Introduction

→ Data : facts, figs, statistics, observations, Records
→ Information : meaningful, useful, relevant data.
→ DB : Collection of related data
→ DBMS : Software, store, access, retrive, update, create, maintain, delete securely.

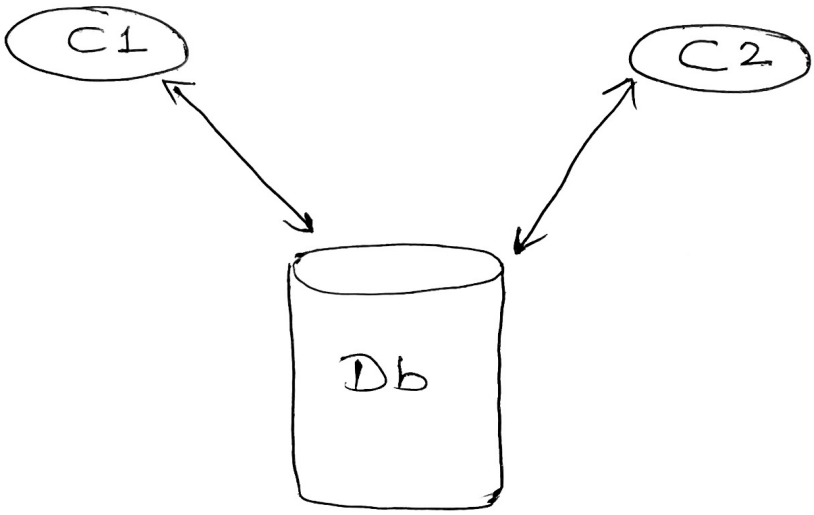
→ File System
- → Data Redundancy
- → Data Inconsistency
- → Difficult Access
- → Data Isolation
- → Security
- → Atomicity
- → Concurency
- → Simple

→

OLAP	OLTP
1) Online Analytical Processing	1) Online Transaction Processing
2) Historical data	2) Operational Current data
3) Analyze large volumes	3) Manage & process real-time transactions
4) Large storage Requirement	4) Small storage Requirement
5) Trend analysis, behavior prediction	5) payment processing & order mgmt & processing.

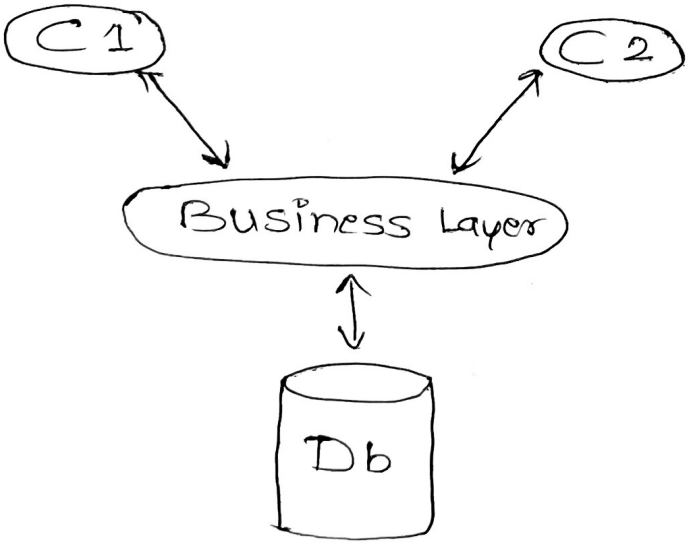
- 2-Tier Architecture :

→ client server Architecture
→ client tier & Data tier
→ Easy but slow
→ Direct connect
→ Less secure



- 3-Tier Architecture :

→ client tier, Business tier, data tier
→ Complex but fast
→ No direct connect
→ More secure

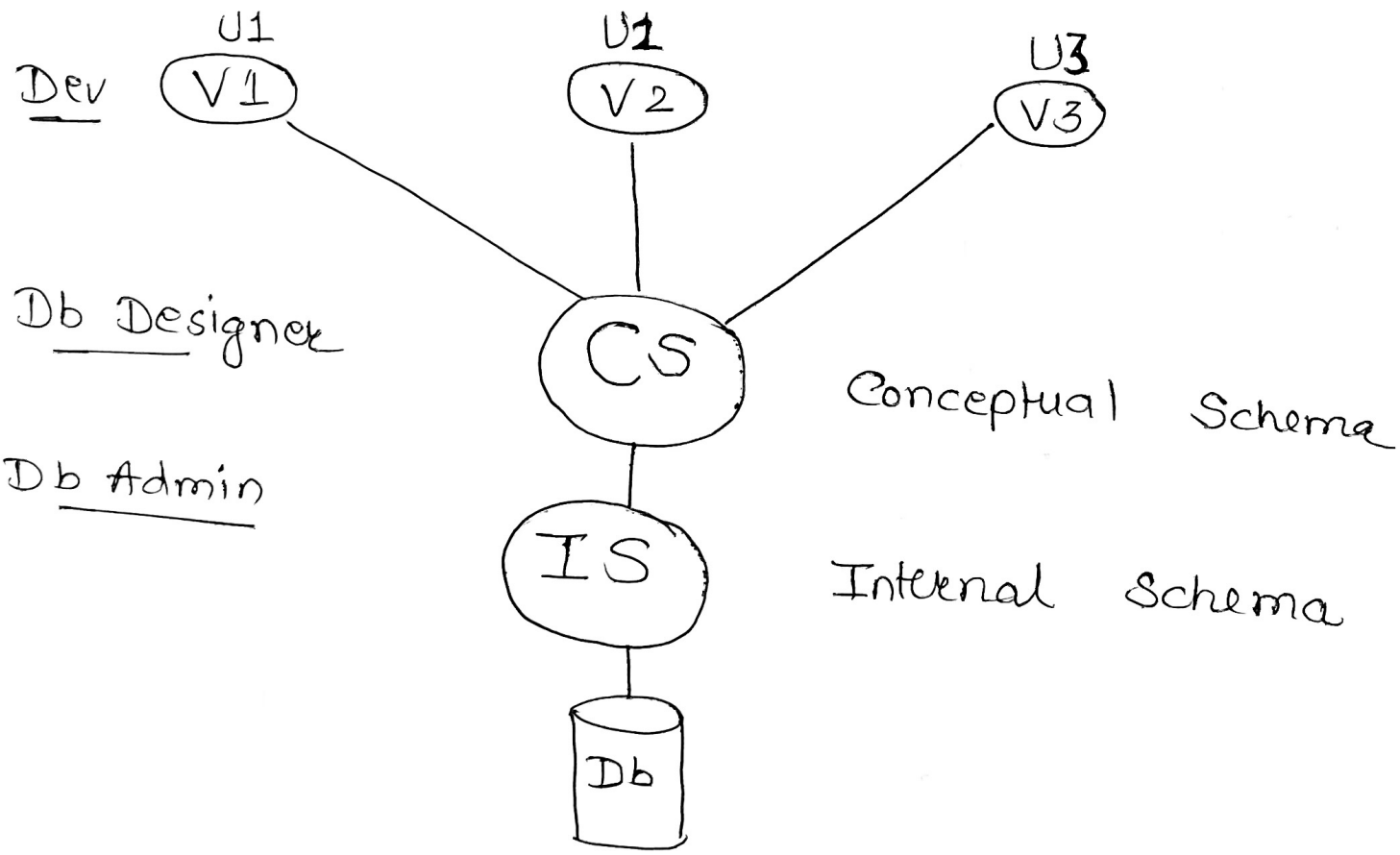


Schema

- logical Representation
- Blueprint of a database
- Organized?
- Relation & Constraints
- Implementation by SQL (RDBMS)

Instance: Info in Db at a given moment.

