p' Bp

Leaf node → #Data pointer = #Keys

$\#keys = \left\lceil \frac{p-1}{2} \right\rceil$ to p-1 keys

→ 1 tree pointer means 1 Bp

• B+tree order p

Root: 2 Bp to 'p' Bp
1 key to (p-1) key

Leaf: $\left\lceil \frac{p-1}{2} \right\rceil + 1$ to p total ptr
↳ Tree ptr

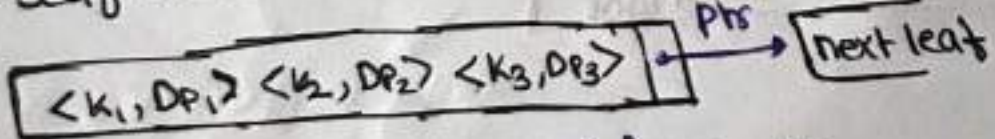
$\left\lceil \frac{p-1}{2} \right\rceil$ key to (p-1) key

give more balance structure

Remaining Internal node: $\lceil p/2 \rceil$ Bp to "p" Bp

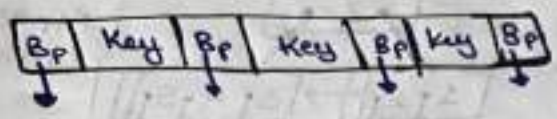
$\lceil p/2 \rceil - 1$ key to (p-1) key

• Leaf node Structure



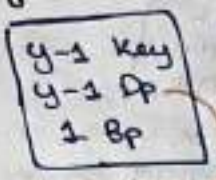
1 Bp → if 'q' keys then 'q' Data ptr.

• Internal node structure: no Data ptr.



ex: Suppose
B.S: 4096B
Key: 4B
pointer: 8B
no header info.
Order of B+ tree
max. no. of pointer per B+ tree node

Leaf node: order y

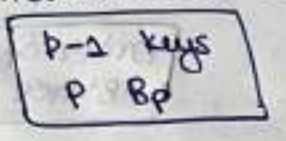


$(y-1)(4+8)+8 \leq 4096$

y=p

if Dp become Bp then structure become same as internal node

Internal node: order p

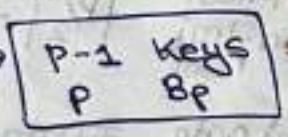


$(p-1)(4)+p(8) \leq 4096$

note

1. if Data pointer is Block ptr:

leaf :
Internal:



⇒ order of leaf node = order of internal node

⇒ order of B+ tree

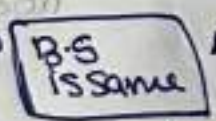
2. if Data ptr is not Block ptr.
↳ Record ptr.

(Record ptr size > Block ptr size)

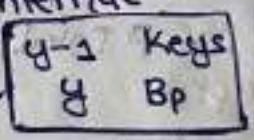
note

Root is Leaf iff only one node in the tree
Otherwise; Root is Internal node

Leaf:



Internal



more key

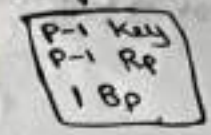
P<Y

order of leaf node ≠ order of internal node

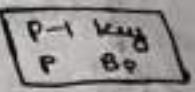
3. If B+ tree has order P:
means All nodes have order P & Rp > Bp then which Block are not fully utilized? ⇒ (Internal Block)

} bc we are fixing the order

Leaf



Internal



Order of every node = 3

① fixed Block Size } every thing here i am fixing

not fixing order and asking order? → but u can't

then order may come out to be different

order of leaf } ≠ { order of internal

② fixed order } fixing the order

then definitly some space will be coasted

- here you can't ask order of B+ tree you can ask order of leaf or order of Internal

ex: Rp: 9B
B.S: 4096B
Key: 4B
Bp: 8B
no header info.
find order of leaf node, order of Internal node.

Leaf: order y

$(y-1)(4+9)+8 \leq 4096 \Rightarrow y = 314.61\ldots$

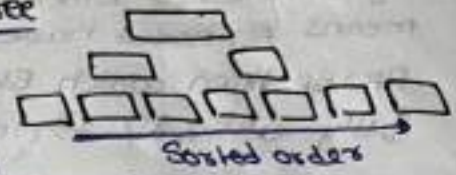
$y = 314$ order of leaf

Internal: order P

$P(8)+(P-1)(4) \leq 4096 \Rightarrow P = 341$

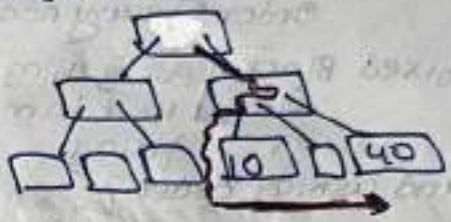
order of Internal

In (B tree)
Search stop anywhere from Root to leaf
↳ O(h)
X Θ(h)

In B+ tree


Search : O(h)
↳ always go till leaf

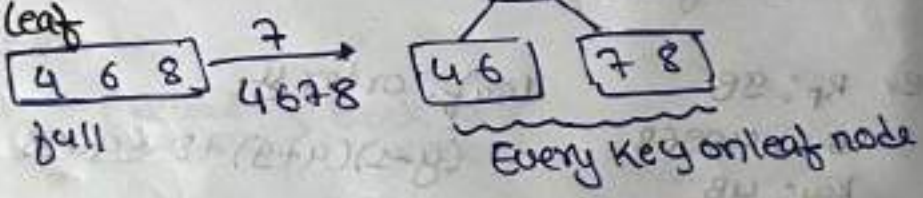
• 2 type of search in B+ tree
① Point search
(Equality search)
Index ← I/O cost : #levels for equality search

② Range search
10 ≤ K ≤ 40


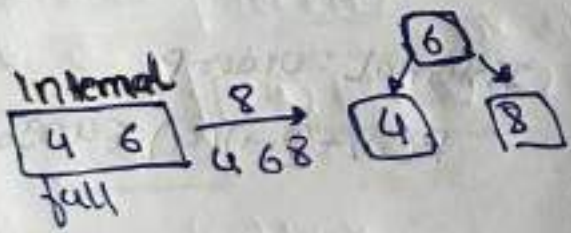
• Insertion in B+ tree → Insertion is same as B tree Except when we split leaf node

Split of Internal node: Same as B tree

Split of leaf node:
Order: 4
max. # Key = 3



Internal node
max Keys : 2



if

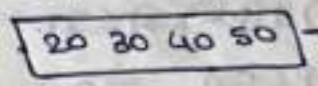

(Left Biased)
(Right Biased)

B tree Insertion ≡ B+ tree insertion
↳ only Difference is when leaf split

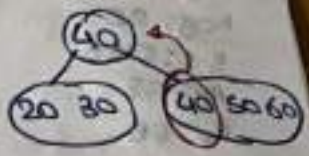
Observation

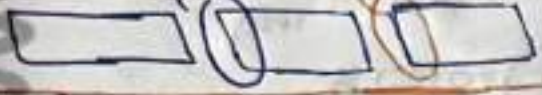
① Right Biased B+ tree

leaf node (full)



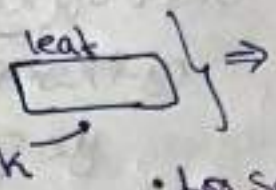
60





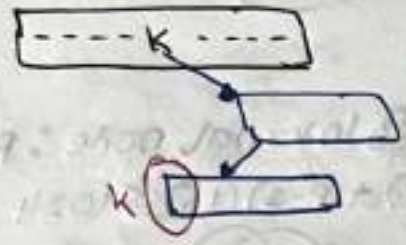
#leaf node = #keys in non-leaf node + 1

• in which condition key 'k' can go to non-leaf node?



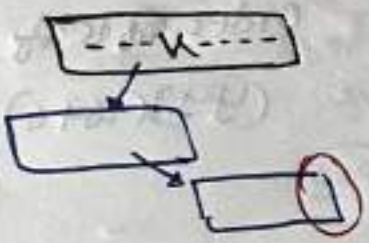
⇒

• for some split k must be the splitting point



Key k of non-leaf node is left most key in Immediate Successor leaf node

② Left Biased B+ tree



- So, B+ tree: Key 'k' is in non-leaf node iff k must have been splitting point of some leaf split.

- Among non-leaf node, no key repeats. (no duplication)

- you can get Different B+ tree of Different height for the same set of key for different order

ex: B+ tree
Key: 9B
B.S: 512B
R_p: 7B
B_p: 6B

Internal node have

P tree ptr
P-1 key

these must fit into a single block

• Order of Internal node: (P)

$P(6) + (P-1)(9) \leq 512$

$P \leq \frac{524}{15} \Rightarrow P \leq 34.93$

$P = 34$

• Order of leaf node = order y

$(y-1)(9+7)+6 \leq 512$

$y \leq \frac{522}{16}$

$y = 32$

$R_p > B_p$

b/c R_p size is more we can't fit more R_p in leaf so order of internal will be more than leaf

ex: Key: 16B
R_p: 5B
B_p: 8B
B.S: 1024B
max. value of order of internal & leaf node

$R_p < B_p$ never happen (ideally) But for these ques. we are considering

Order of Internal node: P

$P(8) + (P-1)(16) \leq 1024$

$P = 43$

order of leaf node (y > P)

$(y-1)(16+6)+8 \leq 1024$

$y = 46$

gate 2002
ex:
Search key on Student table (name)

Student name length: 8B } key

B.S: 512B

Index ptr: 4B } also known as B_p

degree of B+ tree
(no. of ptr. per node)

here $D_p \equiv B_p$ are same

order of B+ tree

Order (P):

internal node point of view (POV)

$P(4) + (P-1)(8) \leq 512$

leaf node

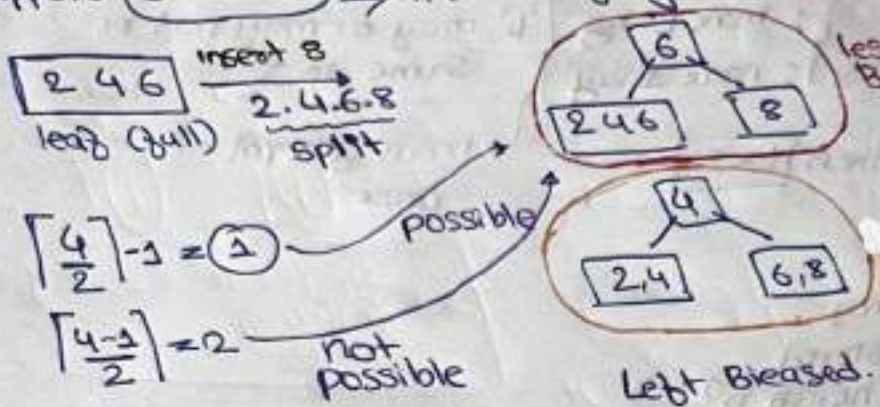
$(P-1)(8+4)+4 \leq 512$

both will give same ans.

$P = 43$

why $\lceil \frac{p-1}{2} \rceil$ but not $\lceil \frac{p}{2} \rceil - 1$?

Suppose (order : 4) ⇒ max. no of key : 3



$\lceil \frac{p-1}{2} \rceil$ why only applied to leaf.

b/c only the leaf split cause's key Repetition.

as $Dp \leq Bp$ we can use any formula we will get same ans.

ex: K : 8B

B.S : 2KB

Block header : 56B

Pointer : 12B

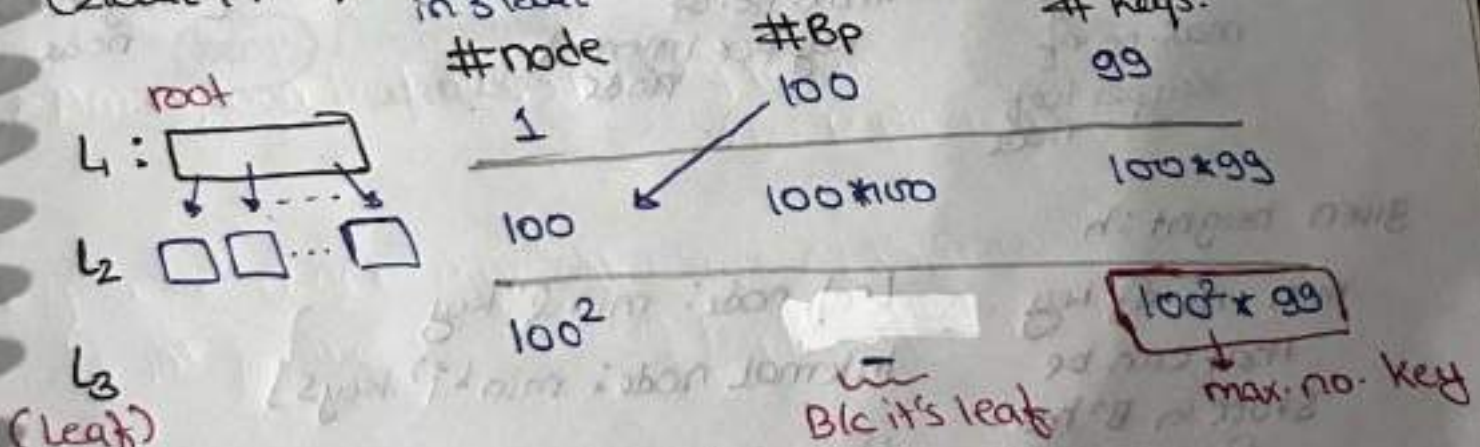
• max no of key in a node (leaf & internal)

$y(8) + (y+1)(12) \leq 2048 - 56$

$y = 99$

max. no. of record we can index with a 3 level B+ tree (2 level + root) | max. no. of key that can be stored in 3 level

max. no. of key in a node : 99



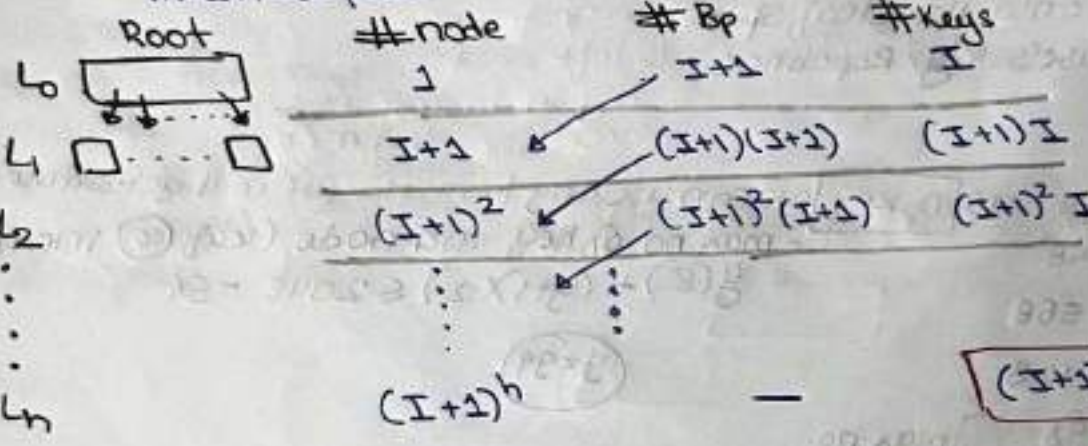
Quest^n type (given height) i.e for given height, max. no of key in node

leaf node : max L key
Internal node : max I key
'L' may or may not be same as I
↳ more general case

if $B_p = D_p$ then $L = I$

height h:
max. no of key that can be stored in B^+ tree of heigh h?

Idea: fill every node as much as possible



max. #keys: $L(I+1)^h$
↳ height
↳ max. no. of key per leaf node
↳ max. no. of key per Internal node

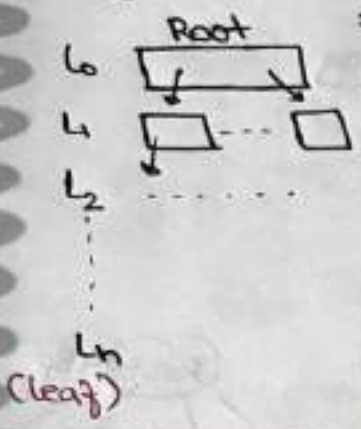
given height: h

min. no of key that can be store in B^+ tree of height h?

Leaf node: min 'l' key
Internal node: min '?' keys

idea: fill every node as less as possible

min no of node



	#node	#Bp	#Keys
L0	1	2	1
L1	2	2*(i+1)	2*i
L2	2(i+1)	2(i+1)(i+1)	2(i+1)i
Lh	$2(i+1)^{h-1}$	—	$2(i+1)^{h-1}(l)$

$2(i+1)^{h-1}(l)$ → min no node

min no. of key $\Rightarrow$ $2(i+1)^{h-1}(l)$

- $(i+1)^{h-1}$ → min #key per internal
- (l) → min. #key per leaf
- h → height

- Question type
  no. of search key } given
  find min (OR) max height of B+ tree

→ Bulk Loading B+ tree Concept can be used → Bottom up Approach

↓ when Key's are already given

Bulk loading (Bottom up Insertion)
Simultaneously Insertion

- B+ tree
- heap (in-memory Data Structure)

Ex: Create a B+ tree of order 5 for the following set of key with min-height?

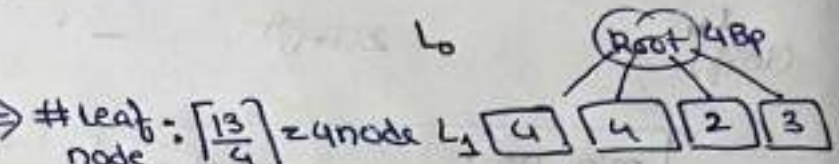
{32, 24, 6, 7, 11, 8, 22, 4, 5, 16, 19, 20, 78}

Keys: (13 keys) ⇒ Bulk loading

Leaf: 2 to 4 Keys

Bottom up: (min. height)
max. key per node

leaf level → (15 keys) ⇒ # leaf node: $\lceil \frac{13}{4} \rceil$ = 4 node

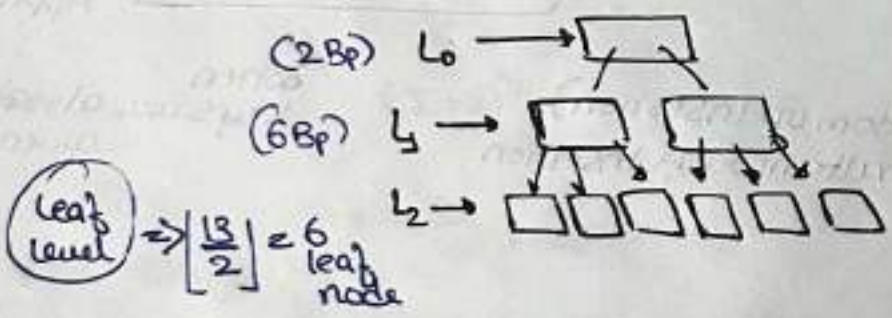


In Internal node: 3 to 5 ptr

So, min. height = 1 & #level = 2 } #nodes = 5 //

If max. height:
fill every node as less as possible

So, max height = 2
#level: 3
#node: 6+2+1 ⇒ 9



(1)

- B/B+ tree which will take less no. of level & node. [BS is same]

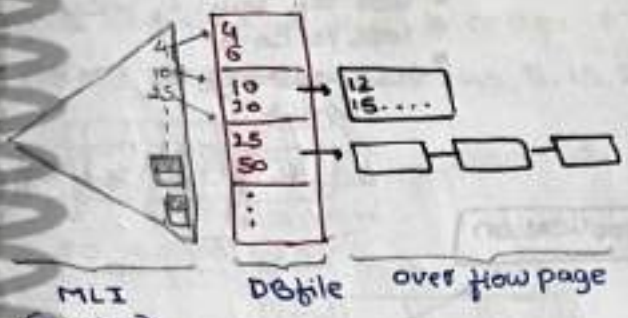
b/c in internal node
B+ tree we can put more child ptr } So, B+ tree will have less height
b/c we save space of Record ptr

(2) [if order of B/B+ tree is same] ⟶ In one Block how many Rp you can put, you are fixing
So, B+ tree will have disadvantage

So, B+ tree will take more level & B tree will take less no Duplicate key

b/c of Duplicate key

■ Static MLI:-



k-level WC I/O cost : O(n)

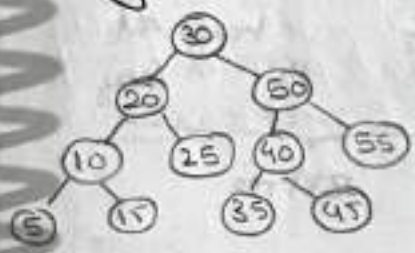
■ Dynamic MLI:-
- NO overflow pages.
- Every index block must atleast 50% occupied (except the last block)