• 3 Schema Architecture:



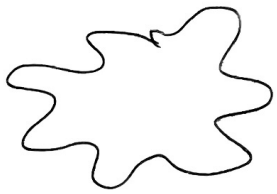
" Data Independence "

- Physical
- Conceptual

Data

Structured

Unstructured



E-R Diagram

- Easy to make
- Easy to understand
- Entity (E)
- Relationship (R)
- Attributes

Student

- object with existence
  - physical
  - logical
- has attributes
- Different from others

Entity

Roll no : 1
Name : Shridhar

Entity type

Student (Roll no, Name)

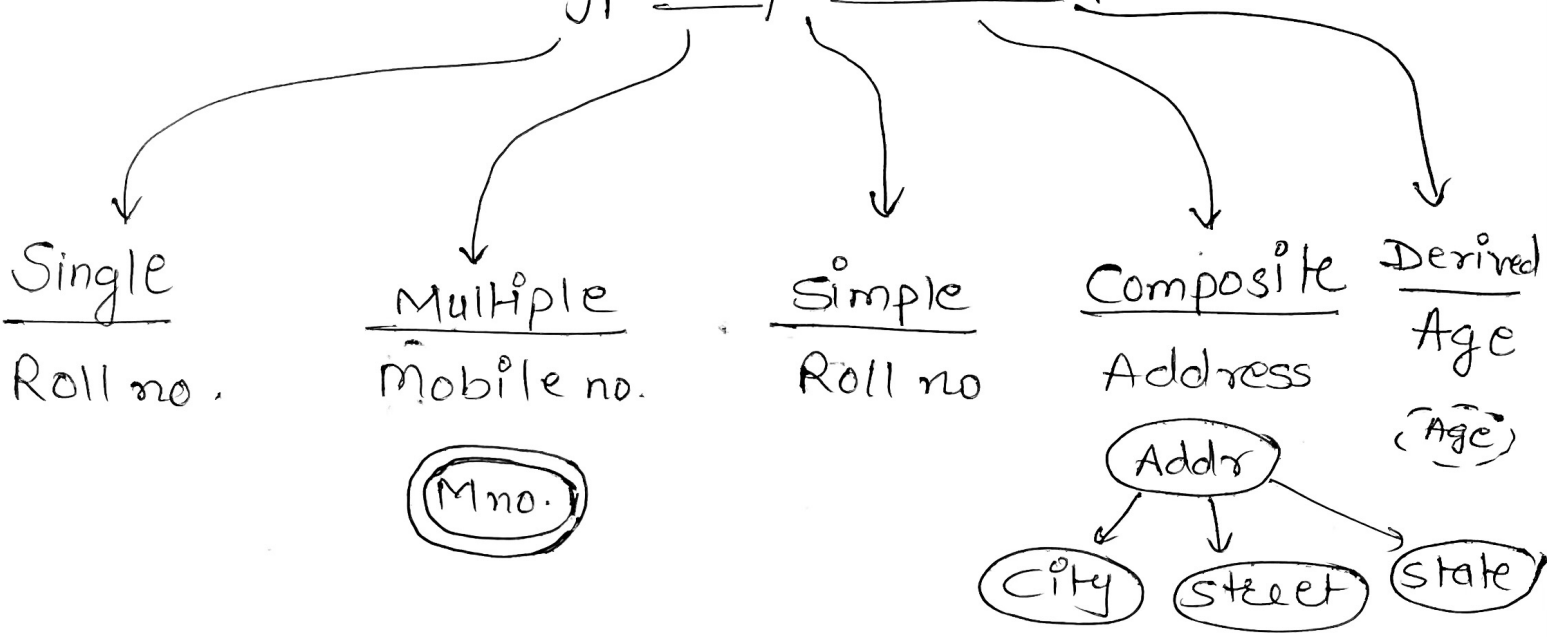
Entity set

S1
S2
S3

→ Attributes

- It is properties / characteristics of entity.
- domain (Roll no :- 2 +ve integers)

Types of Attributes



Descriptive attr → Relationship

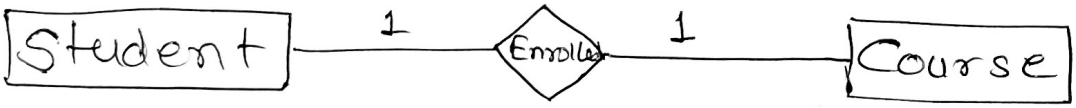
Key attr → Roll no. → uniqueness

→ Relationship → association

diamond shape

→ Degree / cardinality.

1) one to one



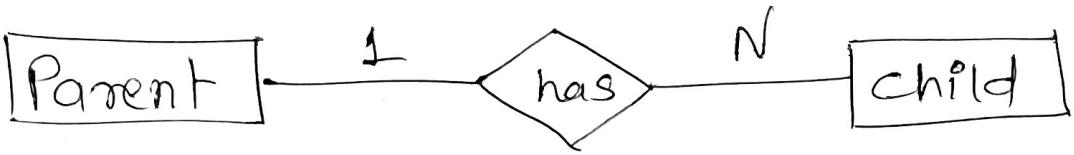
Roll no.	Name
1	A
2	B
3	C

RT

Rollno	Cid
1	101
2	102
3	103

Cid	Name
101	X
102	Y
103	Z

2) One to Many



Id	Name
1	A
2	B
3	C

Id	Cid
1	101
1	102
2	103
3	104
3	104

Cid	Name
101	X
102	Y
103	Z
104	U
105	V

Id	Cid	Name
1	101	X
1	102	Y
2	103	Z
3	104	U
3	105	V

3) Many to Many

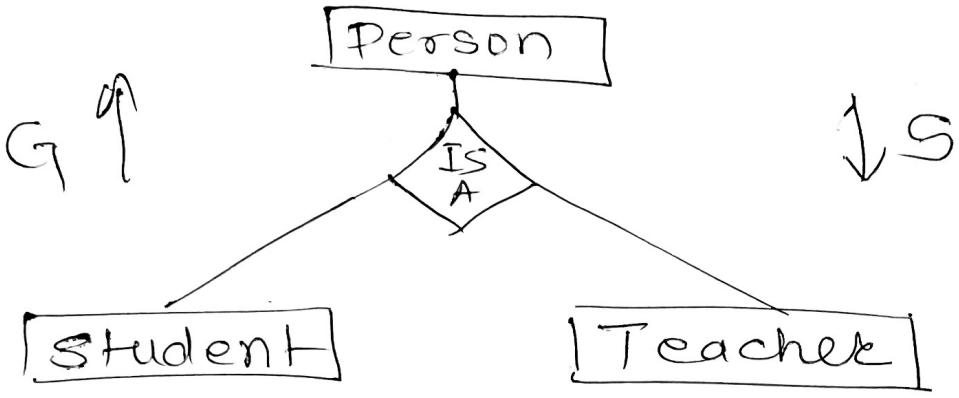
Rollno.	Name
1	A
2	B
3	C

Roll no.	Sid
1	S1
2	S1
1	S3
3	S2
4	S1
5	S2
4	S3
3	S1

Sid	Name
S1	IT
S2	Biology
S3	Maths
S4	Hindi

→ Generalization

- low level → high level
- Bottom Up approach



→ Specialization

- high level → low level
- Top-down approach.

- Integrity Constraint
  - Domain → Type of data & limit, check
  - Entity Integrity → PK ≠ NULL
  - Referential Integrity → FK
  - key → Uniqueness

- Candidate keys (Uniqueness)

Student

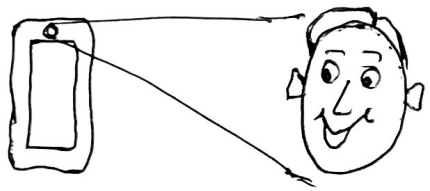
↓

Rollno, Ph.no, Govt.Id, Admissin no.

↓ ↓

PK AK

- Primary key : As per the scenario we select the most eligible key as PK out of all CKs.



(Unique + NULL(x))

Only '1' → PK allowed.

- foreign key

PK

Rollno	Name
1	A
2	B
3	C
4	D

Sid	Name	Roll_no
101	X	1
102	Y	2
103	Z	3
104	V	5

Rollno (sports) references to student (Rollno)

Operations → Insert, Delete & update

- **Super Key → CK + attr (Normal)**

CK = emp_id

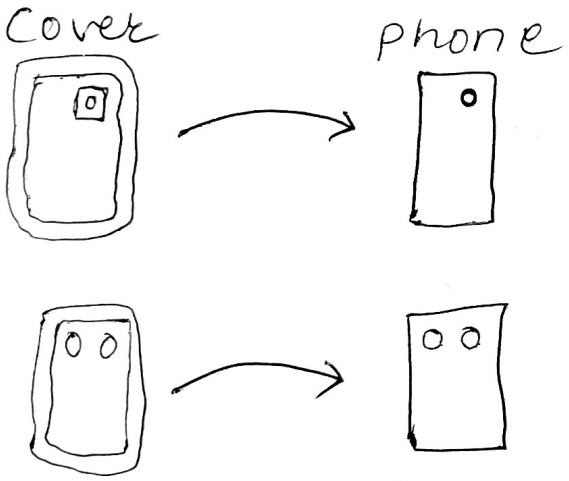
↓

emp_id + name → SK

emp_id + location → SK

name + location → xSK

- **Functional Dependency**



Determinant → α → determines → β ← Dependent

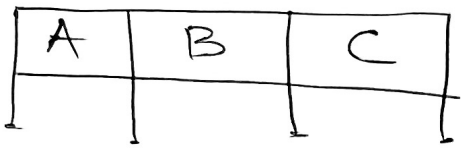
§ $R(\alpha) \rightarrow R(\beta)$

$R_1(\alpha) = R_2(\alpha)$

$R_1(\beta) = R_2(\beta)$

A	B
1	5
1	5
2	6
3	7
3	7

- Closure



$A \rightarrow B$

$B \rightarrow C$

$A^+ = ?$

A

AB

ABC

→ Armstrong Axioms

1) Reflexivity: (Trivial Property)

$A \rightarrow B$; $B \subseteq A$

(B is subset of A)

2) Augmentation: Adding attr.