Standard Data Structure (DS):

B tree, B+ tree } Dynamic MLI

{Balanced search tree}

■ Binary search tree:



Search: 35

30 50 40 35

Probe sequence (to access 35 node access at each level) to search 35

■ WC # Comp(TC) to search key in BST of n distinct key: θ(n)

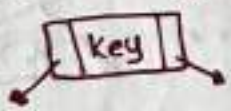
■ WC # Comp(TC) to search key in Balanced search tree of n distinct key: θ(logn)

■ Balanced search tree
(Height restricted search time)
max height of search time should not exceed O(logn) for n distinct key

Balanced search tree

AVL tree Red-Black tree B tree B+ tree

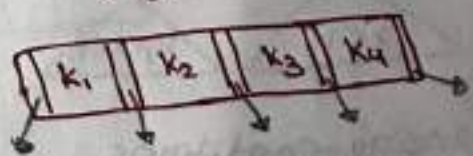
← Balanced binary search tree →



degree 2

← Balanced search trees →

degree P:4

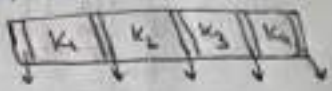


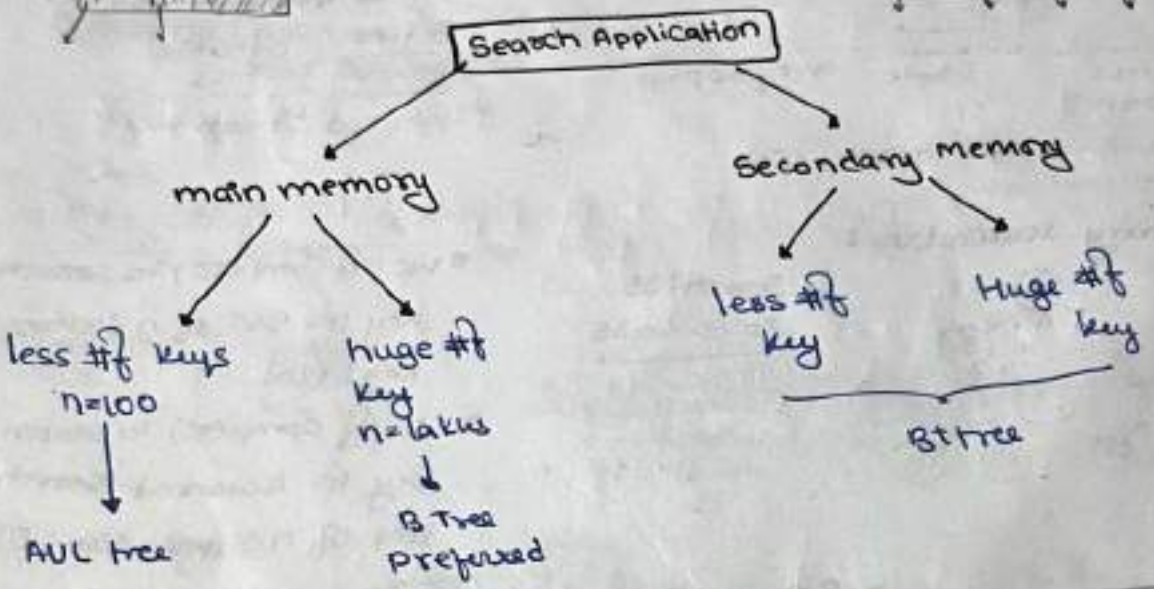
Ques] why B/B+ tree used for DB index rather than Balanced binary search tree?

1. Index must be maintained in Disk b/c DB file in Disk
2. data Access from Disk to MM is block by block

[3] If AVL Tree used for DB index
- more levels of index
- more I/O cost
- wastage of space disk

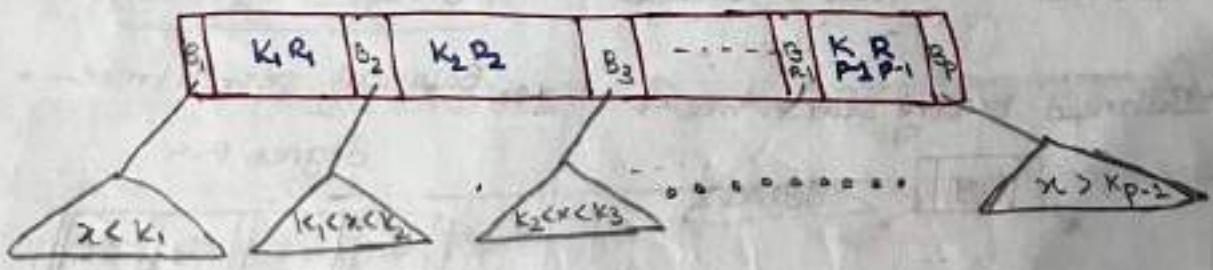
[4] If B/B+ tree used for
- less # level of index
- less I/O cost
- less wastage of disk space





B Tree Order p: max tree ptr (block ptr) per B tree node

[1] Node Structure :- P block ptr, (P-1) Keys, (P-1) Records ptr.



- Balancing conditions

[2] Every Internal root except Root
min: $\lceil p/2 \rceil$ block ptr & $\lceil p/2 \rceil - 1$ key
max: p block ptr & (p-1) key

[3] Root Node
min: 2 block ptr & 1 key
max: p block ptr & (p-1) key

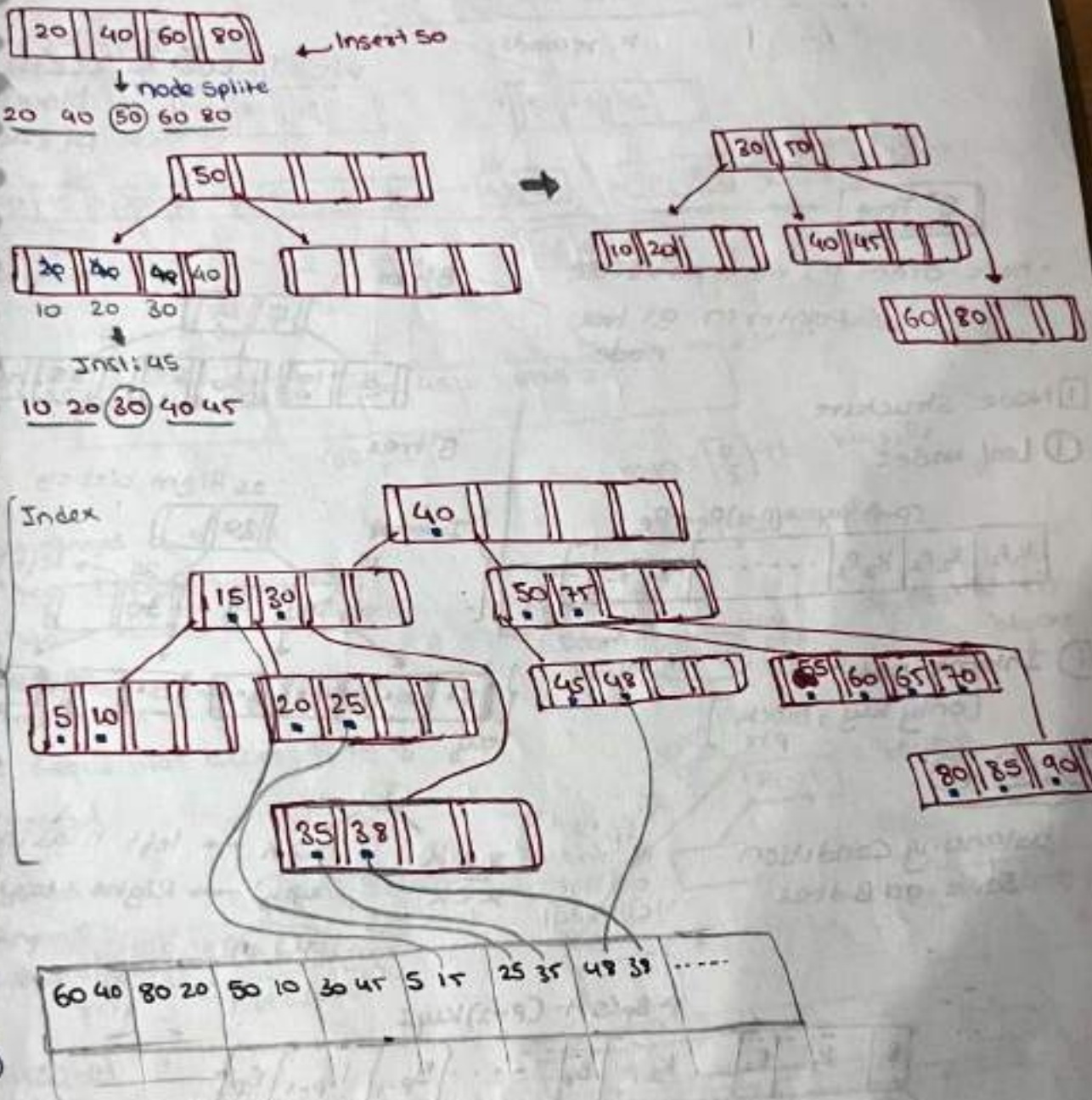
[4] Every leaf node must be at same level
leaf node $\lceil \frac{p}{2} \rceil - 1$ to (p-1) key

- min 50% node should be occupied, other than Root.

Construction of B tree

Give sequence of key & order p:5 [max tree ptr]

60, 40, 80, 20, 50, 10, 30, 45, 5, 15, 25, 35, 48, 38, 65, 75, 85, 55, 70, 90, 95.



- Advantage of B tree
  - B tree index best suitable for random access of any one record

i.e Select *
From R
where, (A = 48) any one record

k : no. of record

wc : I/O cost : k+1 blocks
Bc : I/O cost : 2 block

- disadvantage
  - B tree index not best suitable for sequential access of range of record

• Dis Adv :-

```
Select *
From R
where (A ≥ 45 and A ≤ 85);
```

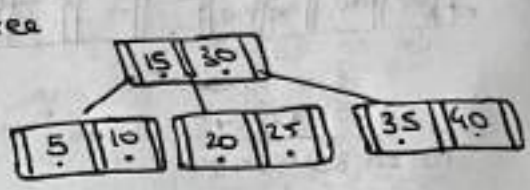
Sequencial access of Range of Record.

x records.

We I/O cost ≈ x(K+1) blocks
≈ θ(x·K)

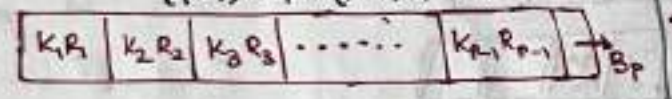
B⁺ Tree

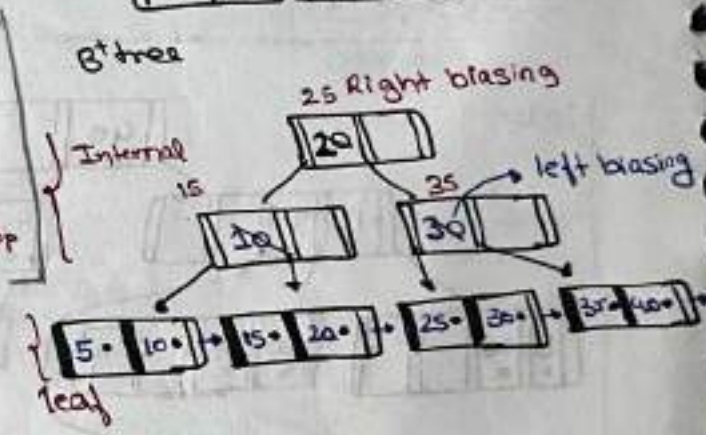
• here order p : max possible pointers in B⁺ tree node



1. Node structure

(i) Leaf Node:

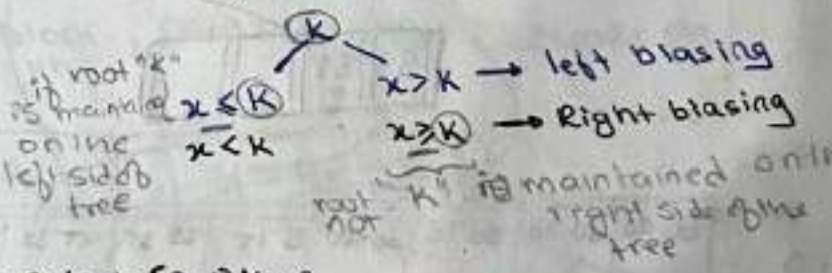


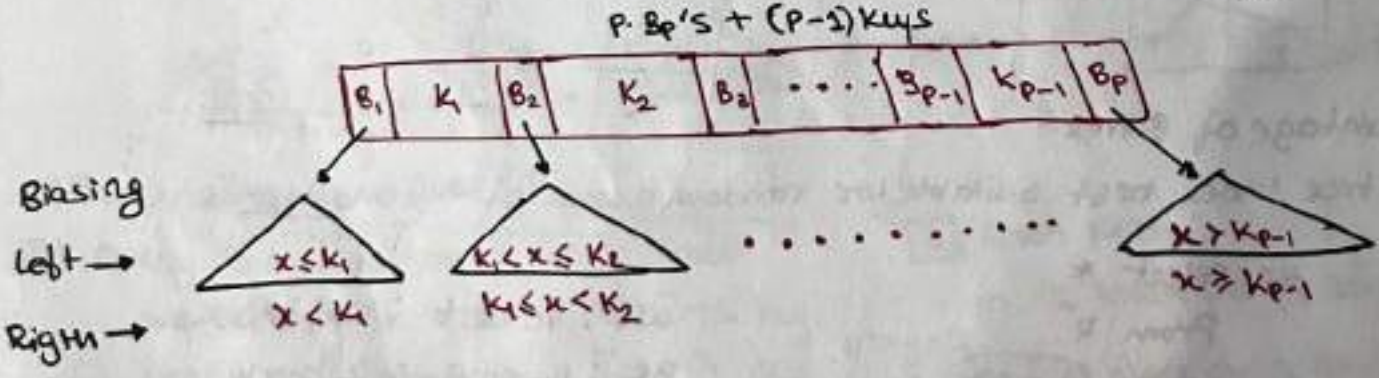


(ii) Internal node:
[only Key, Block ptr]

(2), (3), (4)

balancing condition
Same as B tree

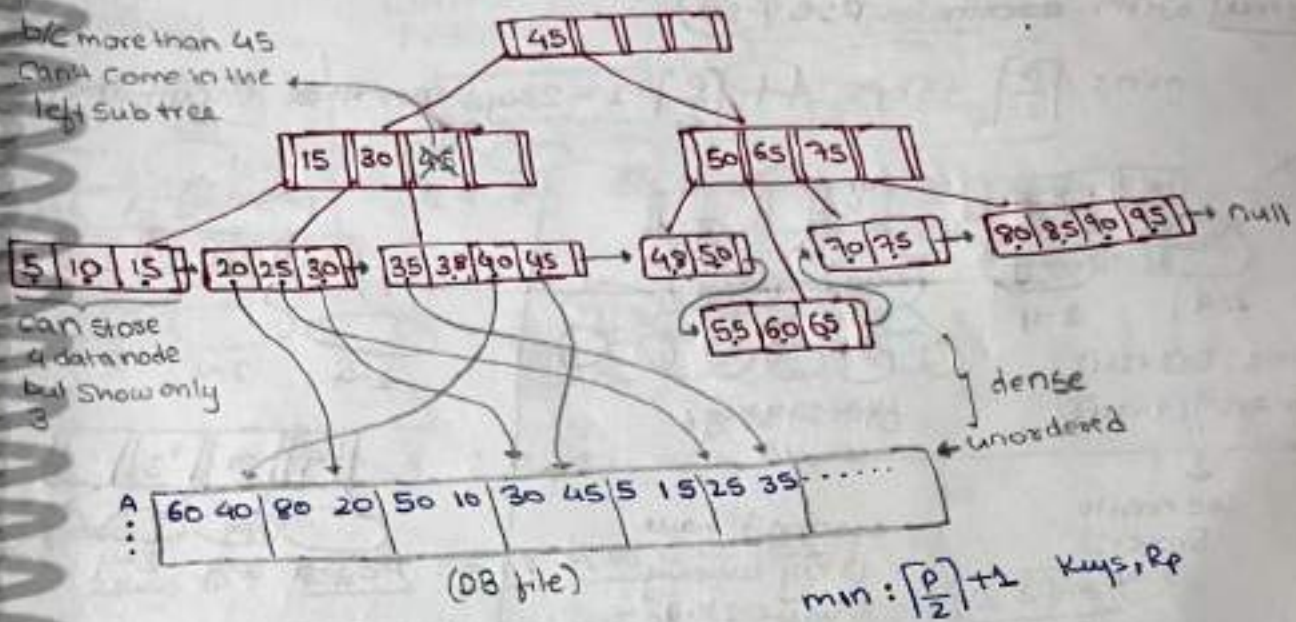




So, all record pointer only at leaf level.
& in Internal node only keys & Blockptr

Construct B+ tree of order P:5 [max ptr per node] & sequence keys

60, 40, 80, 20, 50, 10, 30, 45, 5, 15, 25, 35, 48, 38, 68, 75, 85, 55, 70, 90, 95



min: $\lceil \frac{p}{2} \rceil + 1$ keys, Rp

max: P-1 keys, Rp

Advantages of B+ tree

- B+ tree is best suitable for random access of anyone record
- B+ tree index also Best suitable for sequential access of range of record.

Select * From R where A=45 (any one record)

I/O cost = (K+1) blocks

Select *
From R
where A≥40 & A≤60

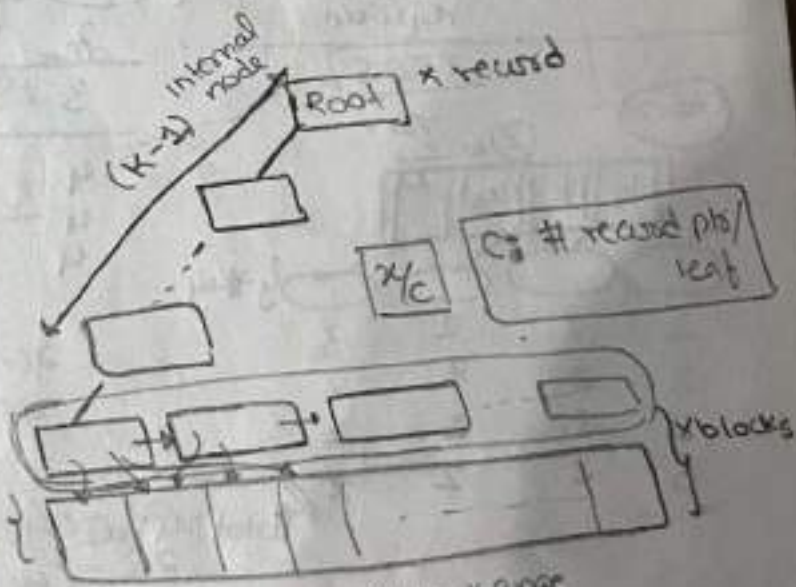
Range of record

x records

Estimated cost :-

$$\text{I/O cost} = \sum (k-1) + \frac{x}{c} + x$$

$$= \theta(k+x)$$



↳ as all record are present in the leaf so we first go to the beginning range element which can cost (K-1) where K is no of level

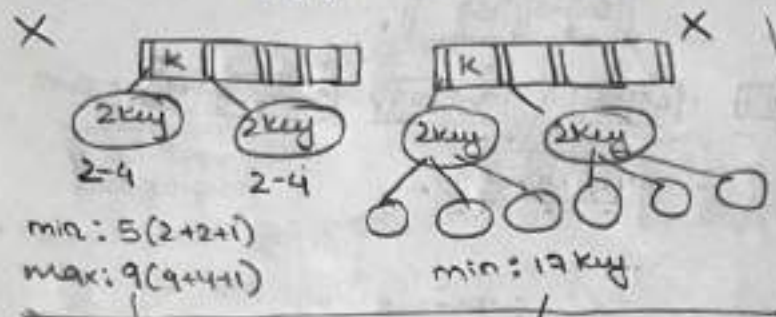
↳ then finding the end of range sequentially

① Problem: How many possible B/B+ trees for given ~~#~~ key & order p?

Ques] How many B tree possible for 12 distinct key

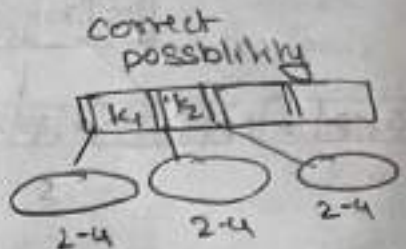
B tree] with ...order p: 5 ? (10)

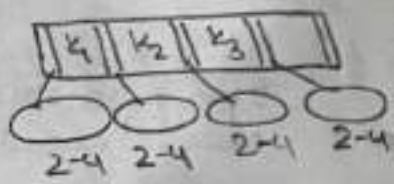
min: $\lceil \frac{p}{2} \rceil$ = 3 bps & $\lceil \frac{p}{2} \rceil - 1$ = 2 keys per node except root.



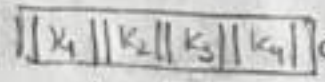
we need to store 12 key

we don't have 17 key we only have 12 key





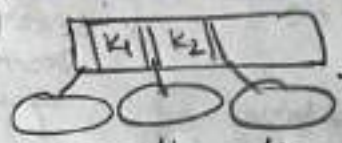
for same reason



can be taken b/c min: min: 14 key required

1 2 3 4 5 6 7 8 9 10 11 12

Case 1



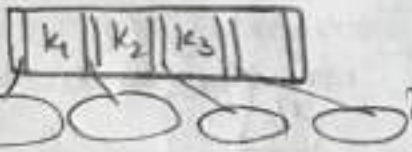
10 keys

2 4 4

3	3/4	4/3
4	2	4
4	3	3
4	4	2

// Total 6 Tree

Case 2



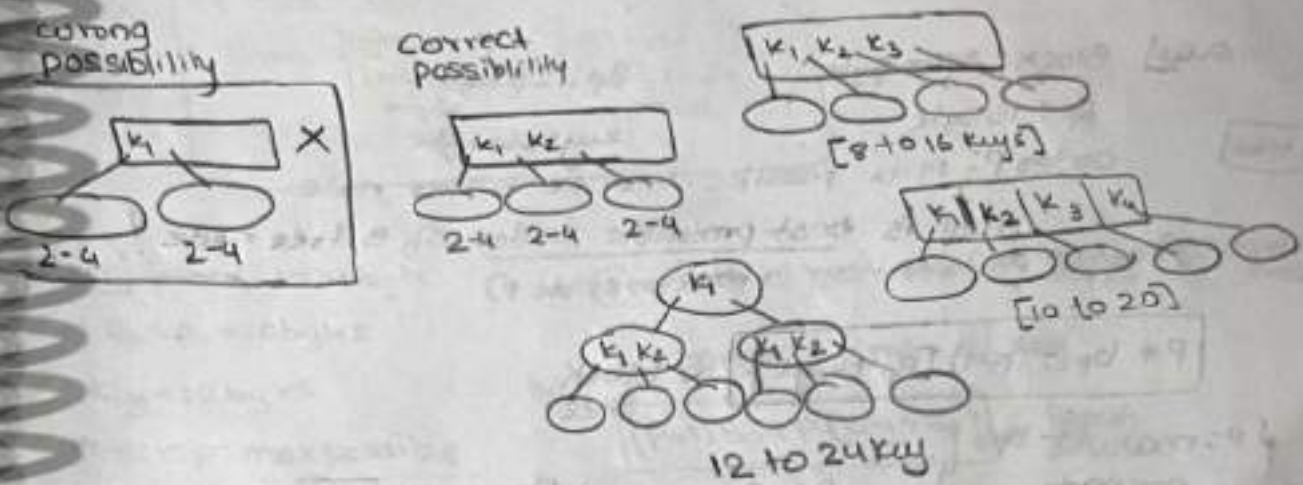
9 key

2	2	2	3
2	2	3	2
2	3	2	2
3	2	2	2

// Total: 4 Tree

Total 10 Tree

B⁺ tree min keys : $\lceil p/2 \rceil - 1$ = 2 keys } except root
max keys : p − 1 = 4 keys



2 levels : →

2 keys IN : 1234 5678 9 10 11 12 = 1

3 keys IN : 2 2 4 4 $^4C_2$ = 6
3 3 3 3 = 1
2 3 3 4 = 12

4 keys : 2 2 2 2 4 = 5!/4! = 5
2 2 2 3 3 = 5!/3!2! = 10

3 level : ..... = 1

Total : 36

& left & right bias ⇒ 36 + 36 ⇒ 72

2334
4!/2! = 12
2244
4!/2!2! = 6
3333
4!/4! = 1
19

Assign
key : 17
order p : 5
How many possible B Tree?

3 key IN :- 10
4 key IN :- 30
5 key IN : 1
3 level IN : 1
Total : 42

⇒ left & right bias = 42 + 42 = 84

② Problem (model) : Find best possible order of B/B+ tree node for given Block size, key size, Bp, Rp size?

Ques] Block: 2048 Byte Bp: 25 byte

Rp: 15 Byte key: 20 byte

B tree

order P: max possible bps in B Tree node

what is best possible order of B tree node?
(max possible P)

$$P * b_p + (P-1)[R_p + key] \leq \text{Block Size}$$

{P: max # bp's per node}

$P * 25 + (P-1)[15+20] \leq 2048$

$60P \leq 2083$

$P = \left\lfloor \frac{2083}{60} \right\rfloor = 34$

B+ tree

{order P: max possible (pointers) per B+ tree node}

what is best possible order of B+ tree

(i) Internal node?

P: max ptrs per node
[# of Block ptrs]

$P * Bp + (P-1) key \leq \text{Block Size}$

$P * 25 + (P-1)20 \leq 2048$

$45P \leq 2068$

$P = \left\lfloor \frac{2068}{45} \right\rfloor = 45$ Ans

(ii) leaf node?

$(P-1)[key + R_P] + Bp \leq \text{Block Size}$

$(P-1)[20+15] + 25 \leq 2048$

$35P \leq 2058$

$P = \left\lfloor \frac{2058}{35} \right\rfloor = 58$ max. ptrs.

Note

If Size(R_p) = Size(B_p)

Then { B+ tree internal node order } = { B+ tree leaf node order }

Ques] Block: 1024 byte

$R_p = B_p = 10$ bytes

Key = 12 bytes

Order P: max possible key per node.

(i) what is best possible order of B tree node?

Order P: max # key

$$(P+1)B_p + P[key + R_p] \leq \text{Block Size}$$

$$(P+1)10 + P[12+10] \leq 1024$$

$$P = 31$$

max # key

(ii) what is best possible order of B+ tree node?

Internal node :- order P: # key

$$(P+1)B_p + P * key \leq \text{Block}$$

$P = 46$

leaf node: P: max # keys

$$P[K + B_p] + 1\,B_p \leq \text{Block}$$

$P = 46$

- If Block size of B tree = Block size of B+ tree

Then ① { B tree node order (P) } < { B+ node order (P) }

B+ tree preferred for DB file index rather than B tree index

② { # B tree nodes for n distinct key } > { # B+ tree nodes for n distinct keys. (for large value of n }

③ # B tree level for n distinct key > # B+ tree levels for n distinct key (for large value of n)

④ [1/o cost] > [1/o cost]

- B+ tree index
  - less I/o cost for Random access of any record b/c of less levels
  - less I/o cost for sequential access of range of record.

• B^+ tree Index Access, Hops is less B tree than random access

• if B Tree order p = B^+ tree order p

then

① { # B tree node for n distinct key } < { # B^+ tree node for n distinct key }

② { # B tree level for n distinct key } < { # B^+ tree level for n distinct key }

Btree

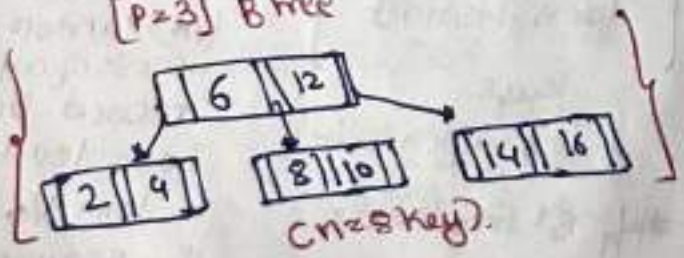
- Structure of leaf Node & Internal node same.
- Record ptr are in both leaf node & Internal node.
- key store in both Internal node & leaf node

B^+ tree

- leaf node structure & Internal node Structure not same
- Internal node consist only block ptr & keys. leaf node Consist (key, record pointer) pair & one block pointer.
- all the keys which are used for Index must be at leaf level each leaf node pointed next leaf node.
- Complete Tree must be left biased/ right biased.

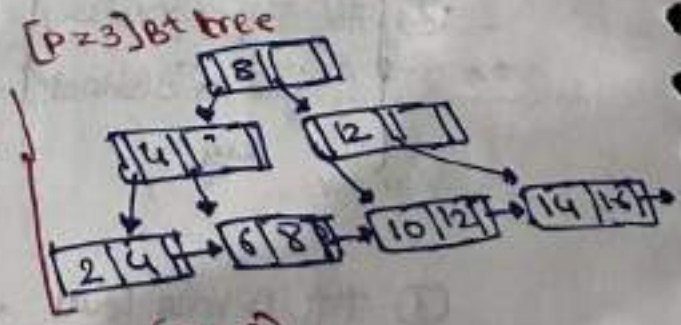
if order of B & B^+ is same.

[p=3] B tree



(n=8 key).

[p=3] B^+ tree



(n=8)

So, it's access cost is more than B tree when orders is same.

Bulkloading B+ tree

[Bottom up construction of B+ tree]

- Sort keys used B+ tree node in ASC order
- Leaf nodes:- distribute (key) into leaf node
- Internal node:- If m node at level l Then m block pointers [tree pointer] at level-1 distribute (m block ptr) into nodes [# nodes at level l-1] nodes

Qus] # key 44 order p is given find min/max level of B+ tree? And min/max node of B+ tree?

Qus] Keys: 1, 2, 3 ... ; 45

Order P: 5 [max ptr in B+ tree node]

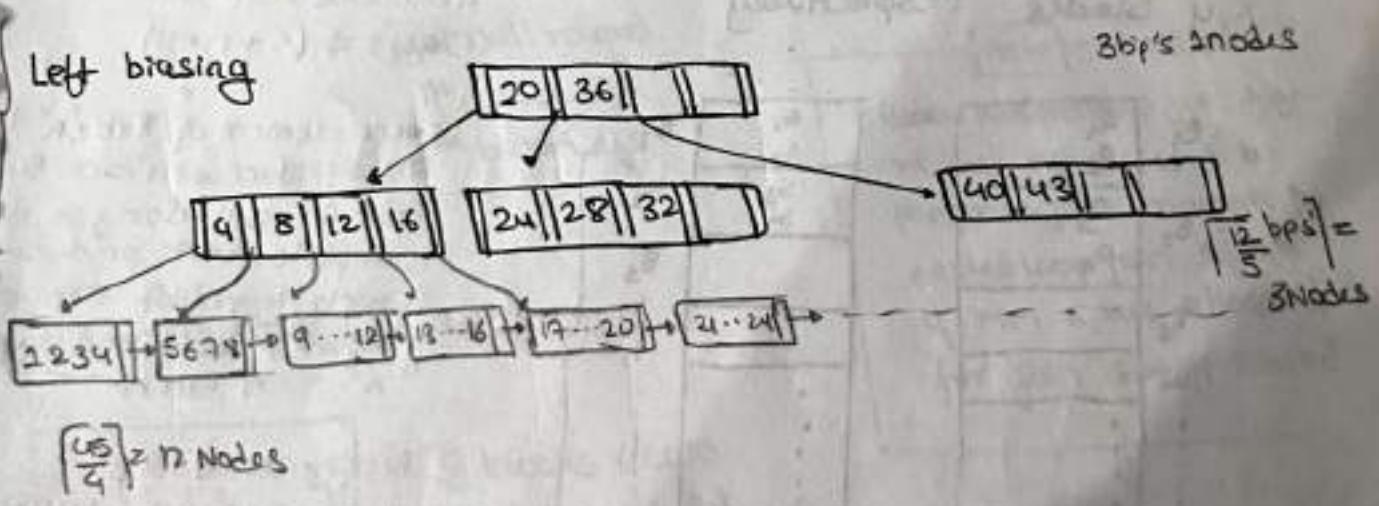
How many min level of B+ tree index?

Min level : 3

Min node : 12+3+1 = 16 nodes

Min level : All each node as most as possible

Max: 5 ptrs. 4 keys per nodes

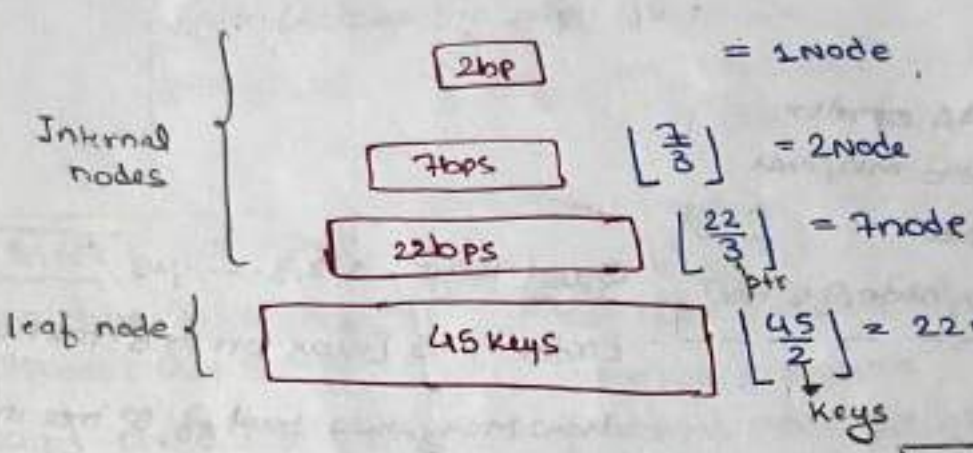


Ques] Keys : 1,2,3 45
Order p: 5
How many max level of B+ tree?

Ans: min : 4 levels
max: 22+7+2+1 = 32 nodes

Max level : Fill Node as least possible

min ptrs : $\lceil \frac{p}{2} \rceil = 3$ & 2 keys for all node except root.

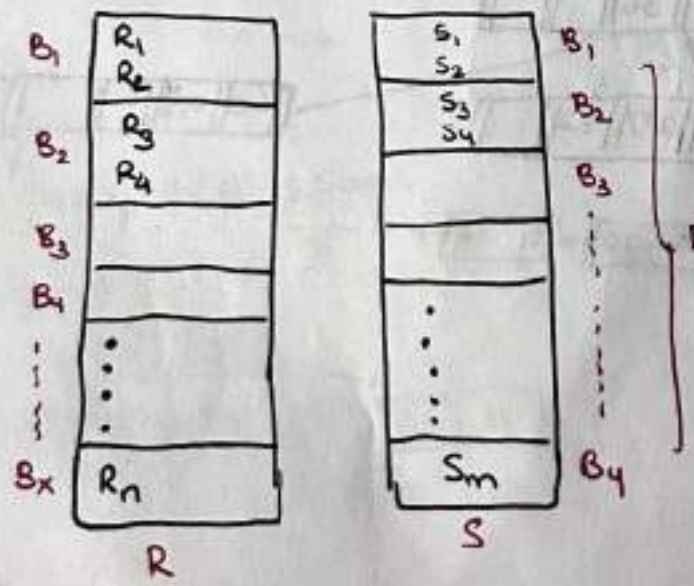


note
$\lceil \frac{45}{2} \rceil$ = 23 nodes
22+2+1 ~~23rd node~~
~~1 key~~

Join Algo

1. Nested loop join
2. Block Nested loop join

R,S relation with n, m records occupied in x, y blocks respectively



In nested loop join

① RMS ⇒ $(x + n*y)$

every element of Table "R" is cross product with each Block of Table "S". will doing so if we complete cross product of each element of a Block than count it as 1, and it for "x" such Block

② Block nested loop

RMS ⇒ $(x + x*y)$: same logic as above but the only difference comes is that instead of every element here we multiple it with every Block of Table "R".

Milan Marwaha

1 Nestedloop join :-

record num of outer Rel

```
R⋈S : for(i=1; i≤n; i++){
        for(j=1; j≤m; j++){
          Comp(Ri, Sj)
        }
      }
```

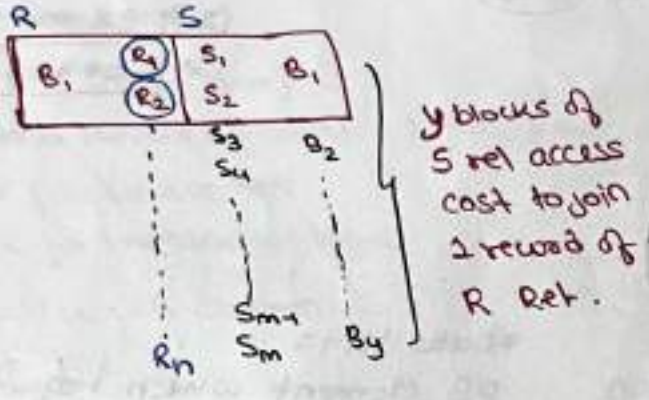
record num of inner Rel

Ri : ith Record of R

Sj : jth record of S

Assume: only 2 blocks of MM allocated for Join
[1 more for store the value before copying in disk]
[Limited MM space allocation for join]

(i) R⋈S Access cost : $(x + n*y)$ blocks



y blocks of S rel access cost to join 1 record of R Rel.

(ii) S⋈R Access cost: $(y + m*x)$ blocks

2 Block Nested loop join :-

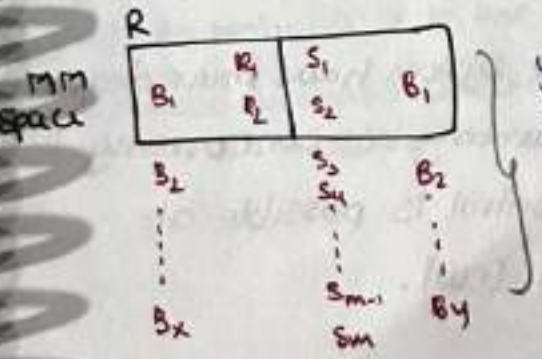
Assume: only two block of MM allocated for join

```
R⋈S: for(i=1; i≤x; i++){
       for(j=1; j≤y; j++){
         join all record
         ith block of R
         jth block of S
       }
     }
```

note

- Block nestedloop join Algo reduce Access cost to perform join compare to nested loop join Algo if MM space allocated for join is very limited

(i) R⋈S Access cost $(x + x*y)$ blocks



y blocks access cost from S Rel to perform join of 1 block of record of R

(ii) $(y + y*x)$ blocks

Pg: 152 Q. 18 & 19

T_1: 2000 record – 80 blocks

T_2: 400 record – 20 blocks

(18) nested loop join
[most approp. order]

$T_1 \bowtie T_2$ = 80 + 2000 * 20 = 40080 block

$T_2 \bowtie T_1$ = 20 + 400 * 80 = 32020 block

(19) Block nested loop join:-
[most app. order]

$T_1 \bowtie T_2$ = 80 + 80 * 20 = 1680

$T_2 \bowtie T_1$ = 20 + 20 * 80 = 1620

So. no of reduction is → 32020 − 1620
→ 30400

in OS means process
& in DBMS, means Transaction

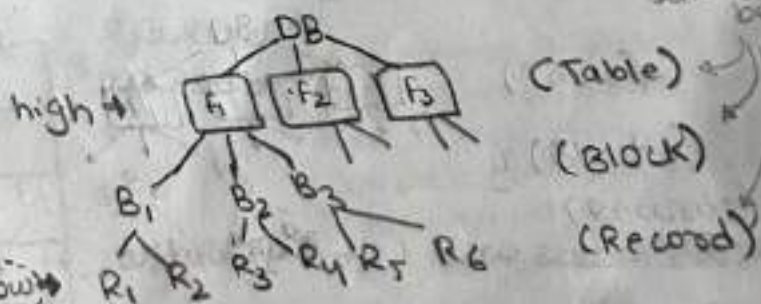
Transaction & Concurrency Control

- Transaction:
  set of logic related operation to perform some task
  [unit of work]

- Data item:
  DB element which required for many trans [shared resource]

- Concurrency control
  method to avoid inconsistency b/c concurrent execution of many transaction over same Database

- Concurrent Execution
  Simultaneous execution of many transaction over same database



- Degree of concurrency
- Number of user that can use database simultaneous is know as Degree of concurrency

- Degree of concurrency is very less using flat file system b/c OS concurrency control is over file level [file is treated as resource]

- DBMS file system have more degree of concurrency b/c concurrency level control is possible at record level.