$A \rightarrow B$ ✓

add X

$AX \rightarrow BX$ ✓

3) Transitivity: $A \rightarrow B$ ✓ & $B \rightarrow C$ ✓

then

$A \rightarrow C$

- Secondary Rules:

Union	Decomposition	PseudoT	Composition
$A \rightarrow B$	$A \rightarrow BC$	$A \rightarrow B$	$A \rightarrow B$
$A \rightarrow C$	$(A \rightarrow B$	$BC \rightarrow D$	$C \rightarrow D$
$(A \rightarrow BC)$	$A \rightarrow C)$	$(AC \rightarrow D)$	$(AC \rightarrow BD)$

Q → R(A B C D)

FD: { A → B , B → C , C → D }

$A^+ = A$ → AB → ABC → ABCD

'A' becomes one of the CK set

$B^+ = B$ → BC → BCD

B, C, D can't be CK.

$(AB)^+ = AB$ → ABCD

so (AB) becomes superkey.

- Minimal Cover : To remove redundancy.

{ A → B, C → B, D → A B C, A C → D }

D → A D → B D → C

- A → B : $A^+ = A$
- C → B : $C^+ = C$
- D → A : $D^+ = D B C$
- D → B : $D^+ = D A B C$
- D → C : $D^+ = D A B$
- AC → D : $(AC)^+ = A C B$

$(AC)^+ = ABCD$

$A^+ = AB$ $C^+ = CB$

Can't remove neither A nor C

- Equivalence of FD:

$FD1 = \{ A \to B, B \to C \}$

$FD2 = \{ A \to B, B \to C, A \to C \}$

If $\left\{ \begin{array}{c} FD1 \subseteq FD2 \\ \& \\ FD2 \subseteq FD1 \end{array} \right\}$

then $FD1 = FD2$

(1) FD1 covers FD2 (Yes or NO)

$A^+ = ABC$ $(A \to B)$ ✓ $(A \to C)$ ✓

$B^+ = BC$ $(B \to C)$ ✓

(2) FD2 covers FD1 (Yes or NO)

$A^+ = ABC$ $(A \to B)$ ✓

$B^+ = BC$ $(B \to C)$ ✓

$\therefore \quad FD1 = FD2$

Decomposition

A	B	C
1	X	P
2	X	P
1	Y	Q

AB	
1	X
2	X
1	Y

AC	
1	P
2	P
1	Q

A	B	C	
1	X	P	
1	X	Q	X
2	X	P	
1	Y	P	X
1	Y	Q	

Lossy

AB	
1	X
2	X
1	Y

BC	
X	P
Y	Q

A	B	C
1	X	P
2	X	P
1	Y	Q

Lossless

- Dependency Preserving

FDs := { A → B , B → C }

R(ABC)

R1(AB)

✗ A → A , B → B ✗
✓ A → B
✗ B → A

R2(BC)

B → B , C → C ✗
B → C ✓
C → B ✗

R1(FD) ∪ R2(FD) = FDs

$FDs = \{ A \to B,\ B \to C,\ A \to D \}$

$R(ABCD)$

$A^+ = ABCD$
$B^+ = BC$
$C^+ = C$
$D^+ = D$

$R_1(AB)$

$A \to B$ ✓

$R_2(BCD)$

$B \to C$ ✓
$C \to D$ ✗
$B \to D$ ✗
$D \to C$ ✗
$D \to B$ ✗
$C \to B$ ✗

$\cup$

$(A \to B) \cup (B \to C)$

$\neq$

(FDs) as

{Closure w.r.t FDs)

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

} But what about 'D' $A \to D$?

- Normalization
  - To Remove Redundancy
    - 1NF
    - 2NF
    - 3NF
    - BCNF

⇒ 1NF
- (NO Multivalued Attr)
- (PK)
- (Distinct C_names)

empid	Name	Projects
1	ABC	X / Y / Z

empid	Name	Projects
1	ABC	X
1	ABC	Y
1	ABC	Z

empid	Name	P1	P2	P3
1	ABC	X	Y	Z

3rd approach

empid	Name
1	ABC

fk

empid	Project
1	X
1	Y
1	Z

→ 2NF

↳ In INF

↳ No partial dependency

↓

A	B	C
1	X	P
2	Y	Q
1	Z	R
3	Z	R
4	Z	R
5	Z	R

FDs: {AB → C, B → C}

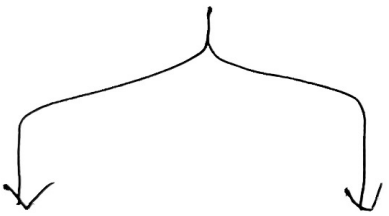
(AB) → key

Now here;

B → ← Non key Attr.

↑

Part of key



A	B
1	X
2	Y
1	Z
3	Z
4	Z
5	Z

B	C
X	P
Y	Q
Z	R

3NF

- In 2NF
- No Transitive Dependency

A	B	C
1	b	x
2	b	x
3	b	x
4	c	y
5	c	y
6	c	y

FDs: { A → B, B → C }

A → key

NK → NK (TD)

CK, SK 'or' PA

A	B
1	b
2	b
3	b
4	c
5	c
6	c

B	C
b	x
c	y

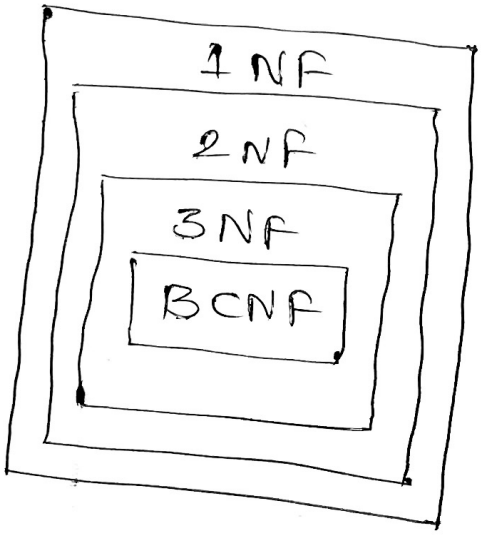
Lossless

→ BCNF (Boyce Codd Normal form)

- → In 3NF
- → L.H.S (α) of each FD must be CK or SK

eg:- CK: {A, B}

FDs: {A → C, A → B, B → D, B → A}



→ 4NF → In BCNF

- → No Multivalued Dependency.

A	B	C
X	1	4
X	2	5
X	3	6

(A →→ B)

for 'X' B & C have multiple values

→ 5NF → In 4NF

- → Should follow Lossless Decomposition

⇒ Joins ⇒ when we need 2 or more tables (Multi table queries)

- Natural Join
- Equi Join
- outer join
  - left
  - Right
  - full
- self join

• Natural Joins

Pk

Rno	Name
1	A
2	B
3	C

student

Pk fk

Sid	Rno
101	1
102	2
103	3

subject

cross product + condition "Student.R_no = Subject.Rno"

R_no	Name	Sid	R_no
1	A	101	1
1	A	102	2
1	A	103	3
2	B	101	1
2	B	102	2
2	B	103	3
3	C	101	1
3	C	102	2
3	C	103	3

1 A 101

2 B 102

3 C 103

* Equi Join

Pk

Student

Rno	Name	Addr
1	A	X
2	B	Y
3	C	Z

PK → Pk

fk

exam

Exno	Loc	Rno
101	X	1
102	V	2
103	W	3

cross product

2nd	1st	Rno	Name	Addr	Exno	Loc	Rno.
✓	✓	1	A	X	101	X	1
		1	A	X	102	V	2
		1	A	X	103	W	3
		2	B	Y	101	X	1
✓		2	B	Y	102	V	2
		2	B	Y	103	W	3
		3	C	Z	101	X	1
		3	C	Z	102	V	2
✓		3	C	Z	103	W	3

Student. Rno = exam. R_no

Student. Addr = exam. Loc

Self Join : Join with self

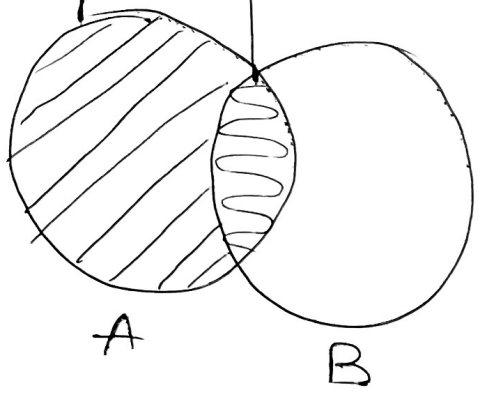
R_no	Sid	Marks
1	101	50
2	101	60
3	102	70
1	103	80

C1 C2

cross product

2nd	1st	R_no	sid	R_no	sid
X	✓	1	101	1	101
		1	101	2	101
✓	✓	1	101	3	102
		1	101	1	103
X	✓	2	101	1	101
		2	101	2	101
		2	101	3	102
		2	101	1	103
		3	102	1	101
	✓	3	102	2	101
		3	102	3	102
		3	102	1	103
✓	✓	1	103	1	101
		1	103	2	101
		1	103	3	102
X	✓	1	103	1	103

- Left outer join
  - Natural join
    +
  - Left exclusive data/Records



R_no	Name
1	A
2	B
3	C
4	D

Sid	R_no
101	1
102	2
103	3

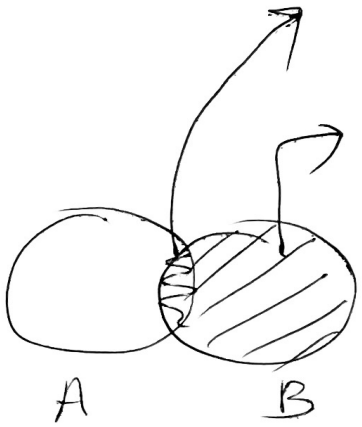
Left outer join.