- degree of concurrency

  Record level > block level > file level > DB level

- Complexity (# of locks) of locks

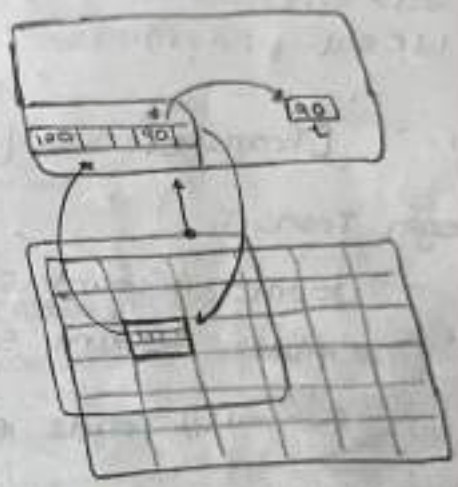
  Record level > block level > file level > DB level.

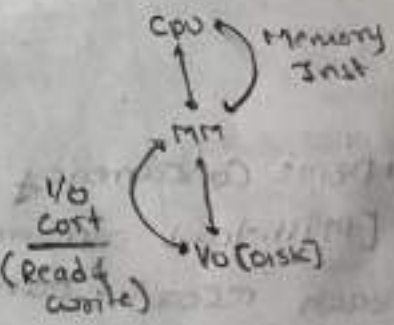
- DBMS uses multilevel concurrency control so that it can achieve more degree of concurrency & less complexity of lock management

- Main operation:

- Read(x): [x is data item]
  access data item(x) from database file to use the value of x for transaction logic
  1. Find block which contain x in Database file
  2. transfer Block to MMemory
  3. find value of data item x value in MMemory block & copy value of x in program variable

  (all together 1,2,3 process are consider as atomic)





- write(x):
  updation of data item x value in database file
  1. Find block which contain x from DB file
  2. transfer block to MMemory
  3. update value of x in the MMemory block
  4. Replace updated block of MMemory into DB file

  (Atomic)

Trans[T]	Trans[T]
R(x)	W(x)
x=x-10	
W(x)	

- Update R set sal=10000 where eid=e;

Trans (T_1)
write (sal=10000 of eid e)

- Update R set Sal=Sal+2000 where eid=e;

Trans (T_2)
R(sal of eid=e)
Sal = Sal+2000
W(sal of eid=e)

Read (A)
A=A-100
write(A)

$\equiv$

Read(A,t) — Put A in t
A=A-100
write(A,t) — put t in A

[Transfer 500 from bal Aid 101 to 102]

```
begin Trans T1
    update Account set bal=bal-5000 where aid=101;
    update Account set bal= bal+5000 where aid=102
    Commit// Trans executed succ.
end trans T1 ;
```

⇓

Sql parser.

- DBMS Concurrency control
  [Multi-level concurrency control]
- each record can be treated as.
  One Resource.
- more degree of concurrency.
- Too complex to manage locks.
- here in DBMS one resource can be
  - one Record or one Block, or
  a file

ACID properties

- to preserve integrity (correctness)
- transaction must obey ACID Rule

A: Atomicity } maintained by recovery mgmt component of DBMS s/w
D: Durability }

I: Isolation } maintained by concurrency control component of DBMS s/w
C: consistency } maintained by user (DBA, DB developers, end users)

■ Atomicity

- execute all operation of the transaction or execute none of the operation of the transaction

i.e All [Commit] / None

i.e
T_i: transfer 500 Rs from A to B

Trans (T_i)
R(A)
A = A − 500
W(A)
R(B)
B = B + 500
W(B)
Commit [state]
Redo operation
(Atomicity failed)

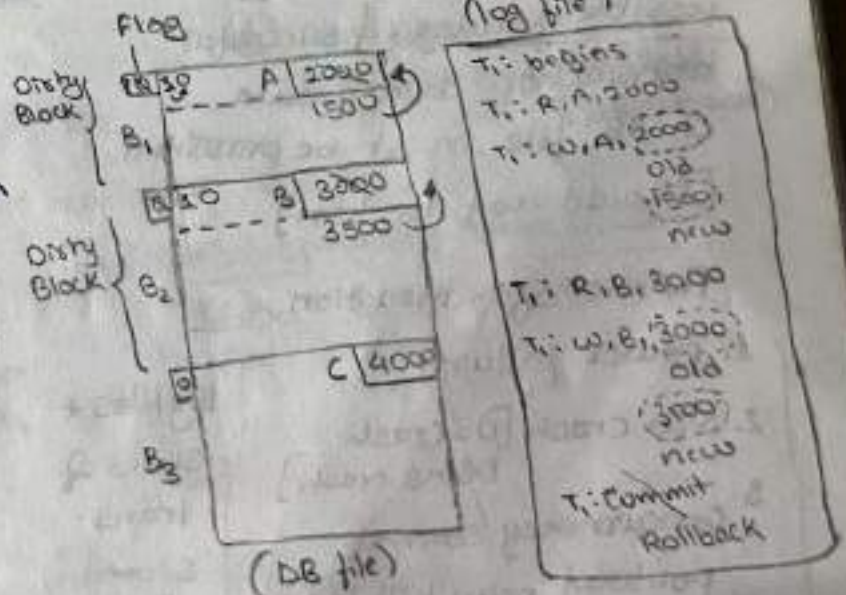
× fail
Rollback (undo operation)

■ Recovery managment Component

- Transaction must be Rollback if transaction fail before commit

● Rollback [Abort] [state]
- undo all transaction of the failed transaction

● Transaction log
- file maintained by recovery manager in disk which is used for recovery of transaction failure.
- all activity of transaction are record in log file

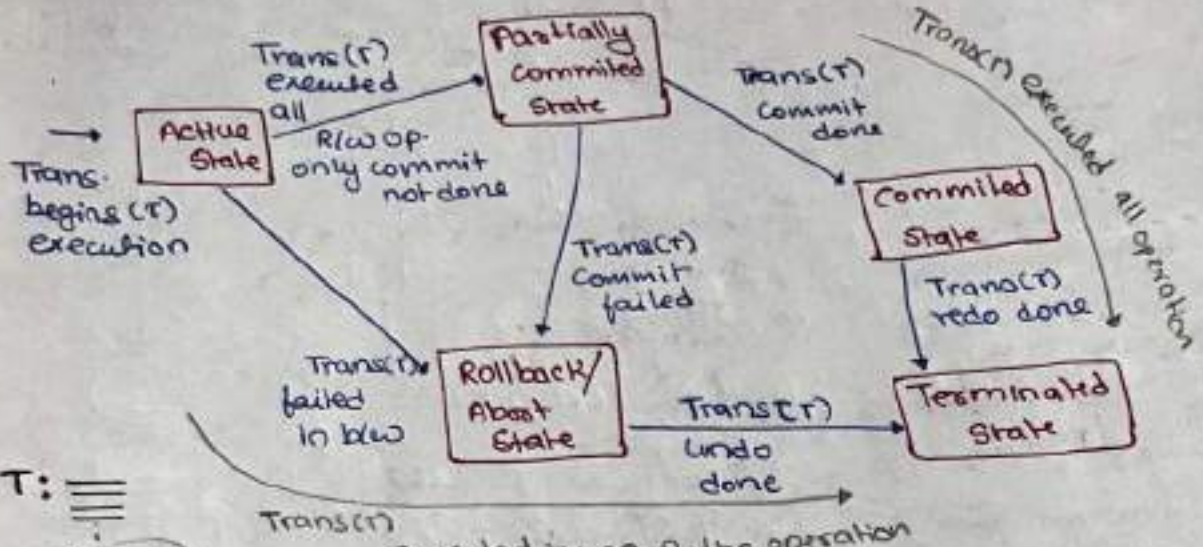


● Dirty block
DB block which is update by uncommited transaction

● REDO of transaction
all change of DB file b/c of transaction commit

■ Undo of transaction
- all change of DB file b/c of transaction failure or Rollback

States of trans



• undo & redo can be perform any no. of time

T:
Commit

- Durability (3rd) Permanency → now after commit, Despite any failure, All changes by T must Reflect in DB (must be made permanent in DB)
- transaction must be able to rollback under every possible failure. & successful transaction data update of DB file must be persistant [should not be lost]

Failure of Transaction

1. power failure
2. S/w crash [OS crash DBMS crash]
3. Concurrency control protocol may kill trans.

→ logfiles + States of trans. Stored DISK

4. DISK Crash (H/w crash) → handle by → RAID Architecture of DISK design used to recover from DISK crash

RAID:
Redundant of Independent DISK's

- RAID-0
- RAID-1
- RAID-2
- RAID-13

RAID-0: no redundant DISK
RAID-1: mirror image of Independent DISK
RAID-13: high end servers

the database is composed of "element"

can be, Relation, Disk block, page, A particular field value of a record .. etc.

- A database has a state, which is a value for each of it's element

- Database consistency Requirm^n
  ① Explicit: Integrity Constraint
  ② Implicit: DBMS Developer

Database State (snapshot/instance)

Consistent State

- All Requirment hold (All consistency) Constraints

Inconsistent State

- Some constraint violation.

- Simplified definition.
- Database element: (A)
- Read(A,t) or Read(A)
  ① Block containing A comes in M.M from Disk buffer of Trans. T
  ② t: temp program variable
  t = A
  A = A

note

- write(A,t) or write(A)
  ① physical same
  ② In memory copy of A
  ③ Store Block containing A in Disk (Database)

- write(X,t) doesn't see the value of X. It simply puts 't' into X (only bring block containing X, doesn't read X)

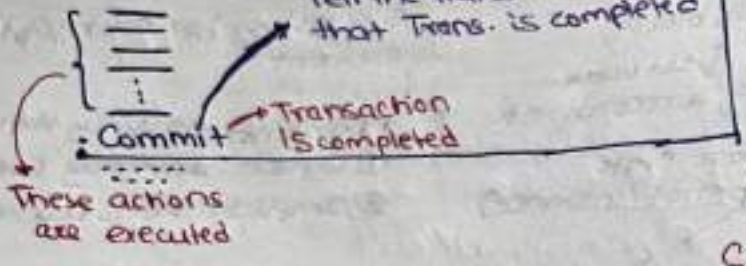
- Simplified Assumption (which may not be true but we will assume it)
  ① every transaction has it's own M.M buffer is not true, they have a common buffer for All transition → but we are assuming it
  ② write operation store updated block into Disk immediately which is not true

- write operatn: Store operatn
- Read operation: Load operatn

Read Block Containing A ≠ Read A

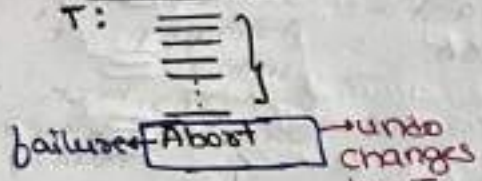
doesn't mean Read A

- **Commit**



- at the time of Commit, changes by trans^n may or may not be Reflected in DB(Disk) immediately but it will eventually (at that point or any point after commit)

- **Abort or Rollback**



- If we don't write Commit @ Abort in last then By default its Commit
- But if it fail then abort should be the last Instn → compulsory

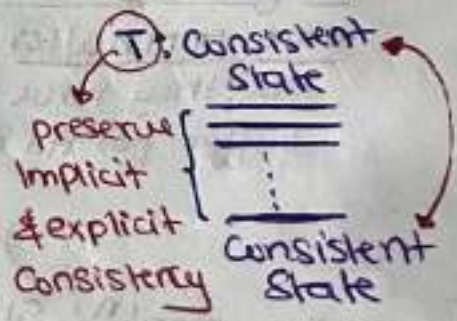
- Concurrent means Interleaved execution
- Database must go from one consistent state to another consistent state after concurrent execution of Transaction.

ACID property

Consistency (or) (Correctness principle)
Property: these particular Transaction must take Database from one consistent state to another consistent state

Individual Trans. coding.

Assume no failure & Assume no concurrent Execution
one transaction executing.



Database Consistency Requirement
- → Explicit: Integrity constraints (can be ensured (enforced by DBMS S/w)
- → Implicit: Applicat^n dependent constraints (within a single transaction, there is a Responsibility of programmer)

ACID → Individual Trans. must be correct (Responsible: Programmer)

↳ the responsibility of the programmer who code a transaction to write correct code.

(to make sure that all the database consistency requirements are satisfied by the Transaction)

• Recovery + Trans. manager:
they will take care of
↳ Concurrent execution (Isolation)
↳ failure within a transaction (Atomicity)
↳ failure after Trans. commit (Durability)

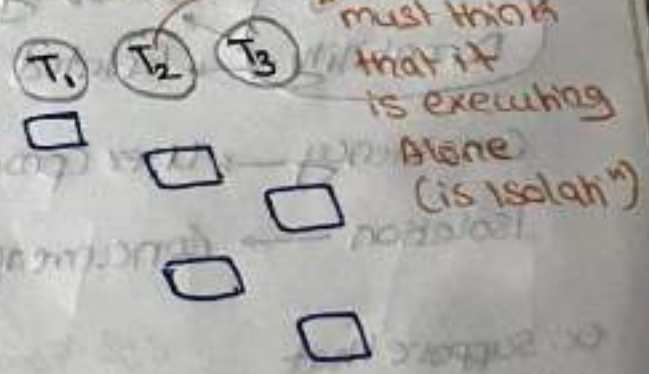
• ACID
↳ every property is responsible for Database consistency
↳ single trans. consistency

ex: can DBMS enforce consistency property of transaction?
(no), we can only assume that code was correct there was no error.

ex: Does serial execution Always presenes Database consistency? yes
& Reason is that
Assumption of correctness of every Transaction. [trans. is always Consistent]

ACID
Isolation:
pseudo feel that you are executing Alone

T_1 T_2 T_3
concurrently

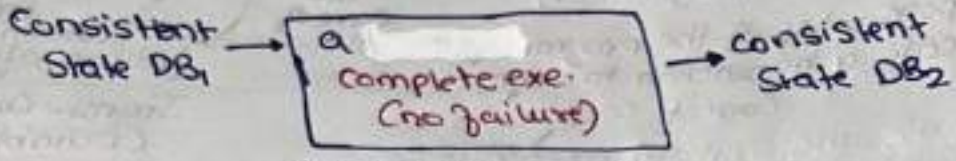


Isolation
↳ To ensure Isolation (many way) → most strongest way → Serializability

↳ a trans. should not see the changes made by other transactions

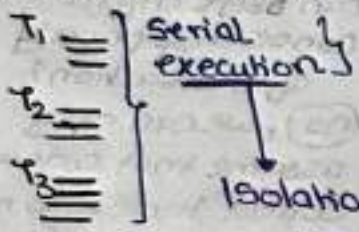
we Assume that a particular Transaction is Always consistent

Consistency in ACID property



(no interleaving of other transactions)

Q. How to Ensure Isolation property holds:



$\Rightarrow T_i$ is executing in Isolation

↳ every T_i is executing in Isolation

Isolation prop. Satisfied But very poor performance (throughput)

(2) All concurrent execution whose effect is equivalent to some serial execution.

ACID

How is responsible

Atomicity → Recovery manager

Durability → Related to failure

Consistency → User (programmer)

Isolation → Concurrency Controller

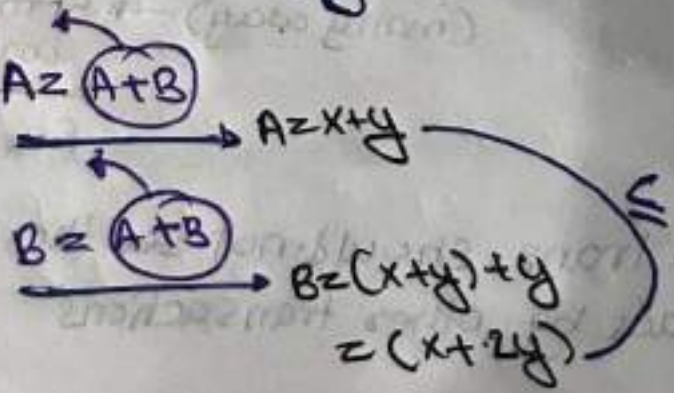
ex: Suppose that Consistency constraint on database is $0 \le A \le B$

transaction preserves Consistency

(1) $A = A+B$; $B = A+B$

Initially $A = x$, $B = y$

$0 \le x \le y$



(2) B=A+B; A=A+B Inconsistent

0≤A≤B Initially A=x; B=y

0≤x≤y

Transaction

↓

Consistent

B=(A+B)

B=x+y

A=(A+B)

A=x+(x+y)

=2x+y

if x=0 & y=1

then B=2 ⇒ inconsistent state

& A=3

Transⁿ.

R(A)

W(A)

Commit

Recovery manager concurrently update A in Disk(DB) at any point

Durability

Due to any failure, we should not forget trans. effect

T_1, T_2, T_3:

Consistent DB_1 state

Serial

T_1 T_2 T_3 → DB_2

T_2 T_1 T_3 → DB_2

T_3 T_1 T_2 → DB_2

Serial

Consistent

may or may not be same

- Serializability: concurrent Schedule(s)

S is serializable iff for every initial consistent DB State, Effect of S is same as some Serial schedule

ex: Swapping two consecutive operatn of S necessarily make it invalid Schedule on T_1, T_2 false

T_1: a_1, a_2, a_3

T_2: b_1, b_2

S_1: a_1, b_1, a_2, a_3, b_2 Valid Schedule

S_2: a_1, b_1, a_3, a_2, b_2 Invalid Schedule

• Swapping two non-consecutive operatn of S necessarily make it invalid schedule on T_1, T_2 (True)

S: — — — — — — — Valid

Swap → Invalid

note: if # Transactions ≥ 3 then these stmt may or may not be true

T_1: a_1, a_2 ....

T_2: b_1, b_2 ....

T_3: c_1, c_2 ....

S: a_1, b_1, c_1 ....

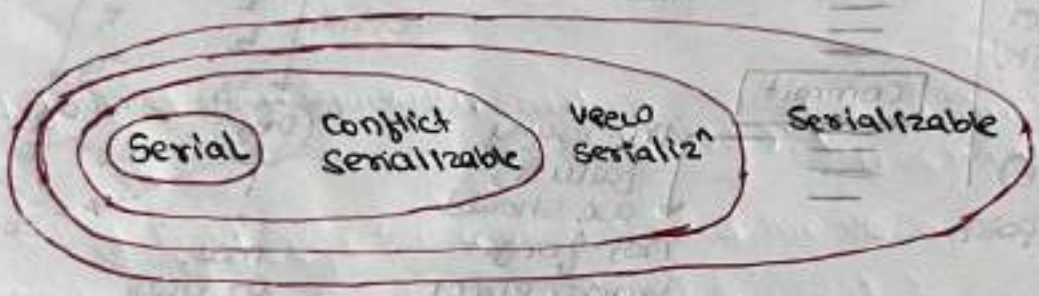
Swap valid

• Serializability:

① Serializability ----→ Theoretical Idea (Almost impossible to implem^n)

② conflict Serializa^ ---→ very easy to implement

③ view Serializa^ ---→ NP complete Impementation



ex:

T_1	T_2
Read(A,t)	
t: t+100	
write(A,t) - - - - - - - - - -	DB, A = C+100
	Read(A,S)
	S: S*2
	write(A,S) - - - - -	DB, A = 2(C+100)
	Read(B,S)
	S: S*2
	write(B,S) - - - - -	DB, B = 2C
Read(B,t)	
t: t+100	
write(B,t) - - - - - - - - - -	DB, B = (2C)+100

A=B → Assume it to be 'C'

↳ Consistency Constraint

S_2 not preserved consistency

So, $S_2 \neq (T_1 \rightarrow T_2)$ and $S_2 \neq (T_2 \rightarrow T_1)$ } S_2 is not Serializable.

T_1	T_2
Read(A,t)	
t: t+100	
write(A,t)	..... DB, A=C+100
	Read(A,S)
	S: S+200
	write(A,S) ..... DB, A=C+300
	Read(B,S)
	S: S+200
	write(B,S) ..... DB, B=C+200
Read(B,t)	
t: t+100	
write(B,t)	DB, B=C+300

A=B → assume to C'

So,

$S_1 \equiv T_1 \rightarrow T_2$

$S_2 \equiv T_2 \rightarrow T_1$

} S_1 is Serializ^

But S_1 is not view or conflict Serializa^

(note) serializability (in General)
Relies on All operatn & computation

Conflict Serializability

based only on the read & write not on the actual value.

- Conflict Equivalent schedule

ex: Swap 2 consecutive non-conflicting operation of two transactions will not change the effect of schedule.

① conflict [redacted] pairs.

S: .... (I) .... (J) ...

T_1	T_2
W(A)	
	W(A)

Conflict pair

≠

T_1	T_2
	W(A)
W(A)	

T_1	T_2
P(A)	
	R(A)
W(A)	
	W(B)

(S) T_1 T_2 T_3 → Repeatedly → (S') T_1 T_2 T_3

Swap non-conflicting Consective opn of Diff. transaction then, S, S' are conflict equivalent.

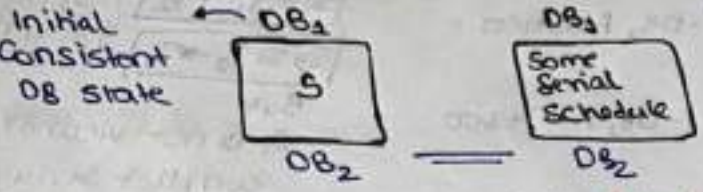
(note)

- no edge from committed transaction to newly started transaction

} → wrong

Commit will be ignored for serializability

• Check for Serializability
given schedule S; for ALL possible DB;

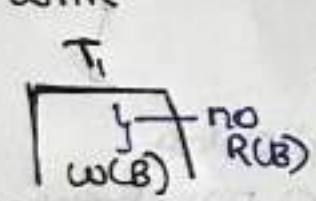


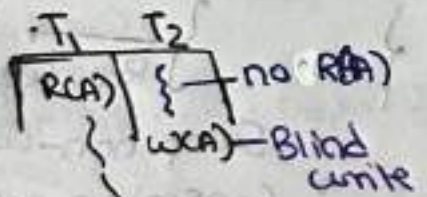
i.e Schedule S & some serial schedule affect DB in same way.

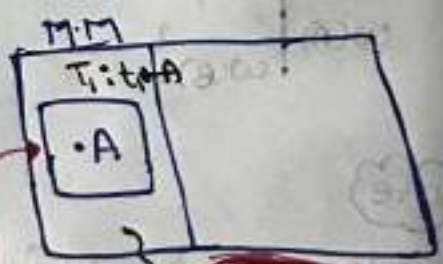
• check for conflict serializability [less strict & more Time compln]
① only focus on Read, write operation (ignore all computation)
② create precedence graph
↳ Acyclic : schedule is C-S
↳ find Topological order of precedence graph to find
(CESS)
Conflict equivalent serial schedule

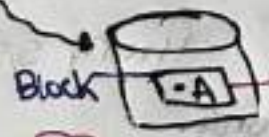
• method to find view equivalen serial schedule (VESS): [more strict & less time compln]
given schedule S;
① for every Read opn, same view in S & the VESS
② for any element e; final write of e must be done by same Transaction in both S, VESS

Blind write

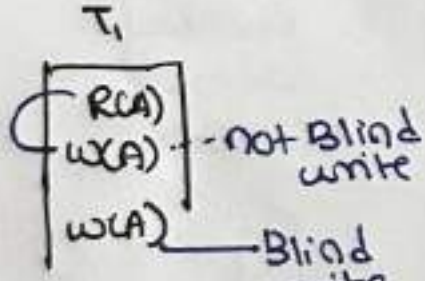








is shared by T1, T2

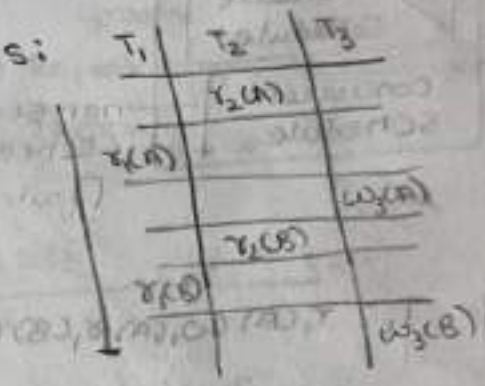


note
• Buffer in M·M is shared by Transaction But Temp. Variable are always their private

Ccorrectness principle) (or)

- C: Consistency
  - Database must be consistant before & after execution of transaction
  - Incorrect result produced by faulty transact^n not recovered by DBMS
- Schedule:
  time order of execution sequence of two or more transaction
  Time order execution sequence:-
  S: $r_2(A), r_1(A), w_3(A), r_2(B), r_1(B), w_3(B)$

- Isolation [concurrency control component]
  - Concurrent execution of 2 or more transaction result must be equal to result of some serial execution of trans action



Serial Schedule

- Transaction must executed one after other

Ex: T_1: Transfer 500 from A to B
T_2: Display A+B balance

S_1: $R_1(A)\ W_1(A)\ R_1(B)\ W_1(B)\ C_1\ R_2(A)\ R_2(B)\ C_2$
$T_1 \longrightarrow T_2$ Serial

S_2: $R_2(A)\ R_2(B)\ C_2\ R_1(A)\ W_1(A)\ R_1(B)\ W_1(B)\ C_1$
$T_2 \longrightarrow T_1$ serial

Adv: every serial schedule is always produce correct result (Preserves Integrity)

Dis Adv: 1. less degree of concurrency
2. Less throughput
3. Resource utilizath not good
4. more response time

[Degree of concurrency : # of trans simultaneously]

Concurrent Schedule

Simultaneous/ Interleaved/concurrent

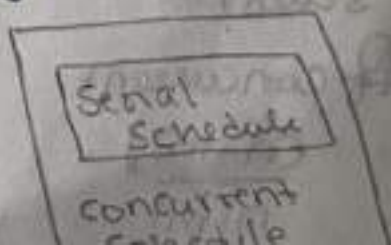
S_3: $R_1(A)\ W_1(A)\ R_2(A)\ R_2(B)\ C_2\ R_1(B)\ W_1(B)\ C_1$
not equal to $T_1 \to T_2$, $T_2 \to T_1$
1500+3000 = 4500
(Incorrect schedule)

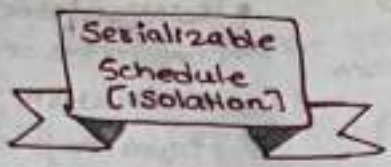
S_4: $R_1(A)\ W_1(A)\ R_2(A)\ R_1(B)\ W_1(B)\ C_1\ R_2(B)\ C_2$
1000
3100
A+B = 5000 (S_2)
A: 2000 1000
B: 3000 3100
[Correct sequence]

S_4: Isolation

Adv: (1) more degree of concurrency
(2) more throughput.

Disadvantage: not every concurrent schedules preserves. Integrity





- Sched S is Serializable iff some serial sched(s) must be equal to sched(s)

Ques] $T_1, T_2, T_3 \cdots T_n$ Trans

#f serial sched : $n!$

$T_1\ T_2\ T_3$
$T_1\ T_3\ T_2$
$T_2\ T_1\ T_3$
$T_2\ T_3\ T_1$
$T_3\ T_1\ T_2$
$T_3\ T_2\ T_1$

$3!$ serial Sched.

Ques] T_1: $r_1(A)\ w_1(A)\ r_1(B)\ w_1(B)$ (4)

T_2: $r_2(A)\ r_2(B)$ (2)

How many Concurrent Sched?

T_1	T_2
1. $r_1(A)$	
2.	$r_2(A)$
3. $w_1(A)$	
4.	$r_2(B)$
5. $r_1(B)$	
6. $w_1(B)$	

$6C_4 * 2C_2 = 15$.
Concurrent Sched

S:[_ _ _ _ _ _]

$\frac{6!}{4!2!} = 15$
Concurrent Sched

#f Concurrent: 15
#f serial: 2
#f non-serial: 13

T_1 T_2 Trans with
n & m #f each
How many concurrent sched?
#f concurrent
$\frac{(n+m)!}{n!m!}$

T_1 T_2 T_3 Trans
with x, y, z #f operation respectively
#f Concurrent Sched = $\frac{(x+y+z)!}{x!y!z!}$

Serializability

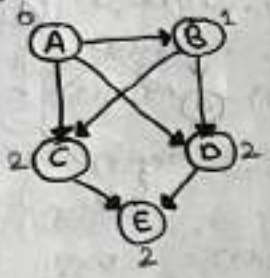
1. Conflict Serializable Sched:-
   1. Topological order
   2. Conflict pair
   3. precedence graph
   4. Conflict equivalent Schedules

- **Topological Order**
  graph traversal algo which is used is applicable for Directed Acyclic graph

Step
1. visited vertex v whose Indegree is "0" & delete vertex v[visited] from graph
2. Repeat step 1 for all vertex of the graph

Indegree

i.e

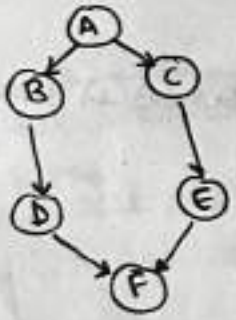


Topological Order

A : B < C : D : E / D : C : E

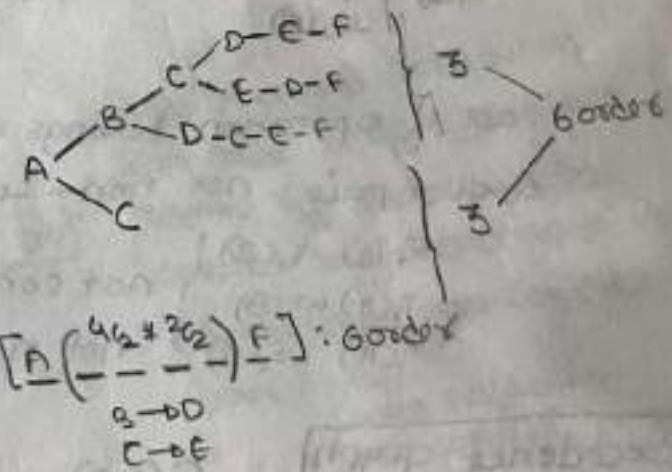
ABCDE } Topological order
ABDCE }

Qu] How many Topological order?



O/P. 6

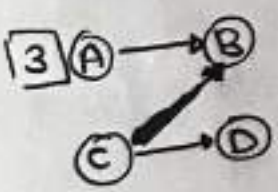
ABDCEF
ACBDF
ABCDEF
ACBDEF
ABCEDF
ACBEDF



$\left[A\left(\frac{4!}{2!}\times\frac{2!}{2!}\right)F\right]$: 6 order

3 → D
C → E

3

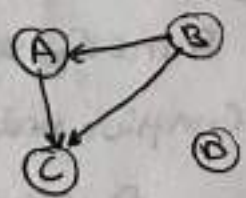


O/P 4

ACBD
ACBD

CABD
CADB

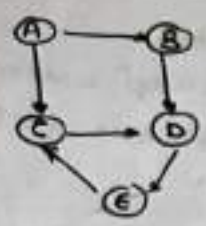
4



DBAC

BDAC

BADC
BACD

5. 

#f Topological order : 0
[cyclic]

6. #f Topological order of null graph with n vertices.

$n!_0$

{ A C B D } : 4!

2. Conflict pair

- two operation of sched S is conflict pair iff atleast 1 write operation, over same data item, from different transaction

S: ... $r_i(x)$ ... $w_j(x)$ } conflict pair
S: ... $w_i(x)$ ... $r_j(x)$
S: ... $w_i(x)$ ... $w_j(x)$

S: ... $r_i(x)$ ... $r_j(x)$ } non-conflict pair
S: ... $r_i(x)$/$w_i(x)$ ... $r_j(y)$/$w_j(y)$

S:

T_1	T_2
$r_1(A)$	
$r_2(B)$	
$w_1(B)$	
	$w_2(A)$

$r_1(A)\, w_2(A)$: conflict pair

$r_1(B)\, w_2(A)$ } non conflict pair
$w_1(B)\, w_2(A)$

Pair of operation belongs same trans in sched(s) are neither conflict pair nor non-conflict pair.

$r_1(A)\ r_1(B)$ } not conflict pair
$r_1(A)\ w_1(B)$
$r_1(A)\ w_1(B)$ } not non-conflictpair

3. Precedence graph $G(V, E)$

Vertices (V): Trans of sched(s)

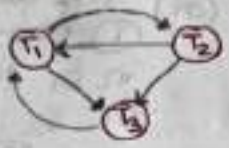
Edges (E): Conflict pair precendnce of schedules(s)

↳ It can be cyclic or Acyclic

↳ no self group

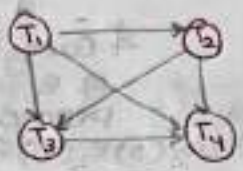
i.e S:- $r_2(A)\ r_2(B)\ w_1(A)\ w_2(A)\ w_3(B)\ w_1(B)\ w_3(A)$

Ans:

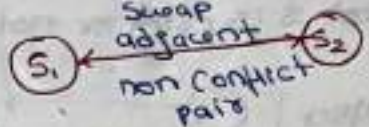


S: $r_1(A)\ r_2(A)\ r_3(A)\ r_4(A)\ w_1(B)\ w_2(B)\ w_3(B)\ w_4(B)$

Ans



4. **Conflict equal schedules:-**
S_1, S_2 conflict equal iff
S_2 derived by swapping adjacent non conflict pair of S_1



: $S_1 S_2$ conflict equal Schedules

Can't solve b/c it's not conflict pair & non-conflict pair

not adjacent so after swaping will create different sequence

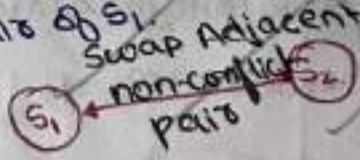
S_1 : $r_1(A)\ r_1(B)\ w_2(A)$
r_2 : $r_2(A)\ w_2(A)\ r_1(B)$

} conflict equal.

Conflict serializable Sched

OR

Conflict equal schedules:-
Schedule S_1 & S_2 conflict equal iff schedule S_2 derived by interchange of consecutive non conflict pair of S_1.



: $S_1\ S_2$ Conflic equal

- Sched(S) is conflict serializable iff (some) serial sched (S') must be conflict equal to sched(S)

{Given Sched (S)} ↔ conflict equal {S': serial Schedule (some)}

Conflict serializable Schedule

{Precedance graph of S also must be Acyclic} ↔ conflict equal {precedence graph of serial sched S' Acyclic}

Topological order

Test condition of Conflict Serializable Schedule

- Schedule S is conflict serializable schedule iff precedence graph(S) must be Acyclic & conflict equal Serial schedule are Topological order of schedule(S) precedence graph

(OR) S_1, S_2 are conflict equal iff (1) each transaction T_i in S_1 must be same in S_2

S_1: ... T_i ... T_j ... $R(A)$ $R(B)$ $W(B)$

S_2: ... T_i ... T_j ... $R(A)$ $R(B)$ $W(B)$

& every conflict pair precedence in S_1 must be same precedence in S_2.

Ques] $S: R_1(A) R_3(C) R_1(B) W_2(B) W_3(C) W_2(D) R_3(D) R_3(E) R_4(E)$

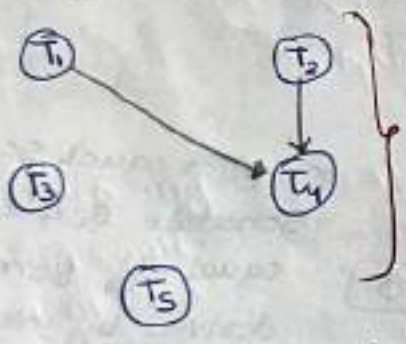
Test S is CSS or not?

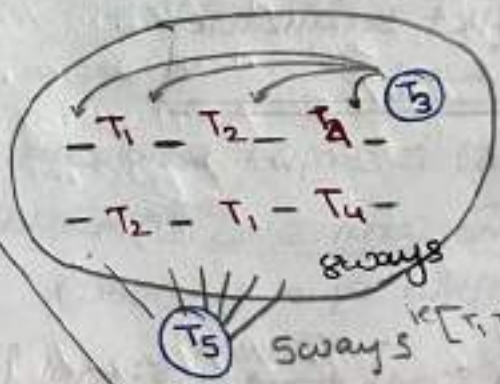
S is conflict Serializable

[S': $T_1 T_2 T_3$ Serial Sched conflict equal to S]

Ques] $S: R_1(A) R_2(A) R_3(A) R_4(A) W_2(B) W_4(B) R_3(C) W_4(D) R_4(E) W_4(E) R_5(A)$

How many serial sched are conflict equal to sched(S)?



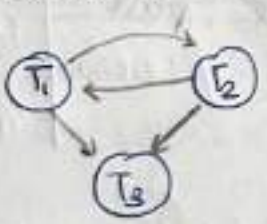


sways

T_5 sways in $[T_1 T_2 T_3 T_4]$

40 Topological order.

Ques] $S: R_2(A) R_2(B) W_1(A) W_1(B) W_2(A)$ $R_3(A) W_3(A)$



- not CSS
- no serial sched conflict equal to S
- 0 Topological order

View Serializable

Schedule S: S_1, S_2

$S_1 \equiv_v S_2$

View equivalent

① Initial Read

- For same read op^n,

Same view

by Same trans & by same write opn

S_1 S_2

Always have Same view (Read)

S_1

T_1	T_2
W(A)	
	R(A)
W(B)	
	R(B)

View → Source: T_2

S_2

T_1	T_2
W(A)	
	R(B)
	R(A)
W(B)	

View → Source: T_0 (Initial DB)

View → Source: T_1

So, $S_1 \not\equiv_v S_2$

② final write

S_1

T_1	T_2
W(A)	
	W(A)

final write on A is done by T_2

S_2

T_1	T_2
	W(A)
W(A)	

T_1

$S_1 \not\equiv_v S_2$

- final write should be done by T_b

Note

S_1

T_1	T_2
1 --- W(A)	
	R(A)
2 --- W(A)	

View source: T_1(2)

S_2

T_1	T_2
W(A)	
	W(A)
	R(A)

View source: T_1(2)

not Same view

$S_1 \not\equiv_v S_2$

Ullman

Ex: VSES?

$r_1(A), r_2(A), r_3(A), w_1(B), w_2(B), w_3(B)$

T_1	T_2	T_3
R(A)		
	R(A)	R(A)
w(B)		
	w(B)	w(B)

Element	Initial Read	Write by	Final write
A	T_1 (from Initial DB T_0) T_2 (---"---) T_3 (---"---)	—	—
B	—	T_1, T_2, T_3	T_3

} → only eqⁿ

is $(T_1, T_2) \rightarrow T_3$

VESS: $T_1\ T_2\ T_3$ & $T_2\ T_1\ T_3$

ex:

T_1	T_2
1. w(A)	
	R(A)
2. w(A)	

Is it VS?

$T_2 T_1$ ✗

→ $R_2(A)$ from initial DB

$T_1 T_2$ ✗

→ $R_2(A)$ from 2nd write of T_1

T_1	T_2
$r_1(A)\ w_1(A)$	
	R(A)

Same transⁿ in the se order it will happen

note

- Take care when any Transaction write Same element more than once
- If no transaction write same item more than once then you can focus only on Transaction; not on any particular write opⁿ

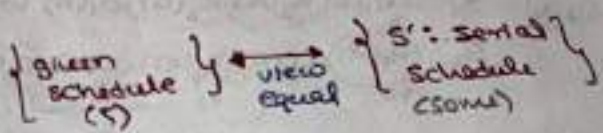
T_1	T_2
w(A)	
	w(A)
w(A)	

} → not C.S but V.S

} b/c
V.S: uses the Blind write carefully
C.S: Doesn't take benefit of the Blind write

- The only difference b/w C.S & V.S is when Blind write are present. (Otherwise they are same)
- if no blind write then } V.S = C.S then check only C.S

View Serializable Schedule

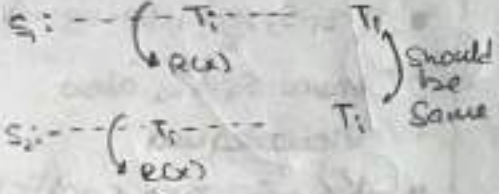


View Serializable schedule.

If some schedule(T) is view serializable iff some serial Schedule(S') must be view equal to schedule(T)

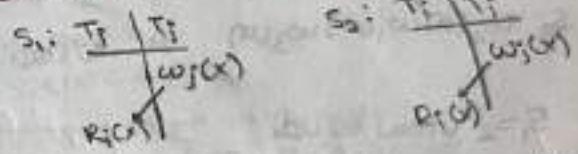
- View equal schedules :-
  S_1 S_2 view equal if

① Initial Read :-
if T_i Read x from initial Database in S_1, then in Schedule S_2 also (T_i) must Read(x) from initial Database



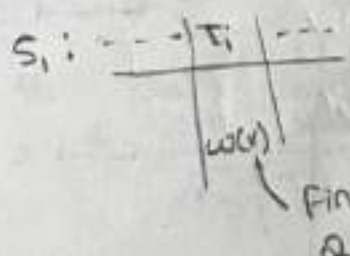
- By default, both should have same set of operation but order can be different

and ② Updated Read [W-R] :-



- if T_i Read(x) which is update by T_j in S_1, then in schedule S_2 also T_i must Read(x) which is update by T_j

③ Final write :-
if final write of x done by transaction T_i in schedule(S_1) then in schedule(S_2) also final write(x) must by Transaction T_i



Ques S_1: $R_1(A)\ R_2(A)\ R_3(A)\ w_1(A)\ w_2(A)\ w_3(A)$ (IR, IR, IR)

S_2: $R_1(A)\ w_1(A)\ R_2(A)\ w_2(A)\ R_3(A)\ w_3(A)$ (IR; update read; update read)

S_1, S_2 not view equal

Ques S_1: $R_2(A)\ R_3(B)\ R_1(A)\ w_2(A)\ w_1(A)\ R_2(C)\ R_3(C)\ w_1(B)\ w_2(B)\ w_3(B)$ (IR, IR, IR, FW, IR, IR, W-R, FW)

FW A: T_1, B: T_3

S_2: $R_1(A)\ R_3(C)\ R_2(A)\ R_2(B)\ w_2(A)\ w_1(A)\ R_2(C)\ R_3(A)\ w_2(B)\ w_1(B)\ w_3(B)$ (IR, IR, IR, IR, FW, IR, FW)

FW A: T_1, B: T_3

S_1, S_2 are view equal & not conflict equal sched.