NJ + LER

1	A	101
2	B	102
3	C	103
4	D	—

Right outer Join

↳ Natural Join
+
Right Exclusive Records



Rno	Spd
1	101
2	102
3	103

Sid	Name
101	A
102	B
103	C
104	D

ROJ

1	101	A	NJ
2	102	B	NJ
3	103	C	NJ
-	104	D	RER

- Full outer Join

↳ LOJ ∪ ROJ

- **Relational Algebra**
  - Procedural
  - Formal — Query Language
  - It helped in making of SQL

Query Language

- Relational Algebra
  - What to do
  - How to do
  - (Procedural)
    - Operator: Basic & Derived.
    - Operand: (Table/Relation)
- Relational Calculus
  - (Non-P)
  - Only kya karna hai

- **Basic op**
  - Projection (π)
  - selection (σ)
  - Cross Product ($\times$)
  - Union ($\cup$)
  - Rename (ρ)
  - set difference ($-$)

- **Derived op**
  - Join ($\bowtie$)
  - Intersect ($\cap$)
  - Division ($/$)

Projection (Π) (Column / vertical selection)

" $\Pi_{CN}(TN)$ "

R_no	Name	City
1	A	X
2	B	Y
3	C	Z

student

$\Rightarrow \Pi_{Name}(Student)$ → Name column of student table

Selection (σ) (Row / Horizontal selection)

" $\sigma_{condition}(TN)$

eg: $\sigma_{R_{no}=1}(student)$

Combine :- $\Pi_{Name}(\sigma_{R_{no}=1}(student))$

Comparisons :- $=, \neq, <, >, \leq, \geq$

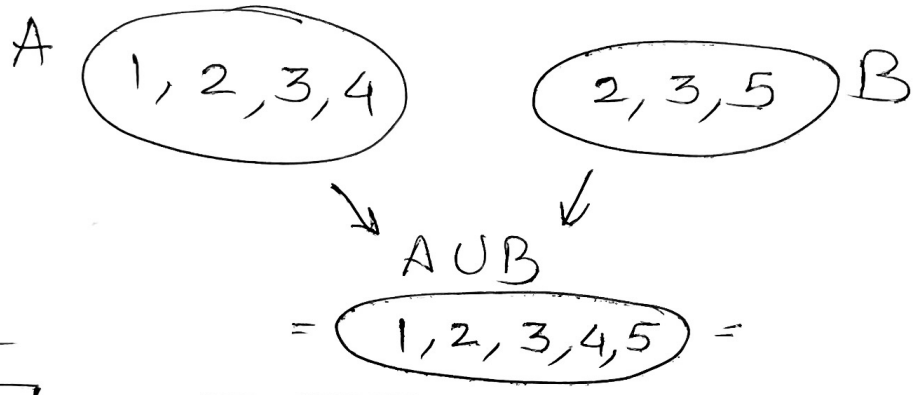
Connect :- $\wedge, \vee, \neg$

Union :- binary operator.

A ∪ B

T_1 T_2

- both A & B should have
  - → same no. of column
  - → same domain



Cricket

CId	Name
1	A
2	B
3	C

HId	Name
1	A
5	D
6	E

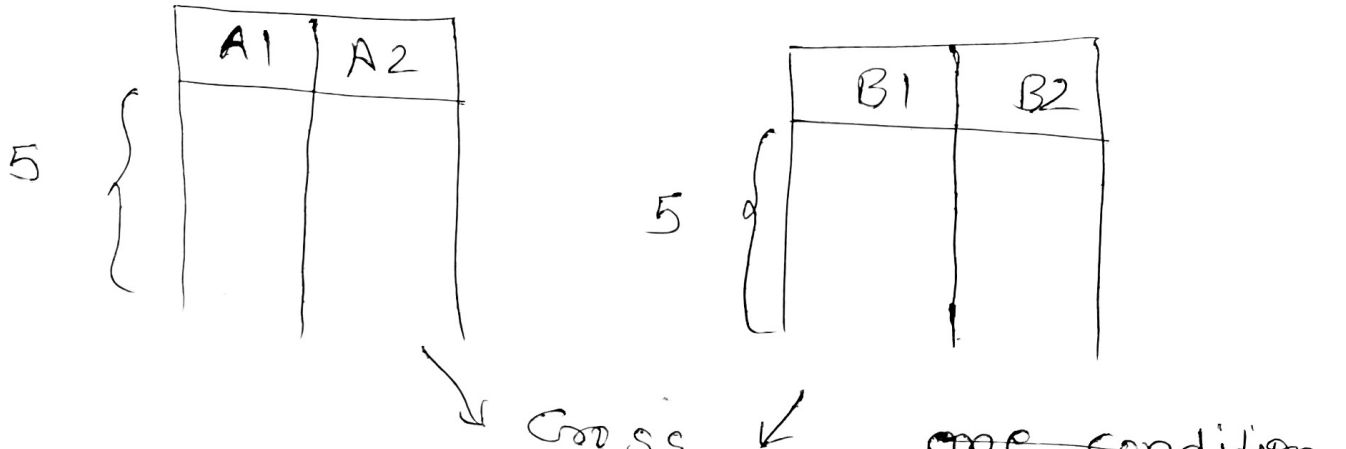
Hockey

C ∪ H →

CId	Name
1	A
2	B
3	C
5	D
6	E

} → common (counted once)

Cross product : A x B

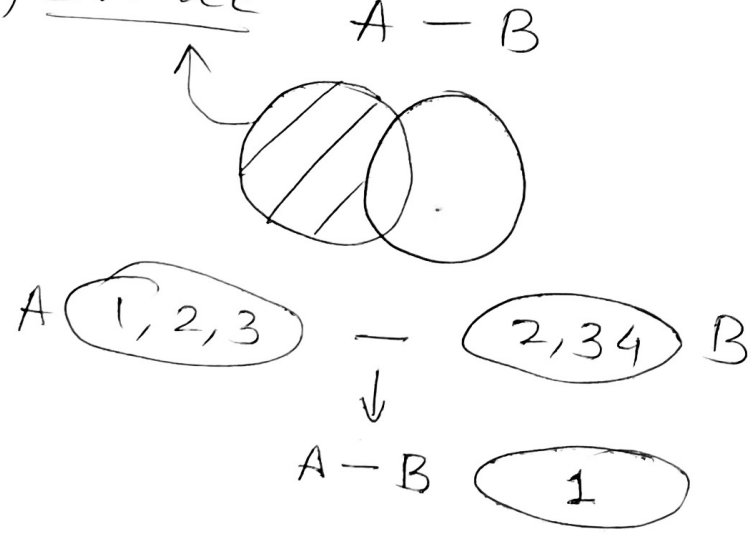


A_1 A_2 B_1 B_2 $(2+2=4)$

$5 \times 5 = 25$

All combination

set difference



→ No. of column & domain of A, B should be same.

CId	Name
1	A
2	B
3	C
4	D

A

HID	Name
2	B
3	X
5	Y
6	Z

B

(A − B) →

CId	Name
1	A
4	D

- Division : Derived (A/B or A ÷ B)

A

P	Q
X	1
Y	2
X	2
Z	4

B

Q
1
2

A/B →

A(P,Q) / B(Q)