③ S_1: $w_1(A)\ w_2(A)\ w_3(A)$ A: ~~50~~ 100 300

S_2: $w_2(A)\ w_1(A)\ w_3(A)$ A: ~~100~~ 50 300

S_1, S_2 view equal but not conflict equal

View Serializable
Sched testing

Ques Test given sched (S) view Serializable? or not

S: $R_1(A)$ (IR) $R_2(A)$ (IR) $w_1(A)\ w_2(A)$ FW: A: T_2

not view equal

S': $R_1(A)$ (IR) $w_1(A)$ (W-R) $R_2(A)\ w_2(A)$ FW: A: T_2

$T_1 \rightarrow T_2$ serial

not view equal

S': $R_2(A)$ (IR) $w_2(A)$ (W-R) $R_1(A)\ w_1(A)$ FW: A: T_1

$T_2 \rightarrow T_1$ serial

S: not VSS

note

- If S_1, S_2 conflict equal then S_1 & S_2 also view equal
- If S_1, S_2 not conflict equal then S_1 & S_2 may or may not be view equal
- if S_1, S_2 not view equal then S_1, S_2 is not conflict equal

S $\xleftrightarrow{VE}$ S': serial

VSS

S $\xleftrightarrow{CE}$ S': serial

CSS

■ VSS Testing Condition:-

Given Sched(S) ⟷ View equal Sched(s)

(i) Final write (FW)

- $x \Rightarrow T_i\ (T_j)_{FW}$ ⇒ $T_i \rightarrow T_j$
- $y \Rightarrow T_i\ T_j\ (T_k)_{FW}$ ⇒ $(T_i, T_j) \rightarrow T_k$ (any order)
- $z \Rightarrow (T_i)$ ⇒ Every serial view equal

(ii) IR (Initial Read)

data item	Initial Read	write	
x	T_i	T_j	⇒ $T_i \rightarrow T_j$
y	T_i (T_j)	T_j T_k	⇒ $T_i \rightarrow (T_j, T_k)$, $T_j \rightarrow T_k$
z	T_i T_j	-	⇒ every serial view equal.

(iii) Update Read (UR)

$W_i(x) \rightarrow R_j(x)$ && no other Trans write x

$T_i \rightarrow T_j$: { $T_k\ T_i\ T_j$; $T_i\ T_k\ T_j$; $T_i\ T_j\ T_k$ }

$W_i(y) \rightarrow R_j(y)$ && some other trans T_k also write y

$T_i \rightarrow T_j$ && T_k should not blw $T_i T_j$ { $T_k\ T_i\ T_j$; $T_i\ T_j\ T_k$ }

Qus) Sched(S) view serializable or not? S : $R_2(A)\ W_1(A)\ W_2(A)\ W_3(A)$

Given Sched(T)

FW: A : $T_1\ T_2\ (T_3)$

IR :

data item	Initial Read	write
A	T_2	$T_1 T_2 T_3$

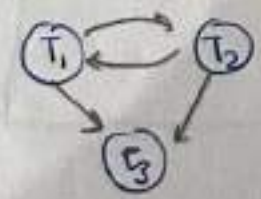
View equal Sched

- $(T_1, T_2) \rightarrow T_3$
- $T_2 \rightarrow (T_1, T_3)$

S' : $T_2\ T_1\ T_3$ Serial Sched view equal to Sched S.

∴ S is VSS

(ii) Conflict Serializable



S not CSS

Ques 5: $R_2(A)\ R_2(B)\ W_2(A)\ R_3(C)\ W_1(A)\ R_1(C)\ R_3(A)\ W_1(B)\ W_2(B)\ W_3(B)$

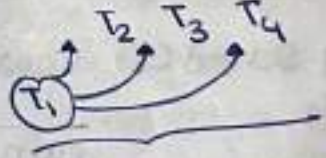
Give schedule	View equal serial (S')
(FW) A: $T_2 T_1$ B: $T_1 T_2 T_3$	$T_2 \rightarrow T_1$ $(T_1, T_2) \rightarrow T_3$
(IR) A: $T_2 T_1$ — T_2 B: T_2 — $T_1 T_2 T_3$ C: $T_3 T_1$ — –	$T_2 \rightarrow T_1$ $T_2 \rightarrow (T_1, T_3)$
(WR) $W_1(A) \rightarrow R_3(A)$ All others reads write A: T_2	$T_1 \xrightarrow{\times T_2} T_3$

Serial schedule (S')
$T_2 T_1 T_3$
View equal to S
∴ S is VSS

Ques Schedule is view serializable or not? & How many serial schedule view equal to S?

S: $R_1(A)\ R_2(A)\ R_3(A)\ R_4(A)\ W_1(B)\ W_2(B)\ W_3(B)\ W_4(B)\ R_3(C)\ W_4(C)\ W_2(D)\ W_3(D)$?

Given schedule	View equal serial (S')
(FW) A: – B: $T_1 T_2 T_3 T_4$ C: T_4 D: $T_2 T_3$	– $(T_1, T_2, T_3) \rightarrow T_4$ – $T_2 \rightarrow T_3$
(IR): A: $T_1 T_2 T_3 T_4$ — – B: – — – C: T_3 — T_4 D: –	– – $T_3 \rightarrow T_4$ –
(UR):	



$T_1 T_2 T_3 T_4$
$T_2 T_1 T_3 T_4$
$T_2 T_3 T_1 T_4$
} Serial view equal to S.

Ques] S: $w_1(A)\ R_2(A)\ w_3(A)\ R_4(A)\ w_5(A) R_6(A)$

FW: A: $T_1 T_3 T_5$ $(T_1, T_3) \rightarrow T_5$

data item	Initial Read	write
A	$T_2 T_4 T_6$	$T_1 T_3 T_5$

UR:

$w_1 \rightarrow R_2$ ~~$\rightarrow$~~ $T_1 \rightarrow T_2$ $[T_3 T_5]$ ×

× w_3 × w_5

$w_3(A) \rightarrow R_4(A)$ $\rightarrow$ $T_3 \rightarrow T_4$ $[T_1, T_5]$ ×

other write A : $T_1 T_5$

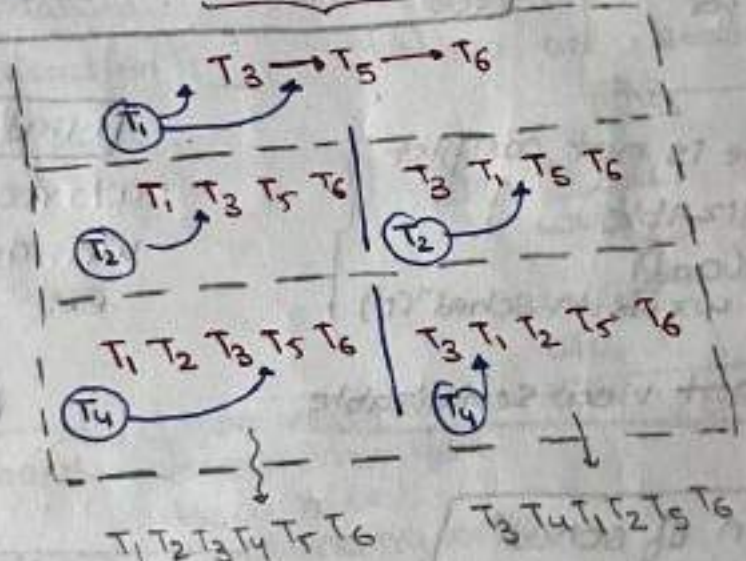
$w_5(A) \rightarrow R_6(A)$ $\rightarrow$ $T_5 \rightarrow T_6$ $[T_1, T_3]$ ×

other write(A) : $T_1 T_3$

Serial schedule view equal to S?

S':

1. $T_1 T_2 T_3 T_4 T_5 T_6$
2. $T_3 T_4 T_1 T_2 T_5 T_6$



Ques] S: $w_1(A) R_2(A)\ w_3(A) R_4(A) w_5(A)\ R_6(A) w_7(A)\ R_8(A)$

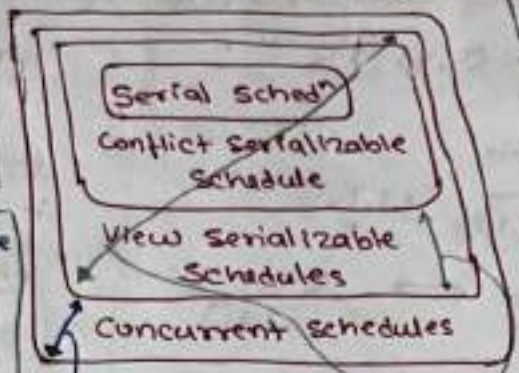
$3! = 6$ serial sched view equal to S

- given schedule is Serializable then to check we apply view serializable schedule

- conflict Serializable schedule condition is only sufficient but not necessary for Seriability

- view serializable schedule condition is both sufficient & necessary for seriability

expressive power of CSS & VSS



not view serializable = non-Serialization

Serializability
CSS / VSS
Isolation

atleast 1 blind write

note

- if schedule S is conflict serializable schedule then S is also view serial^n
- if sched^n (S) is not conflict serial^n schedule then S may or may not be view serial^n
- every conflict serial^n is also view serial but not every view serial^n is conflict

- if { Schedule is not conflict Serializable (and) no blind write in sched^n(T) }

Then S is not view serializable

blind write

- w/o reading if we are writing

ex: ---, ---, ... } no R(A)

w(A) → Blind write

classification of sched^n based on Recoverability

Concurrent execut^n may lead

1. Irrecoverable Problem
2. Cascade Rollback problem
3. Lost update problem.

Note

- these problem can occur even schedule is Serializable

n the testing of Recoverability
inital read is not a problem.
o we can ignore it
problem is inital write.

complete recover not possible

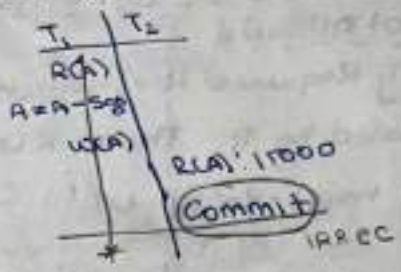
Irrecoverable Schedule

- if T_j Read(x) which is update by T_i & commit of T_j before commit / Rollback of T_i.

T_1: withdraw 5000 from A
T_2: check bal of A

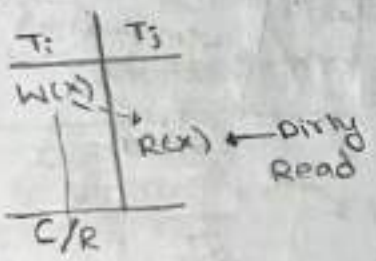
ex: A: 20000
15000
20000

ex: exam hall



Uncommited Read [Dirty Read]

- T_j Read(x) which is updated by uncommited transaction T_i



Recoverable Schedule

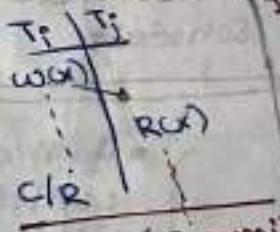
(protocol) condition

Schedⁿ(S) is Recoverable iff

1. NO Dirty Reads in S

OR

2.



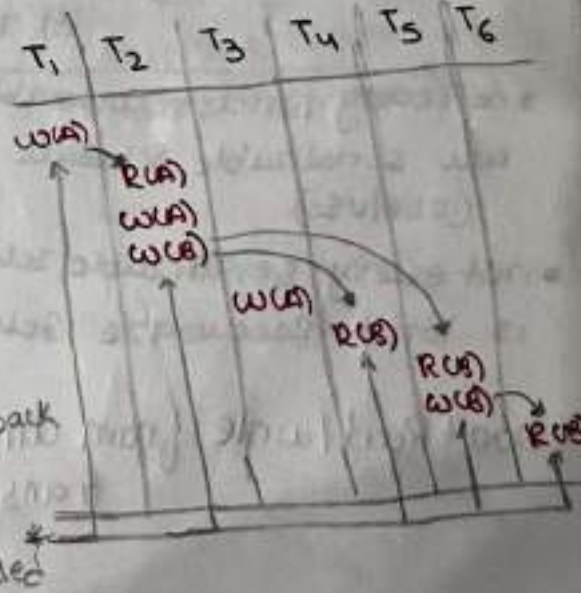
If T_j Read x which is updated by T_i Then commit of T_j must delays until C/R of T_i

Cascading Rollback problem

→ if you want to recover then you have to do challenging work

- failure of 1 transaction force to Rollback some set of other transaction there Kind of forced Rollback is cascading Rollback
- you are reading from uncommitted trans

S:



bc of T_1 failure forced to Rollback

$T_2 T_4 T_5 T_6$ also Cascading RB's

easily recoverable (don't have to do much work)

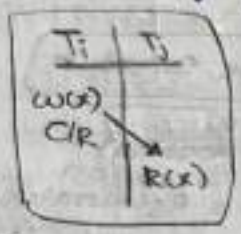
Cascadeless Rollback Schedule

also known as Avoiding Cascading Rollback schedule (ACR schedule)

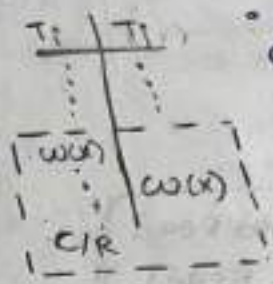
[Avoid cascading aborts]

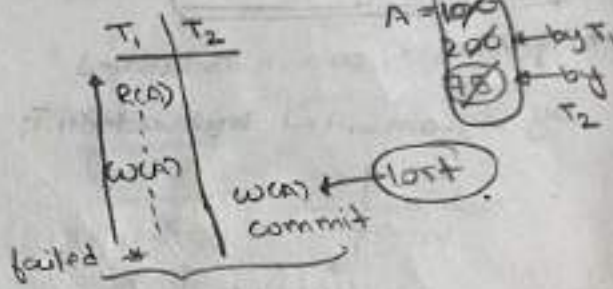
Lost update problem

- uncommitted Reads / Dirty Reads not allowed
- If T_j Required R(x) which is updated by T_i Then R(x) if T_j must delay until C/R of T_i

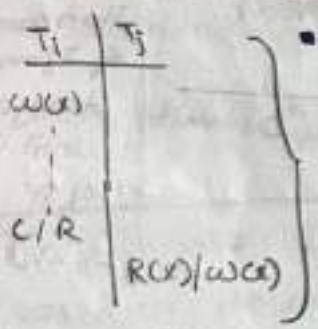


- if T_j write(x) which is already write by uncommited transaction T_i
- allowed by cascadless

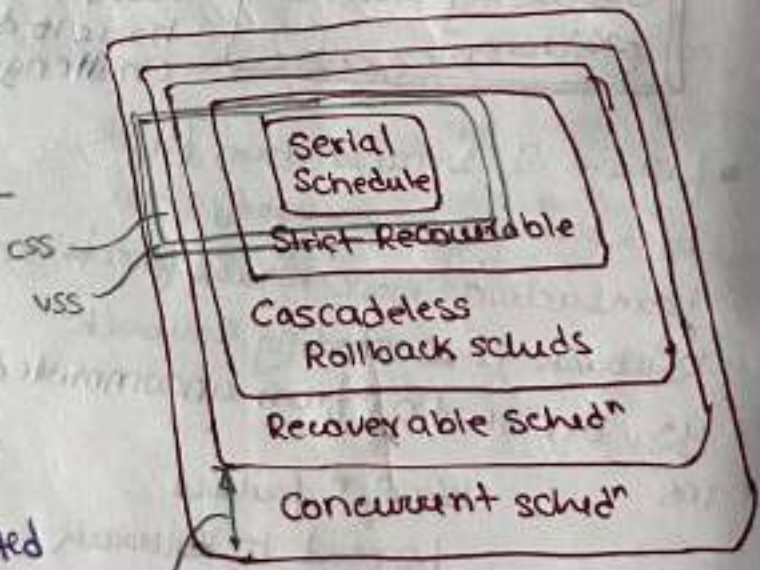




Strict Recoverable Schedule



- if T_i write(x) Then R(x)/ W(x) of T_j must delay until C/R of T_i



- not every strict recoverable are serializable schedule (css/vss)
- not every serializable schedule is strict Recoverable Schedule

- Don't Read / write from uncommited trans.

- Serializability
  If no failure than due to concurrent exec. of trans.
  ① Ensure database consistency
  ② Ensure isolation.

(note)
Showing wrong result/value to user is also considered violation of consistency of database

exserializability concept help us ensure?
- ✓ Isolation
- ✓ Consistency of Database
- ✓ Consistency of schedule
- ✗ Consistency of trans.
- ✗ Atomicity
- ✗ Durability

Consistency of DB
→ To preserve this DB Consistency
- Atomicity (All or none)
- consistency of trans.
- Isolation
- Durability

Recoverability

System: T_1 – commited
T_2 – uncommited
* failure occurs.

Action: Roll back (undo) T_2
T_1 must persist in DB
T_1 must not be undone

Irrecoverable Schedule → violate Atomicity property

T_1	T_2
W(A)	
	R(A)
	Commit
commit/ Abort	

To undo T_1 → we have to undo T_2 (b/c T_2 had Read from T_1 made some Change)
(So we can't undo T_1 correctly) (Atomicity violated)
But we can't undo T_2 (b/c T_2 commited)

ex: ~~Recoverable~~ or not

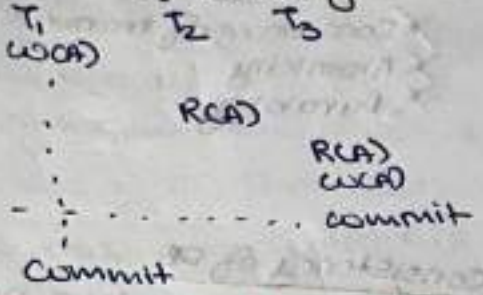
T_1	T_2
W(A)	
✗	W(A) --- Blind write
	R(A)?
	commit

(T_2) is not Dependent on T_1

Commit

Note: Serializability & recoverability are orthogonal concept
↓
means nothing to do with eachother

Violation of Atomicity



To preserve Atomicity
Abort T & Abort all trans. which have Read changes by T.

if you undo T_1 & undo all transactions who are dependent on T_1
↓
& if you are unable to do these means Violation of Atomicity

- Bad idea
  given a schedule S: $r_1(A), r_2(B), w_2(A), r_1(B)$ ← check here at these time that is it conflict serial or not is too late
  ↳ it's too late b/c system has done a lot of work, if we undo all the work, which is painful
  ↳ schedule complete then check for Serializability
- Good idea
  During execution, ensure serializability

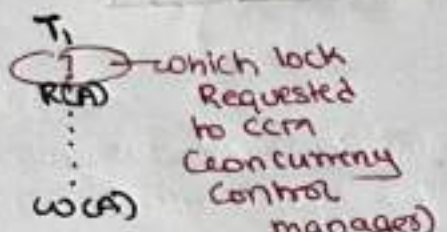
so we will Design Concurrency Control Protocol (Set of Rule)

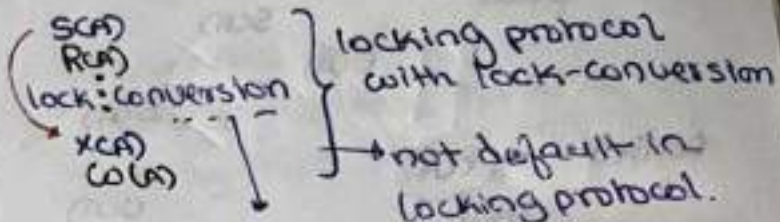
1. Lock Based protocol
2. Time stamp Based protocol

Each transaction, individually follows these rule

(i) locking Based protocol:

- Shared_lock (Read lock) only Read
- Exclusive lock (write lock)
  - write & possibly Read





b/c of there there are problem

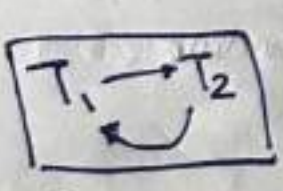
(i) T_1: R(A) → S(A) or X(A)

(ii) T_1: X(A) → R(A) → W(A) — some will use Exclusive lock.

- So, using General (S,X) locking protocol
  non-serializable schedule are possible (can be generated) / Inconsistency possible (so Isolation not guarranted)

T_1	T_2
X(A)	
W(A)	
	X(B)
	W(B)
wait X(B)	
	X(A) wait

- using lock in transaction as described, does not guarantee serializability
  - Deadlock possible
  - Starvation possible
  - Inconsistency possible



Deadlock

• Starvation possibility

T_1	T_2	T_3	T_4
	S(A)		
Denied → X(A) wait			
		S(A)	
	U(A)		
			S(A)
Denied → X(A) wait			
		U(A)	

T_2 Request → S(A) X(A) ← T_1 holds

	S(A)	X(A)
S(A)	yes	no
X(A)	no	no

• Inconsistency But no deadlock, never preferred

Preferred ↳ Deadlock But consistency

C·S vs V·S

Different only when Blind writes exist

V·S vs General Serializability (final state Serializability)

Different when computations given Along with R/W operatn

not given then both are same

• when only Read, write operatn shown (considered)

then Serializable ≡ View serializable

Strict schedule: no uncommitted (Dirty Read) & no over writing an uncommited write

Ensure

✓ Recoverability

✓ Cascadeless schedule

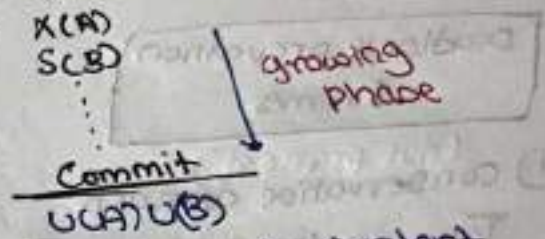
T_1	T_2
W(A)	
⋮	no W(A) no R(A)
C_1	

• 2PL ⟶ Conflict serializability

2PL + Recoverability
- Strict 2PL
- Regorious 2PL

• Strict 2PL:
- Release x-lock only after comithit / Abort
+
- 2PL

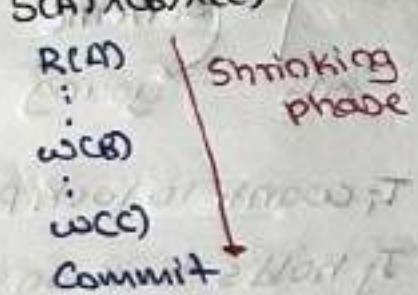
• Regorious 2PL:
2PL + Release All X-locks only After commit/Abort



• Regorious 2PL equivalent Serial schedule : order of commit operatn of Transaction

• Conservative 2PL:

↳ Take all the lock you need before starting Transaction 'T' → Deadlock prevention approach



↳ Dead lock not possible

• 2PL with Lock conversion.

Two new instructions

upgrade(A): you already have S(A) (already have) ⟶ change ⟶ X(A)

downgrade(A): X(A) (Already have) ⟶ change ⟶ S(A)

2PL with lock conversion → b/c of these some serious kind of deadlock will occurs. [i.e in every possibility you will find deadlock]

Request lock; upgrade possible } growing phase

Release locks; Downgrade lock possible } shrinking phase

Deadlock prevention Schemes

(not practical)

① Conservative approach

Take all the lock before Transaction start execution

Conservative 2PL

- It is possible that you will not get all the locks, so there is possibility of Starvation.

(b/c before starting of the execution we don't know what are the item we will access/lock)

good but have many disadvantage

② putting partial ordering the Data items & Accessing items only according to this ordering

↓

the programmer has to know the ordering on the data item

③ use transaction time stamp

① Wait-Die Scheme

Old → Young

T_i want to lock A

T_j holds lock on A

Case 1: $TS(T_i) < TS(T_j)$ (old) (young)

Old (T_i) waits for T_j to release lock

Case 2: $TS(T_j) < TS(T_i)$ (old) (young)

T_i Dies (Abort)

Item A held by T

now, younger than T want to lock A:

Die (Abort)

Older than T wants to lock A: → wait for T to Release A.

(4) wound wait

T_i want to lock A

T_j already holds lock on A.

- old is need wounds (Abort) young
- young in need waits for old

Case 1: $TS(T_i) < TS(T_j)$ (old)

old T_i wound (Abort) young T_j & snatch lock A from T_j

Case 2: $TS(T_j) < TS(T_i)$ (young)

T_i waits for T_j to release.

note very Imp.

when a trans Aborts, it will restart after some time But with it's own old time stamp (to avoid Starvation)

Case 1: Abort transaction T starting with new latest timestamp

then T has become younger possible T can again Abort

T may keep aborting (less priority)

(Starvation)

Case 2: Abort transaction T starting with it's original old time stamp : then T is older now (more priority.)

(no starvation)

note → In wait-Die, wound-wait

Aborted Trans. started with it's original old time stamp.

(to avoid Starvation)

wait-Die : non-preemptive (Don't Disturb executing trans.)

wound-wait : preemptive (preempt & Abort)

- wait-for graph

whenever (at any time)

if T_i no longer waiting for T_j then remove $T_i \rightarrow T_j$ edge.

- 2 phase locking (2PL)
  - ↳ Determine Serializability order of conflicting operations at runtime while txns executes

- Timestamp ordering (T/O)
  - ↳ Determine serializability order of txns before they execute
- Serializability order is fixed "before execution" in order of timestamps.

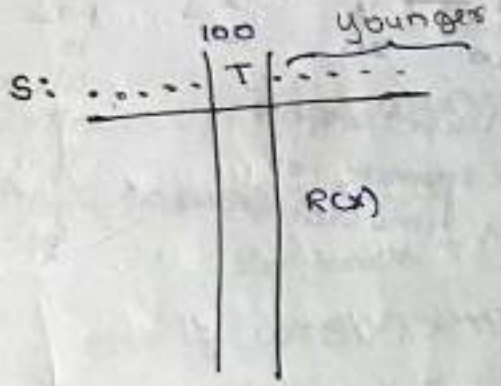
Time Stamp ordering Protocols

- Time stamp values of Tran's :- unique value assigned by DBMS for all trans. in sequence order
  - no two value will have same TS value

TS values for data item(x) :-

Read_TS(x) :- Highest Trans TS value which has executed R(x) successfully

write_TS(x) :- Highest Trans TS value which has executed W(x) successfully

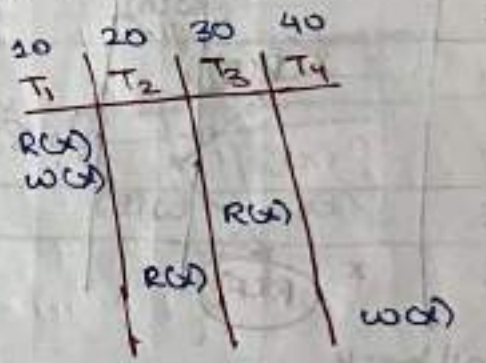


Any younger Trans than T Read(x)?

if RTS(x) > TS(T)
{then younger than T done R(x)}
else
{ younger than T not done R(x)}

- these rule will be followed by individual Transaction.

WTS(x) = ~~10~~ (40) youngest trans which have done W(x)



RTS(x) : ~~10~~ (30) youngest trans done R(x)

Basic idea of timestamp ordering protocols (TSO)

- Concurrent execution of two or more transaction result must be equal to serial schedule based on time stamp order of trans.

(S):

10	30	20
T_1	T_2	T_3
	-	
		-
-		
	-	
		-
-		

- S allowed to execute by TSO protocols if S equal to serial schedule

$T_1 \rightarrow T_3 \rightarrow T_2$
10 20 30
(based on TS orders)

ex:-

(S)

10	30	20
T_1	T_2	T_3
1.		R(A)
2.	W(A)	
3. ✓	R(B)	
4.		W(B) ✓
5.	W(B)	
6. ×	R(A)	

Rollback the trans.

$\Longleftrightarrow$ equal to serial $[T_1 \rightarrow T_3 \rightarrow T_2]$ (10 20 30)

∴ all these conflict are correct for the timestamp order

∴ conflict pair is also allowed if it follow TSO

1 Basic TSO protocol

Possible Rollbacks :-

(S)

10	20
T_i	T_j
	W(x)
R(x)	

* Rollback T_i b/c Read request

(S)

10	20
T_i	T_j
	R(x)
W(x)	

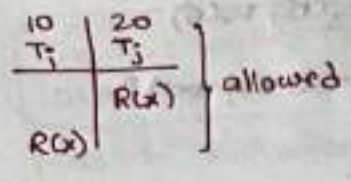
RB T_i

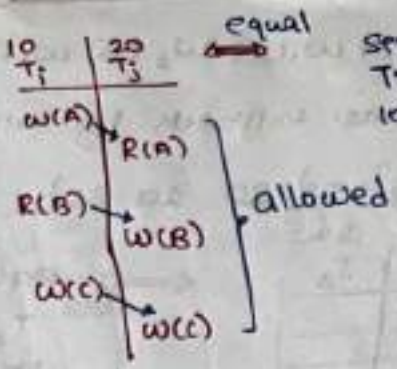
(S)

10	20
T_i	T_j
	W(x)
W(x)	

RB T_i

Rollback b/c of write requests





Serial
$T_i \rightarrow T_j$
10 20

Basic TSO protocol Condition

younger

[1] Trans (T) issues R(x):-

(a) if (WTS(x) > TS(T))
{ Rollback T }
else
{ allowed R(x) by T
Set RTS(x) = max{RTS(x), TS(T)} }

younger

[2] Trans (T) issue W(x):-

(1) if (RTS(x) > TS(T))
Roll back T
else if (WTS(x) > TS(T))
Roll back T

else {
allowed W(x) by T
Set WTS(x) = TS(T)
}

10 20
T_1 T_2
W(x)
R(x) → Abort & Restart

- if you are late, it's your fault (Abort)
- if your turn is gone, it's your fault.

if we start with old T.S then again it will abort so we need to start with new one (new T.S) (latest)

Ques] S : $R_2(A)\ R_2(C)\ W_1(B)\ W_3(A)\ W_1(C)\ W_3(D)\ R_1(D)$

which trans. Rollback by using BTSO Protocol?

(i) $\{T_1\ T_2\ T_3\} = \{20\ 10\ 30\}$

20 T_1	10 T_2	30 T_3
	R(A)	
	R(C)	
W(B)		
		W(A)
W(C)		
		W(D)
R(D)		

Rollback

$\Longleftrightarrow$ equal serial

$T_2 : T_1 : T_3$

10 20 30

RB of T_1

(ii) $\{T_1\ T_2\ T_3\} = \{30\ 10\ 20\}$

T_1	T_2	T_3
	R(A)	
	R(C)	
W(B)		
		W(A)
W(C)		
		W(D)
R(D)		

$\Longleftrightarrow$ equal serial

$T_2 : T_3 : T_1$

10 20 30

// NO RB's

(no RB's)

- Basic Timestamp ordering protocol allowed to execute schedules if S is conflict Serializable && conflict equal serial order

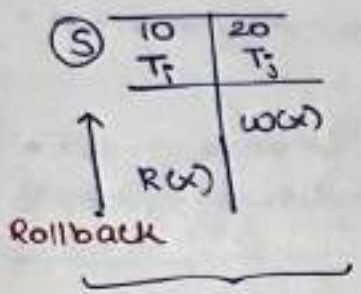
(Topological order) same as TS ordering of Transactions then you will not get any Rollback

- otherwise atleast 1 Rollback guarantee.
- so we can never have cycle in the wait-for graph

(no Deadlock)

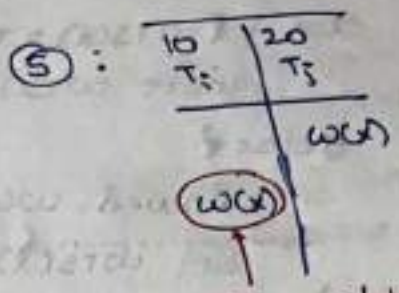
Thomus write TS ordering protocol → TWRTSO

Possible Rollbacks :-

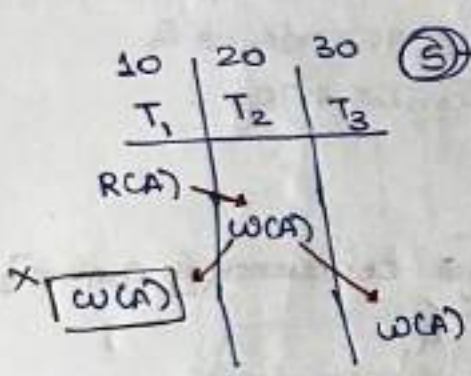


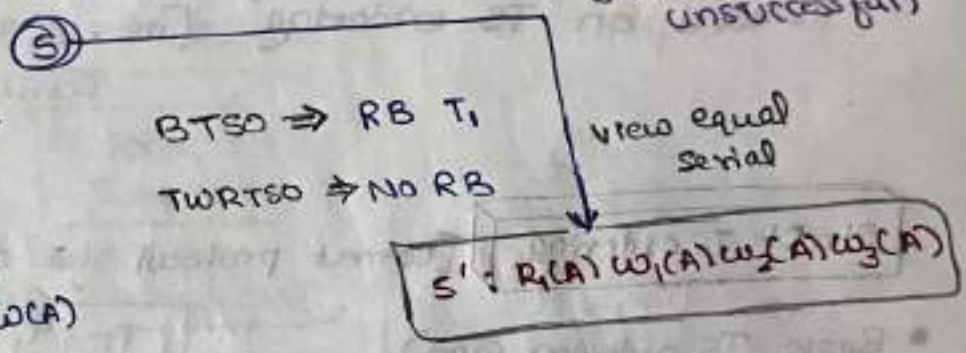
Rollback b/c of Read request

RB b/c of write



Ignore / don't execute W(x) of T_1 & continue execute of T_3 (consider as unsuccessful)





① Trans (T) issue R(x) :-

```
if (WTS(x) > TS(T))
    { Rollback T }
else {
    allowed R(x) by T
    Set RTS(x) = max { RTS(x), TS(T) }
}
```

not using thomas write Rule "non-conflict serializable" are also Allowed

but correctness will not go

② Trans(T) issue w(x) :-

① if (RTS(x) > TS(T)) { Roll back T }

else if (WTS(x) > TS(T)) {
Ignore w(x) of T & continue T
} else {
allowed w(x) by T
Set WTS(x) = TS(T)
}

- TWTSO Protocol allowed to execute (no RB(s)) schedules if result of executed schedule view equal serial based on TS ordering. [increase the degree of concurrency

Strict TS ordering [Correct protocol for concurrency control]

- Basic TS ordering (and)

Strict Recoverable Condition

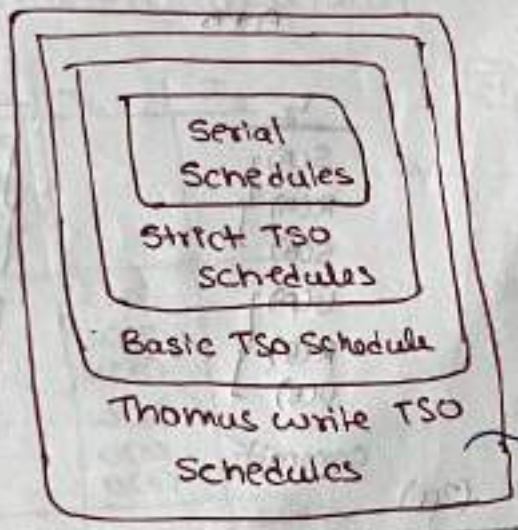
Should satisfy

T_i	T_j
W(A)	
⋮	
C/R	
	R(A)/W(A)

Basic TS ordering protocols	Thomus write TS ordering protocol	Strict TS ordering protocol
• Ensure conflict Serialization (Executed schedule Conflict equal to Serial based on TS ordering)	• Ensures view Serialization (Executed schedule view equal to Serial based on TS ordering)	• Ensure Conflict/view Serialization (Executed schedule Conflict/view equal to serial based on TS ordering)

Basic TS ordering Protocol	Thomus write TS ordering protocols	Strict TS ordering protocol
• Free from Deadlock	• Deadlock free	• Free from Deadlock
• not Free from starvation	• not Free from Starvation	• not free from Starvation
• not guaranted Strict Recoverable	• not guaranted Strict Recoverable	• Guaranked strict Recoverable

Same trans may rollback many time



Shared-Exclusive Locking

[Read only lock]

- Shared lock(S): Shared lock is Required for reading a data item. Many transaction may hold a lock on the same data item in shared locking mode
- Exclusive lock(X): If a transaction is to write an data item, it must have exclusive lock to that data item [Read-write lock] (more degree of concurrency)

Lock compatible table

new Request by transaction T_j	Transaction T_i already hold lock: S	X
S	True	False
X	False	False

S(A): R(A)

X(A): [R(A) W(A)]

2 phase locking protocol [2PL]

[ensure Serializability] / [ISOLATION]

- Trans(T) allowed to request lock on any data item in any mode until first unlock done by T.
- lock request allowed only in growing phase
- these rule are followed by individual trans.

Trans(T)

S(A)
X(B)
S(C)
X(D)
U(C) ← 1st unlock
U(A)
U(D)
U(B)

locking phase [growing phase]
lock point (position of last lock / 1st unlock of tran(T))
unlocking phase [shrinking phase]

ex:- T_1 : Transfer 500 from A to B

T_2 : display A+B

T_3 : Set A, B, C as 1000

T_1
X(A) — lockA
R(A)
W(A)
X(B) — lockB
U(A)
R(B)
W(B)
U(B)
Commit
(2PL)

T_2
S(A)
R(A)
S(B)
U(A)
R(B)
U(B)
Commit
(2PL)

T_3
X(A)
W(A)
X(B)
W(B)
X(C)
U(A)
U(B)
W(C)
U(C)
Commit
(2PL)

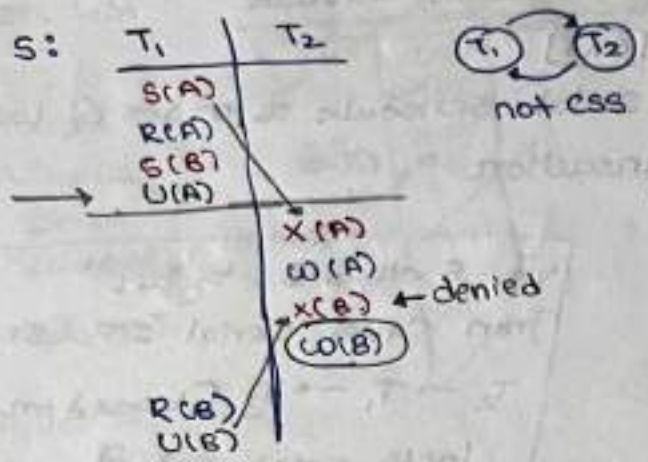
- Concurrent ex. of T_1, T_2, T_3 guaranteed for serializability [every transaction should be in locking & unlocking phase]

** Imp 1 If Schedule (S) not conflict serializable → cycle in precedence graph
Schedule then Schedule not allowed by 2PL

2PL protocol

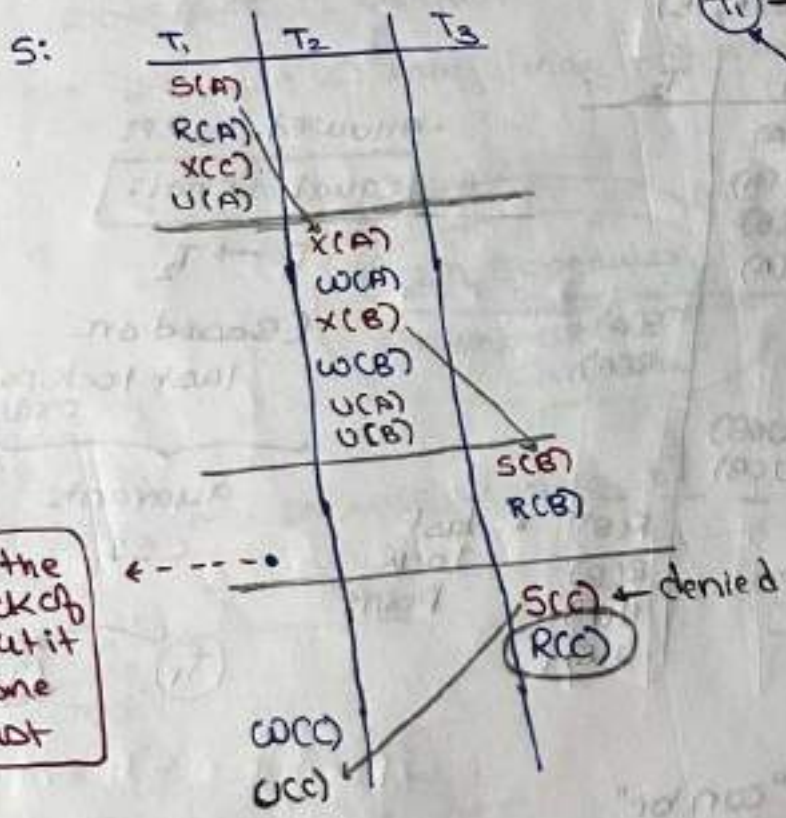
- If every transaction individual follows 2PL rule then every schedule that will be generated will be conflict Serializable & the equivalent serial Schedule is in the order of lock point.

ex: $S: R_1(A)\ W_2(A)\ W_2(B)\ R_1(B)$



- not CSS guaranted not allowed to execute by 2PL

$S: R_1(A)\ W_2(A)\ W_2(B)\ R_3(B)\ R_3(C)\ W_1(C)$



- not CSS
- not allowed by 2PL

need the unlock of C but it is done at last

- S is not conflict serializable then a schedule S not allowed by 2PL

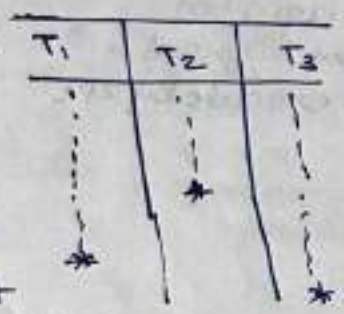
- Schedule S is not 2PL means that S can't be produced by 2PL Scheduler.