P
X

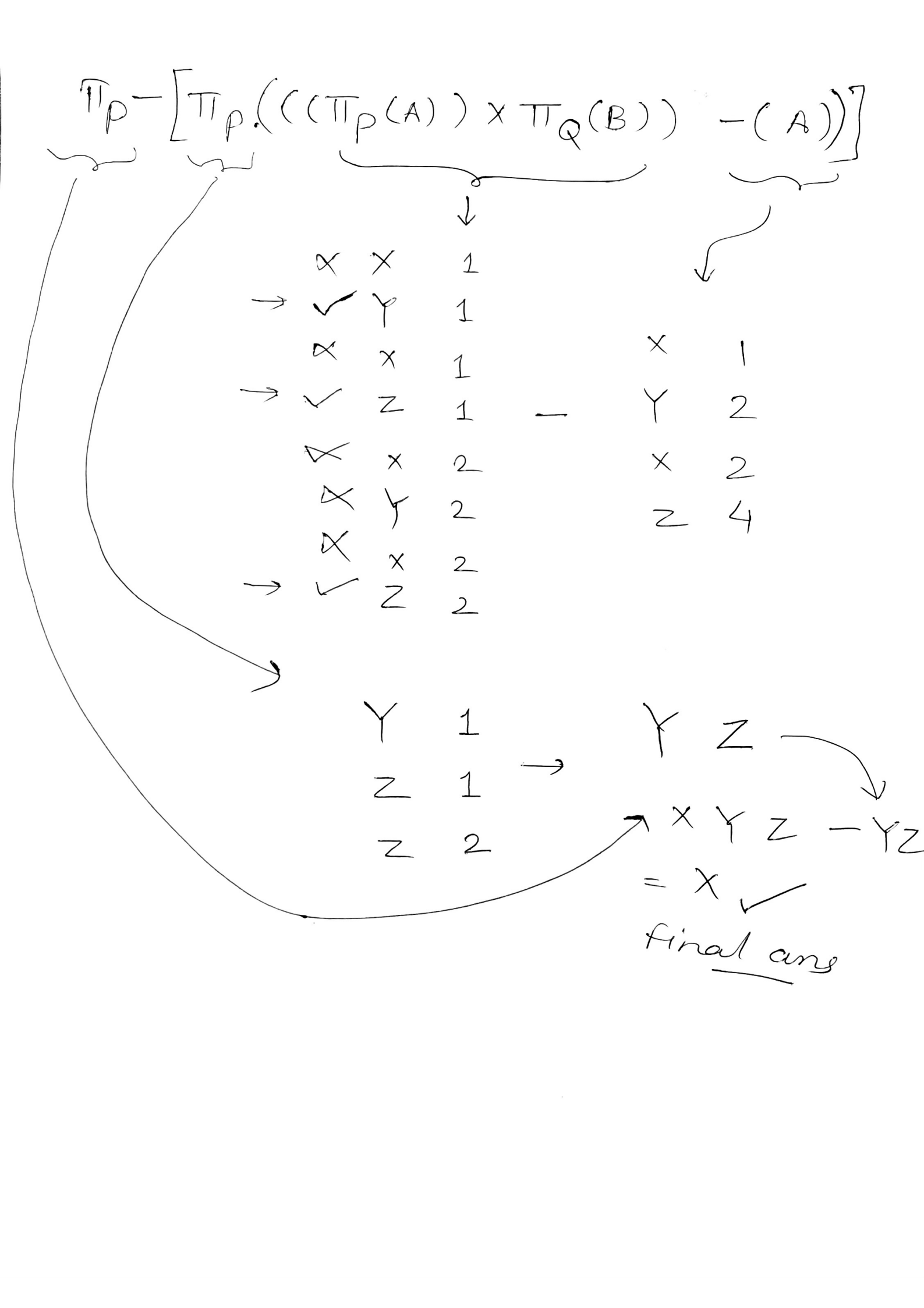
It returned P values that matches with all Q values

X → 1, 2

Y → 2

Z → 4

$$\pi_P - [\pi_P((((\pi_P(A)) \times \pi_Q(B)) - (A))]$$



• Rename : " ρ " Rho

$\rho_{Rename}(Table)$

↳ eg :- Rename student table to Alumni

$\Rightarrow \rho_{Alumni}(Student)$

↓ In more detail we can change the column name as well.

$\rho_{Alumni(Ano)}(Student(RollNo))$

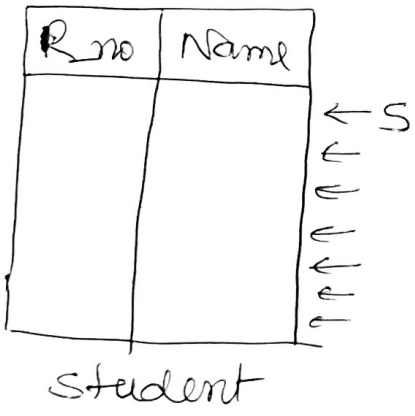
- Relational Calculus
  - Non-procedural query language
  - Only what to do
  - Tuple RC (Row wise)
  - Domain RC (column wise)

⇒ Tuple RC

Query ⇒ $\{t \mid C(t)\}$

- t → final Tuple
- $C(t)$ → Condition / Rule.

- So the tuples 't' will be seen in Result, which satisfies C(t).



$\{S.Name \mid Student(S)\}$

$\{S.Name \mid Student(S) \wedge S.R_{no} = 1\}$

⇒ $\wedge, \vee, \neg, \exists, \forall$

⇝ Unsafe Expression:

$\{S.Name \mid \neg Student(S)\}$

↳ ∞

- Domain Relational Calculus
  - → Column / attributes
  - → $\{ c1, c2, c3 \ldots cn \mid C(c1, c2, c3 \ldots cn) \}$

Eg: $\{ (Rno, Name) \mid Student(Rno, Name) \wedge Rno = 1 \}$

- $(Rno, Name)$ → O/p or resultant attr.
- $Student(Rno, Name)$ → I/p Relation
- $Rno = 1$ → selecting a row based the condition

SQL → Structured Query Language

Data — Operations — Db Specific

DDL
- Create
- Drop
- Alter
- Truncate
- Rename

DML
- Select
- Insert
- Delete
- Update

DCL
- Grant
- Revoke

TCL
- Commit
- Roll back

① CREATE DATABASE students;

② CREATE TABLE students.academic(
RollNo INT PRIMARY KEY
Name Varchar(100)
Marks int);

INT, SMALLINT, BIGINT, FLOAT, VARCHAR, CHAR

③ ALTER Table students.academic
ADD PhoneNumber INT;

④ ALTER Table students.academic
DROP column phoneNumber;

⑤ ALTER Table academic
RENAME column Name TO SName;

⑥ ALTER TABLE academic
RENAME TO study;

⑦ DROP Table academic;

⑧ DROP Database students;

(9) Create Table emp(
emp_id int primary key,
FOREIGN KEY(Dep_id) REFERENCES Dept(Dep_id));

(10) Insert into students (RollNo, Name, Marks)
Values
(1, x, 10)
(2, Y, 20)
(3, z, 30);

(11) DELETE FROM students
where RollNo = 1;

(12) DELETE FROM students;

(13) Select RollNo from students where C
↑ ↑ ↑
π Table σ

(14) select '*'
↖ for all attr.

(15) Select distinct location from Map.
↑
(NO Duplicates)

(16) Select marks + 10 from academic
↓ → ↓
10 → 20
20 → 30
30 → 40

(17) select * from academic where marks > 35

(18) select Name from academic
where marks > 35 and Attendence > 75;

(19) select * from academic
where marks ≥ 35 and marks ≤ 90

↓

where marks between 35 and 90

(20) where marks not between 35 and 90

(21) Select Name from Cricket
Union
select Name from Hockey.

(22) select Name from Cricket
Insect
select Name from Hockey

(23) select Name from Cricket
Except
Select Name from Hockey.

(24) Grant insert, select on student to xyz;

(25) Revoke insert, select on student from xyz;

(26) select Name, sports
from student, sports

(27) where student.Id = Sports.Id

(28) from student natural join sports.
inner
ON student.Id = sports.Id

(29) from student left join sports

(30) from student Right join sports.

(31) from student full outer join sports.

- Aggregate fn

(32) select Count(*) from student

(33) select avg(marks) from student

(34) select sum(marks) from student

(35) select max(marks) from student

(36) select min(marks) from student

(37) select * from student where marks > 35
order by name;

↓

(38) order by name desc;

(39) select count(location) from student
where marks > 35
group by (location)

(40) select Name from student
where Name like '%s'

→ 's%', '_s%', '___', '_s_'

(41) Create view customers as
Id int,
Name varchar(100);

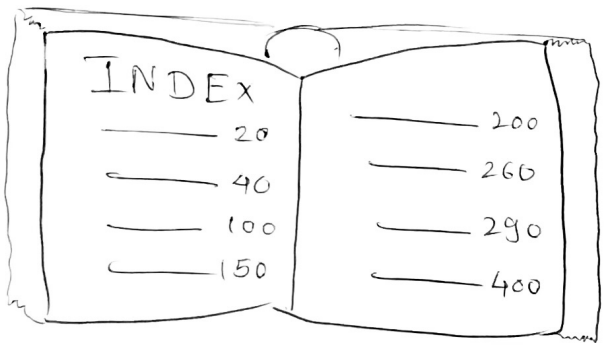
(42) Insert into customers values
(1, A),
(2, B),
(3, C);

(43) Truncate Students;

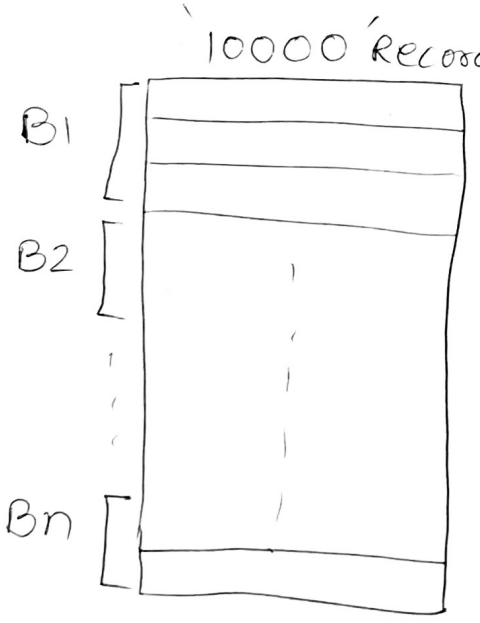
(44) select Name from student
where marks = (select max(marks) from student)

(45) select Name from student where marks = (select marks from student where city = (select city from student where city like 'P___'))

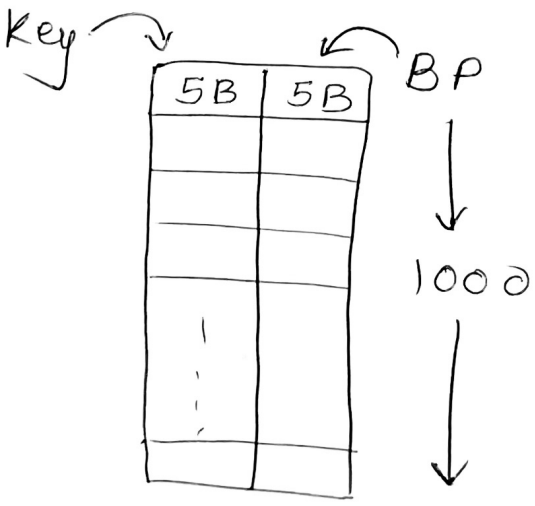
- Indexing



→ sorted
→ keywords ✓
→ x description
→ small number of pages



Block size = 1000B
Record size = 100B
∴ No. of rec/block = $\frac{1000}{100}$ = '10'
Total No. of Blocks = $\frac{10000}{10}$ = '1000'

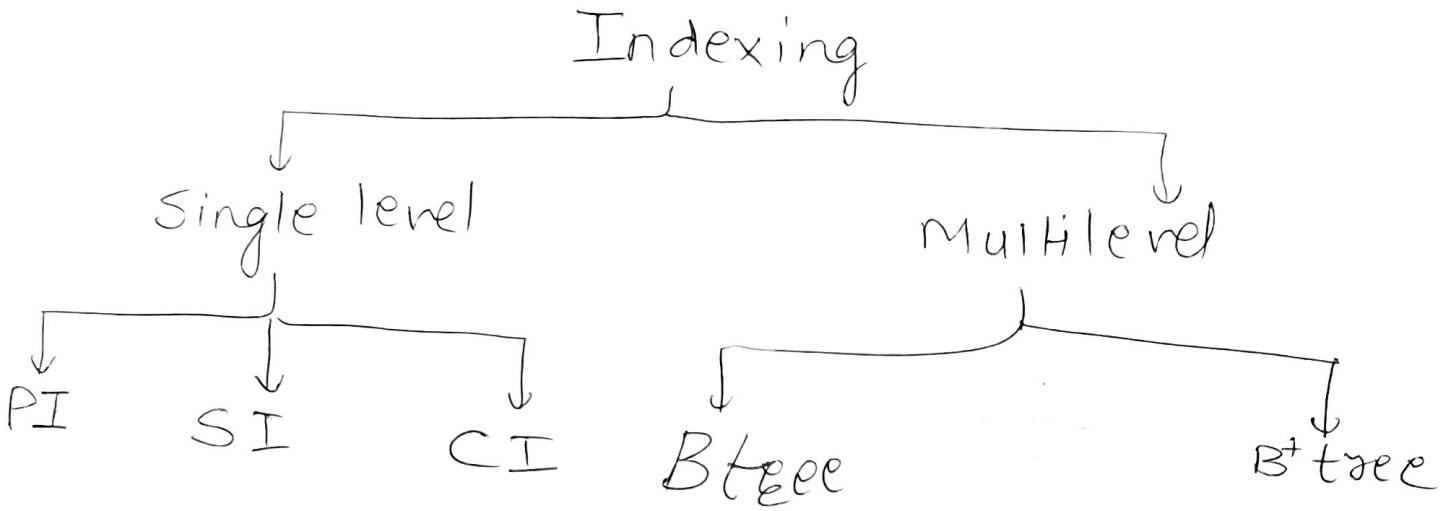


→ $\frac{BS}{10B} = \frac{1000}{10}$

$= \underline{100}$

∴ $\frac{1000}{100} = \underline{10}$

Total no. of blocks



Primary Indexing

- single level Indexing
- S.key = Prime attr.
- Ordered file
- Sparse Indexing

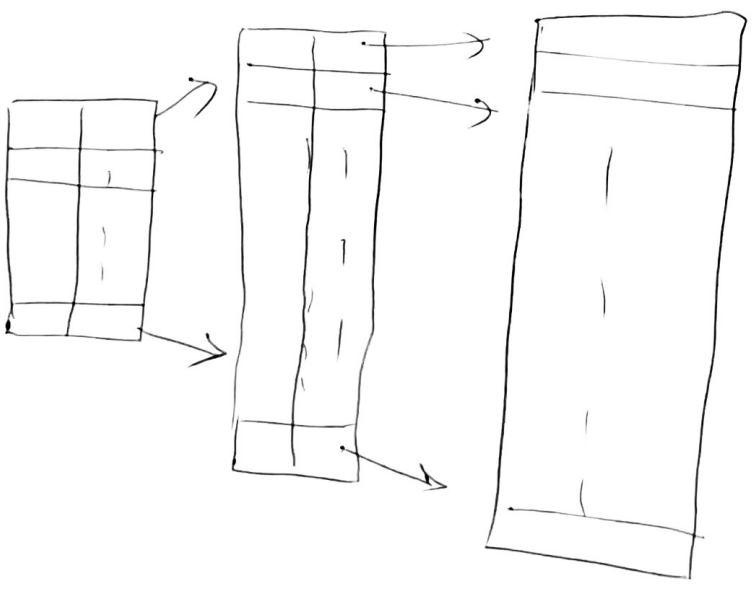
Clustered Indexing

- S.key = Non key attr.
- ordered file
- Sparse or dense Indexing

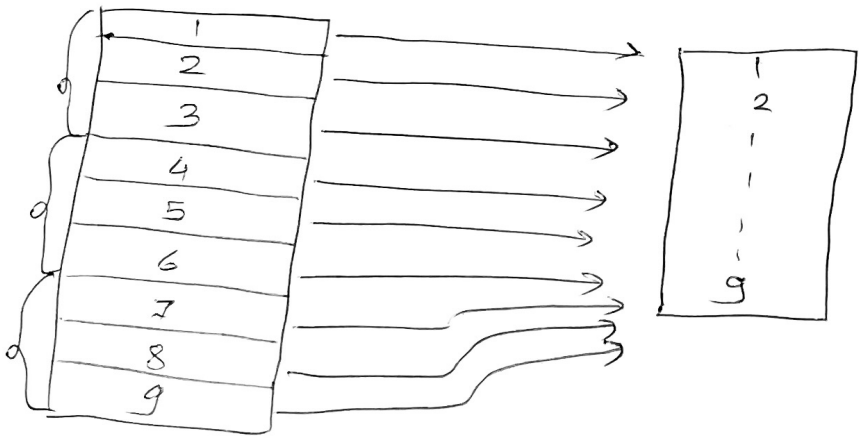
Secondary Indexing

- S.key = key or non key
- unordered
- Dense Indexing.