② if Schedule (S) allowed by 2PL then schedules (S) is conflict serializable schedule

and

Conflict equal serial schedule is order of lock points of transaction

executed by 2PL (S):

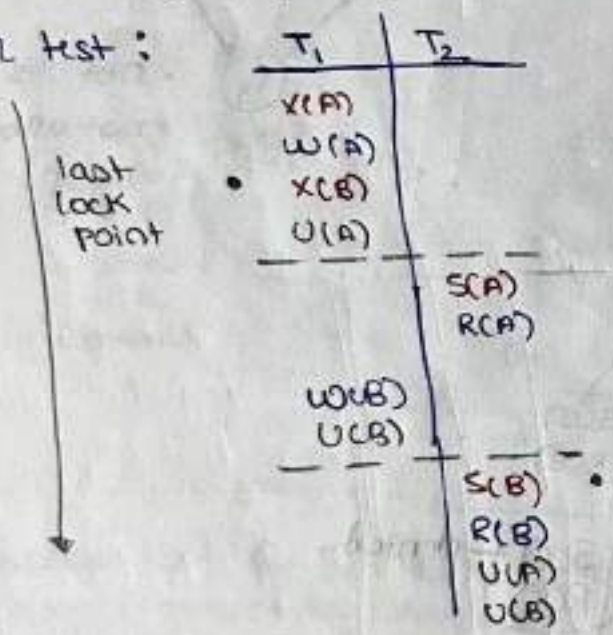


- if S allowed by 2PL Then equal serial schedule $T_2 \rightarrow T_1 \rightarrow T_3$ [based on lock point order]

Lock point order = Topological order

* S: $W_1(A)\ R_2(A)\ W_1(B)\ R_2(B)$

2PL test:



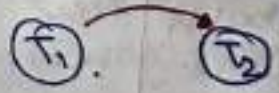
- Allowed by 2PL

equal serial:

$T_1 \rightarrow T_2$

(Based on last lock point order)

guarante CSS



ex: given a schedule S "can be"

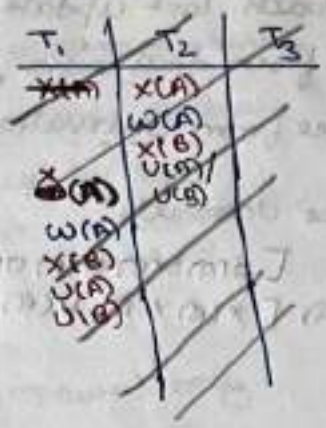
(or) "can't be" generated by 2PL

Try to generate S while following 2PL Rule for each transaction

- serial schedule can be found using lock point

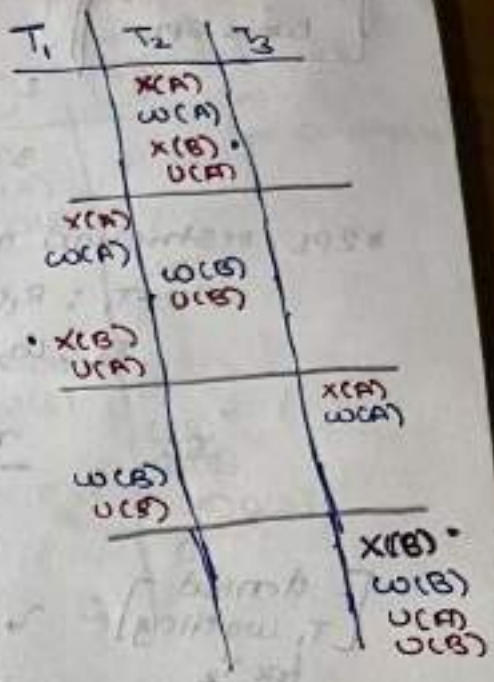
S : $w_2(A)\, w_1(A)\, w_2(B)\, w_3(A)\, w_1(B)\, w_3(B)$

2PL test :

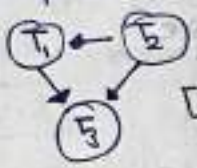


S allowed by 2PL equal Serial Schedule

$T_2 \rightarrow T_1 \rightarrow T_3$



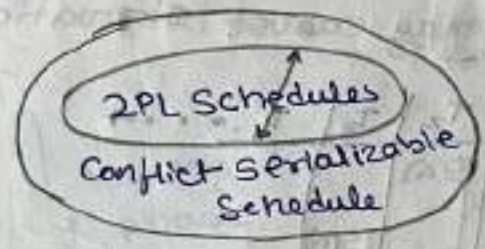
⇒ Conflict SS



$[T_2 T_1 T_3]$

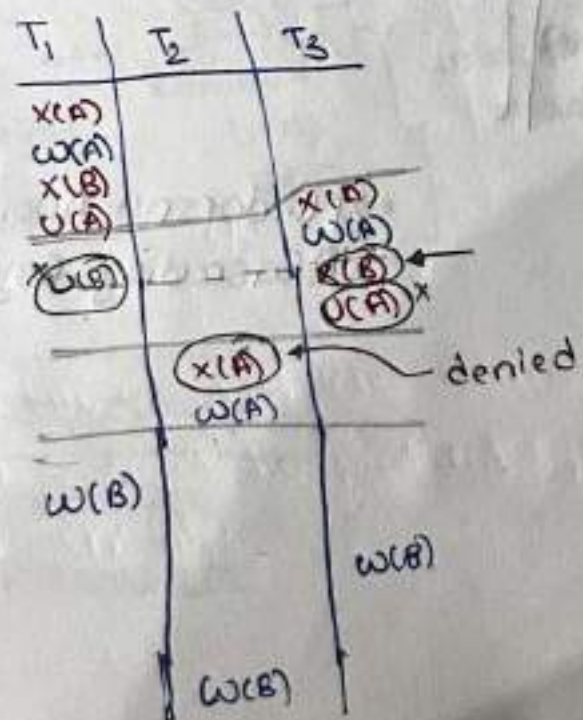
3. • If schedule (S) is conflict serializable

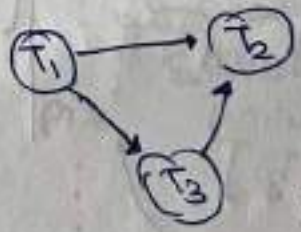
Then Schedule (S) may / may not allowed by 2PL



- Every 2PL schedule is conflict serializable
- not every conflict serializable are allowed by 2PL

4. S: $w_1(A)\, w_3(A)\, w_2(A)\, w_1(B)\, w_3(B)\, w_2(B)$





- S is CSS
- not allowed by 2PL

Limitation of Basic 2PL

1. not free from irrecoverable, cascading Rollback, lost update problems (doesn't ensure)
2. not free from deadlocks
3. not free from starvation

→ suffer from

- 2PL restriction may cause deadlock

T_1 : $R_1(A)$ $R_1(B)$ [S(A) S(B) U(A) U(B)]
T_2 : $W_2(B)$ $W_2(A)$ [X(B) X(A) U(B) U(A)]

S:

T_1	T_2
S(A)	
R(A)	
	X(B)
	W(B)
S(B)	
R(B)	
X(A) ← denied [T_2 waiting for T_1]	
W(A)	

[denied T_1 waiting for T_2]

- Deadlock possible

- 2PL restriction may cause starvation

T_1	T_2	T_3	T_4 ... →
	S(A)		
denied → X(A)		S(A)	
[T_1 waiting for T_2]	U(A)		
X(A)			S(A)
denied		U(A)	
[T_1 waiting for T_3]			
X(A)			
:			
↓			
denied [T_1 waiting for T_4]			

- T_1 suffering from starvation

- 2PL doesn't Avoid Cascading Rollback

[3] 2PL restriction not Sufficient for Strict Recoverable

S: $w_1(A)\ R_2(A)\ w_1(B)\ w_2(B)\ C_2\ C_1$

- S is not Recoverable schedule
- but may allowed by 2PL
- 2PL can generate Irrecoverable Schedule

T_1	T_2
X(A)	
W(A)	
X(B) U(A)	
	S(A)
	R(A)
W(B)	
U(B)	
	X(B)
	W(B)
	U(A) U(B)
	C_2
C_1	

S is 2PL but S is Irrecoverable

Strict 2PL

1. Basic 2PL :-
   - lock request allowed only in growing phase of transaction

and

2. Strict Recoverable :-
   - All exclusive lock of Trans T must hold until commit / Rollback of Trans T

T_1	T_2
X(A)	
W(A)	
⋮	
C/R	
U(A)	
	R(A)/W(A)

Strict Rec. ⊄ 2PL

[Strict 2PL transaction]

Trans (T_i)
- X(A)
- R(A)
- W(A)
- S(B)
- R(B)
- X(C)
- W(C)
- S(D)
- R(D)

locking phase

- U(B)/U(D) Commit
- U(A)
- U(C)

unlocking phase

← unlock Shared lock before commit but not exclusive

- Strict Recoverable
- Serializable
- but Deadlock & Starvation may occur

2PL + Recoverability
- Strict 2PL
- Rigrous 2PL

- S is Allowed by Strict 2PL → S is Cascadeless Recoverable

&

- S is not cascadeless Recoverable → S is not Allowed by Strict 2PL

Regorious 2PL

① Basic 2PL : lock request allowed only in growing phase of transaction

and

② All locks (exclusive/shared) of Trans T must hold until commit/Rollback of Trans T.

[Regorious 2PL trans]

Trans(T_1)

X(A)
R(A)
W(A)
S(B)
R(B)
X(C)
W(C)
S(D)
R(D)
Commit
U(A) U(B)
U(C) U(D)

locking phase

lock point →

unlocking phase

Concurrency Control

- restrict concurrent execution which causes inconsistency
- allow concurrent ex which not causes inconsistency

③ **Conservative 2PL** (prevent DL)

- Transaction (T) must lock all required data item in initial stage

Trans (T)

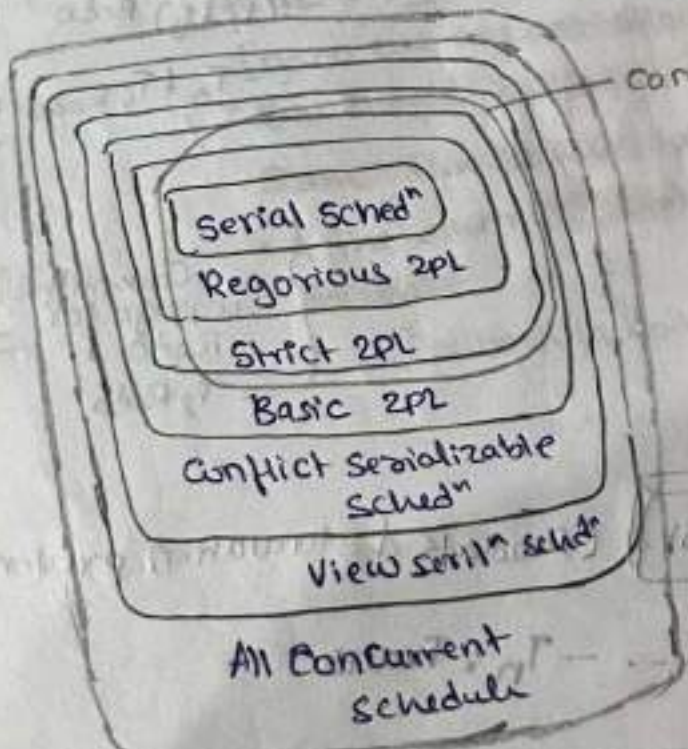
S(A)
X(B)
S(C)

lock all data item

R(A)
R(B)
W(B)
R(C)
U(A) U(B)
U(C)
Commit

- free from dead lock
- Starvation possible
- less degree of concurrency

	Basic 2PL	Strict 2PL	Rigorous 2PL	Conservative 2PL
1. Guaranteed serializability	yes	yes	yes	yes
2. guaranteed strict Recoverable (Exclusive locks held until commit/abort)	no	yes	yes	no
3. free from Deadlock	no	no	no	yes
4. free from starvation	no	no	no	no
5. equal serial Schedule (2PL)	based on lock point order	based on lock point order	based on Commit order.	based on lock point order



Crash Recovery :-

- All transaction which are in committed until previous checkpoints are perform redo
- all uncommitted transactions perform undo {entire schedule}

Checkpoints :- All committed trans until previous checkpoint perform Redo.

log file :-

1. T_1 begins
2. T_2 begins
3. T_2 commits
4. T_3 begins
5. T_3 commit
6. checkpoints — all committed trans until previous checkpoint perform redo

T_2 & T_3 go for Redo

7. T_4 begins
8. T_5 begins
9. T_6 begins
10. T_5 commit
11. T_6 commit
12. checkpoint

already done (T_2 & T_3) Redo

T_5 & T_6 Redo

13. T_7 begins
14. T_8 beigns
15. T_8 commit
16. System crash

Committed will go for Redo

So T_8 Redo

undo of T_1, T_4, T_7

Wait - Die protocol (Deadlock & starvation prevention protocol in locking protocol)

$T_1, T_2, T_3 \dots\dots - T_n$: Trans

T_i T_j Trans such that $TS(T_i) < TS(T_j)$

Older younger.

(1) $T_i \longrightarrow T_j$ if T_i required resource holded by T_j then T_i allowed to wait

old young

(1)

	T_i	T_j
		X(A) ← Hold
request →	S(A)	

↑ Allowed to wait

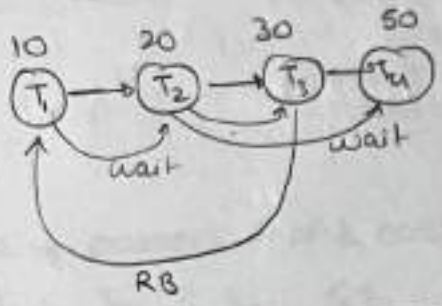
(2)

T_i ← T_j

old young

if T_j required data (resource) Holded by T_i ∴ Then T_j Roll back (and) T_j restart with Same T_s value

- Deadlock occur whenever cyclic Redundancy



Wait-die

O → y	✓	O ← y	X

T_s value

hold →

	T_i	T_j
	X(A)	
		S(A)

Rollback T_i & restart with same T_s value

main Reason to avoid Starvation.

Wound-wait Same as above protocol. then only change comes in 1st condition [now we are not allowed to wait]

- you should Rollback & Restart with Same Timestamp value
- & the others (2nd condition) is allowed to wait

[Rollback in both case is done for younger]

- After group by the order of the attribute matters.

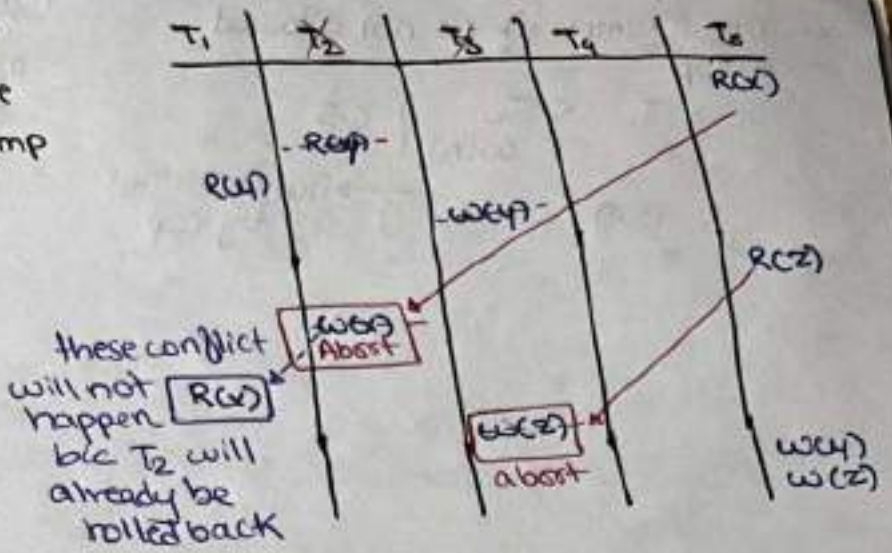
necessarily precedes the tuple (5,5,5)

<0,90> } before this
<5,5,6>

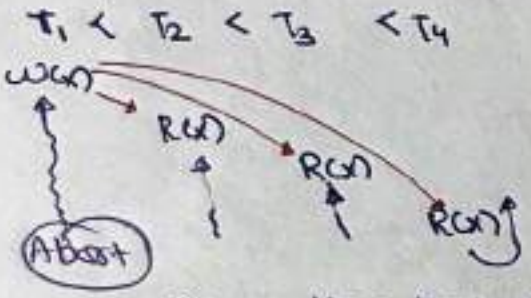
• Timestamp Example

① no of transaction are aborted by timestamp scheduler?
with timestamp
1, 2, 3, 9, 5

So, T_2 & T_3 will be rolled back



② Timestamp protocol Avoid cascading Rollbacks (no)



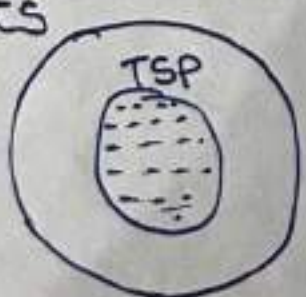
"Cascading Rollback"

ex: A write is only performed if transaction has a timestamp ≥ the read timestamp for the data item True.

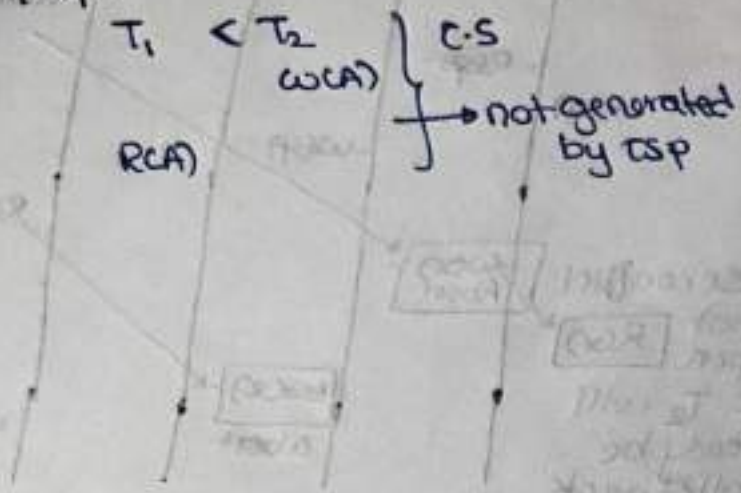
$TS(T) \geq RTS(x)$

Successful iff
$TS(T) \geq RTS(x)$ &
$TS(T) \geq WTS(x)$

ex: Timestamp generates (only) conflict serializable schedules.



ex: one example of CS not allowed by TSP



note
- using Thomas write Rule we can Allow/generate some view serial^ but non-CS schedule.