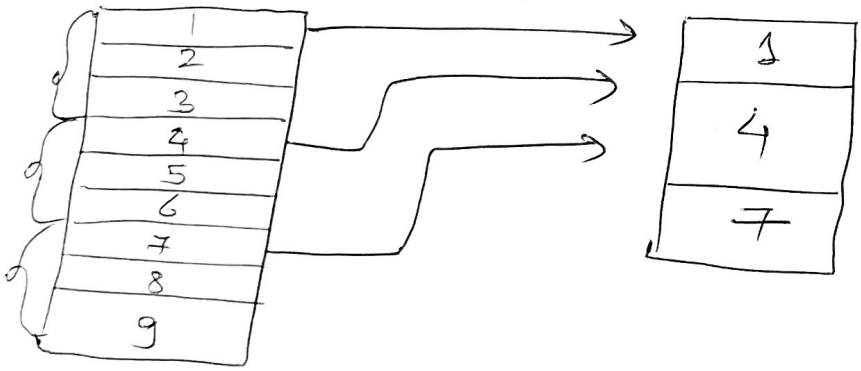
Multilevel Indexing



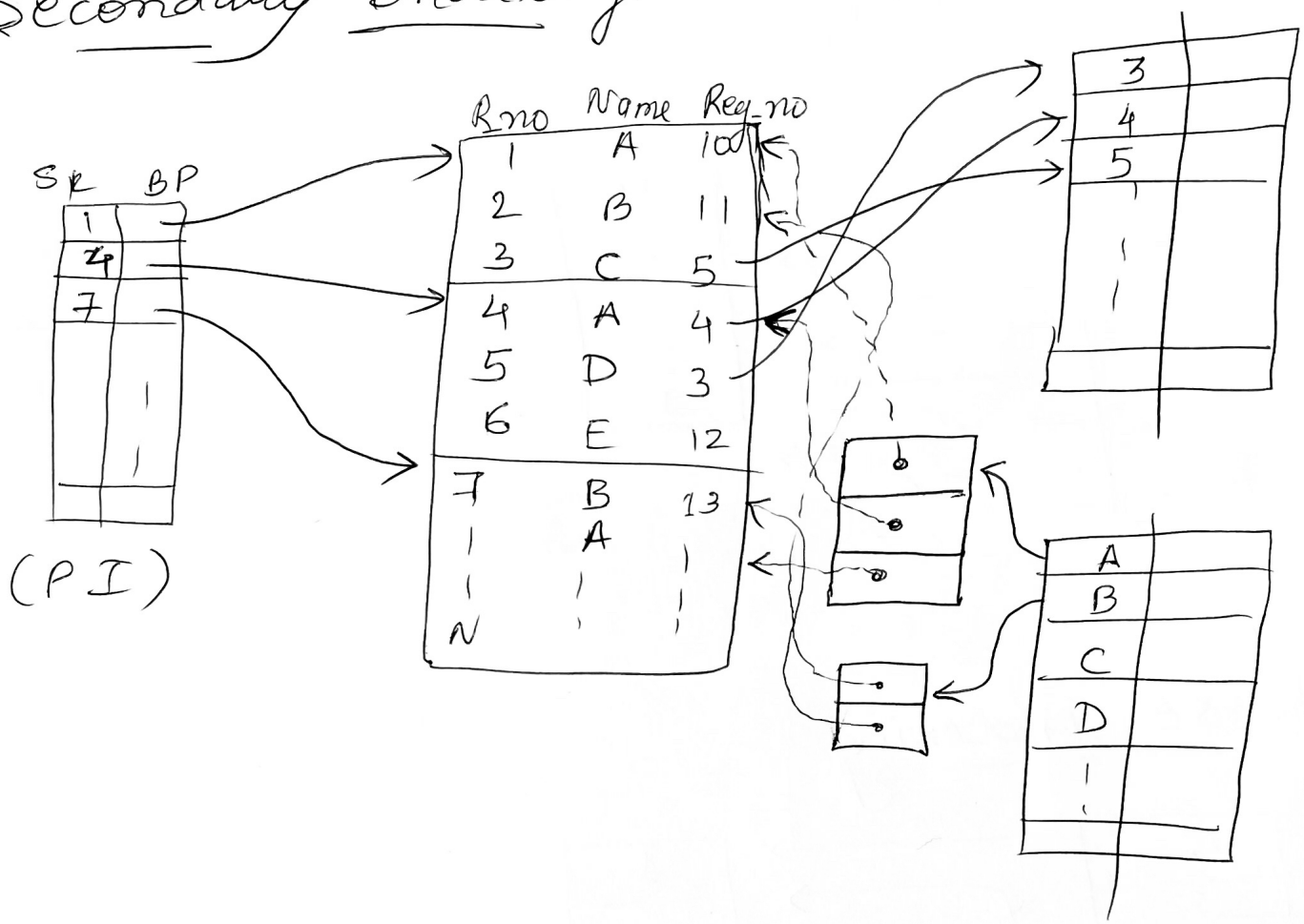
- Dense Indexing.



- Sparse Indexing

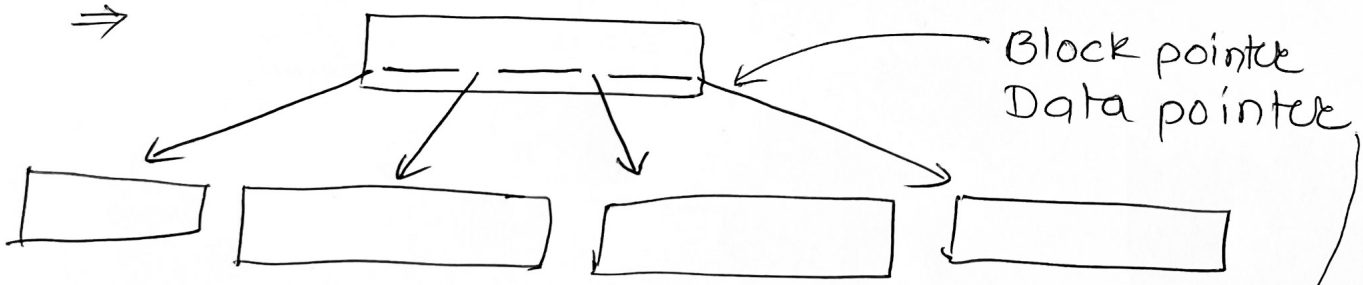


Secondary Indexing.

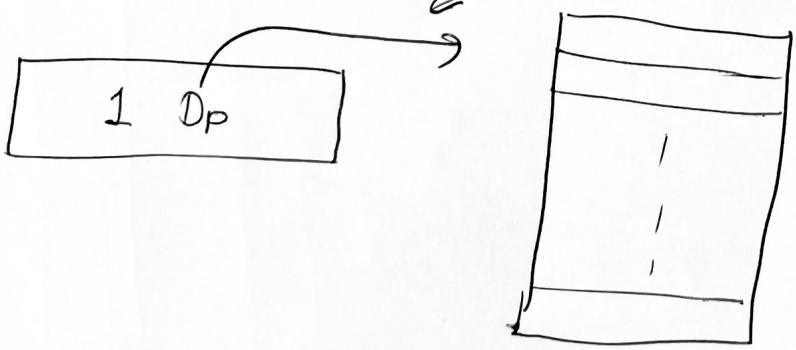


→ B tree

⇒



Its a Balanced Tree



⇒ Root → max child → n

→ min child → 2

n = order of Btree

$n-1$ = no. of keys (max)
= Dp

⇒ Intermidiate node → max child → n

→ min child → $\lceil n/2 \rceil$

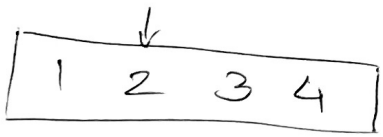
$\lceil \frac{n}{2} \rceil - 1$ = min no of keys

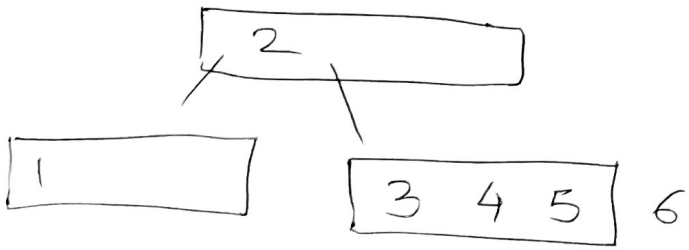
Ex:- 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

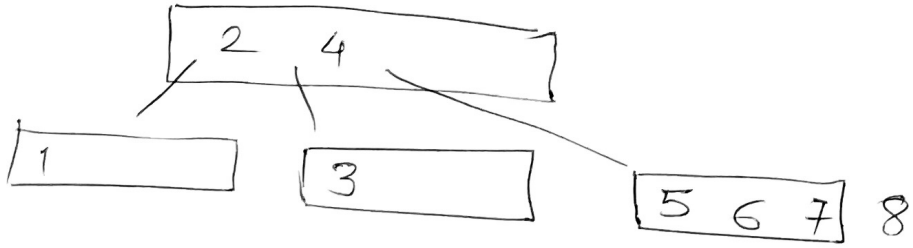
Order of B-tree = 4

∴ Max no of keys = $n-1 = 3$

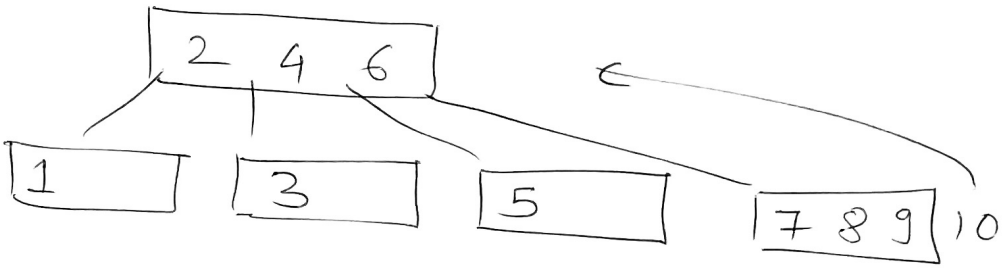
Min no of keys = $\lceil \frac{n}{2} \rceil - 1 = \lceil \frac{4}{2} \rceil - 1 = 1$

① 

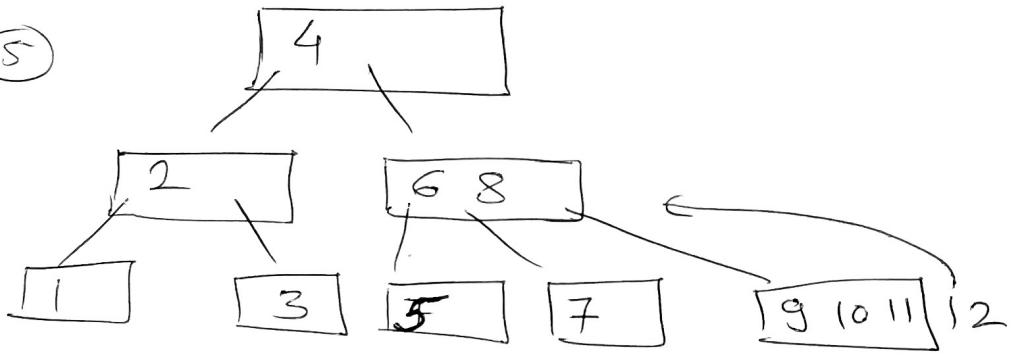
⇒② 

③ 

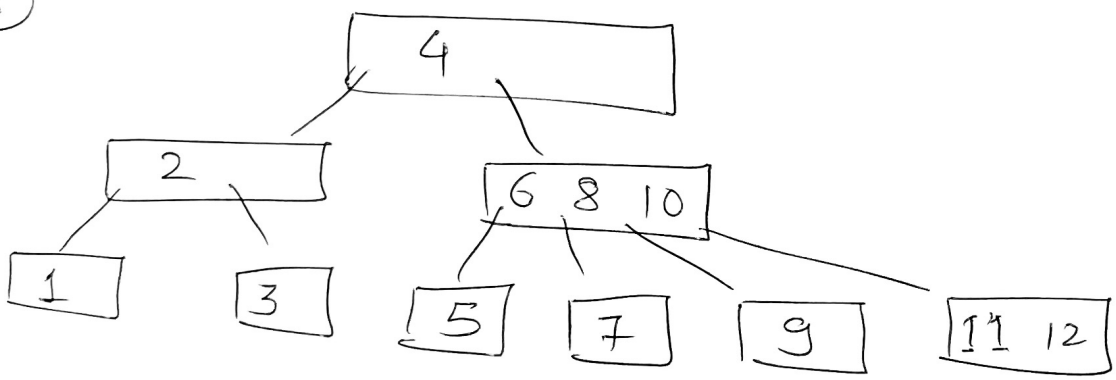
④



⑤



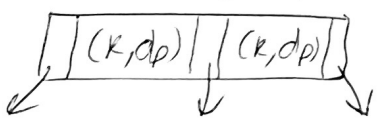
⑥



B^+ tree :-

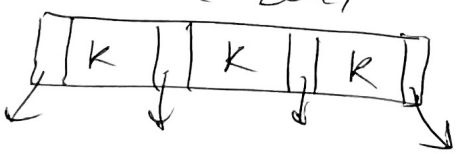
→ Data pointer only in leaf node

Non-Leaf



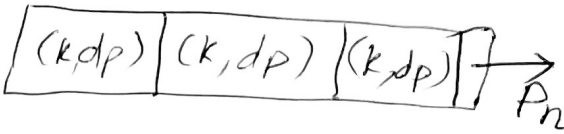
⟹

non-Leaf

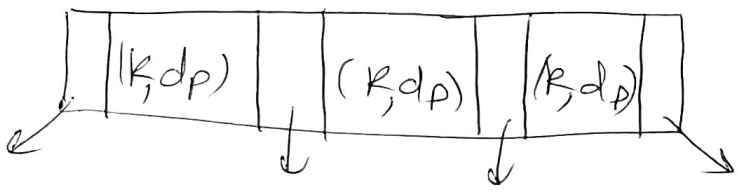


Leaf node

⟹ Leaf node



⇒ order of B-Tree



K size = 10B
BS = 512 B
dp = 8 B
Bp = 5 B

$= n \times Bp + (n-1)\,key + (n-1)\,dp \leq BS$

$= n \times 5 + (n-1)(10) + (n-1)(8) \leq 512$

$= 5n + 18n - 18 \leq 512$

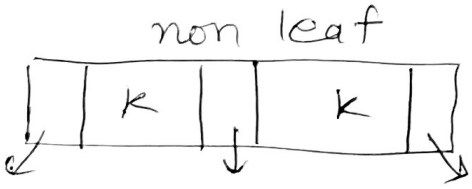
$= 23n \leq 512 + 18$

$23n \leq 530$

$n \leq \frac{530}{23} = 23.043$

∴ order of b tree = 23

⇒ order of B^+ Tree



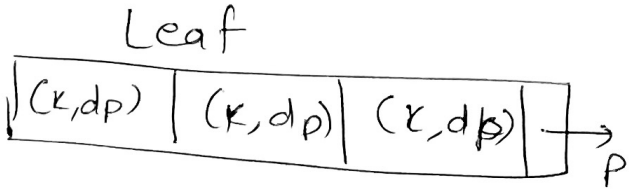
$n \times Bp + (n-1)K \leq BS$

$5n + 10n - 10 \leq 512$

$15n \leq 522$

$n \leq \frac{522}{15}$

≥ 34.8



$n(K+dp) + Bp \leq BS$

$n(10+8) + 5 \leq 512$

$18n \leq 507$

$n \leq 28.16$

∴ 28

Transaction

↳ Set of Logically related operation

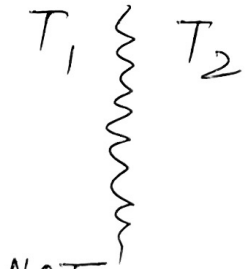
↳ Read()
↳ write()

1) Enter Amount
2) Enter Pin
3) SEND
4) X − amount
5) Y receives
6) Y + amount

A — Atomicity
"All" or None

C — Consistency
Db Remain Consistent Before & after

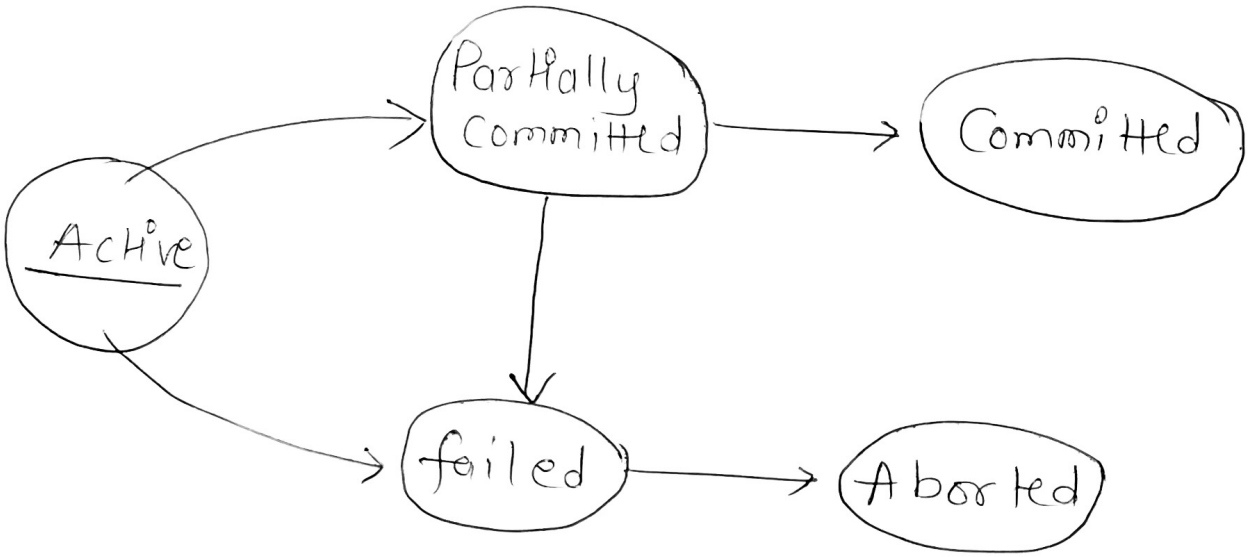
I — Isolation



NOT Interfere

D — Durability
Changes in Db after (T) completes should be durable (intact)

Transaction states



- **Active** : → First state
  → T's operations are performed

- **PC** : → After last/final operation
  → Not fully committed as it can go for 'aborted' due to some error.

- **failed** : → Due to some H/w, S/w, operation it enters in failed state
  → Can come here from Active or PC state.
  → Roll back (T).

- **Aborted** : → (T) has been rolled back
  → Db restored to original state.

- **Committed** :- Enter here after successfull completion of (T). & consistent update is made.

- **Schedule** : Execution sequence of multiple Transactions.

  - Serial ($T_1 \rightarrow T_2 \rightarrow T_3$) (waiting)
  - Parallel (T_1 ⟶ , T_2 ⟶ , T_3 ⟶) (fast)

Serial →

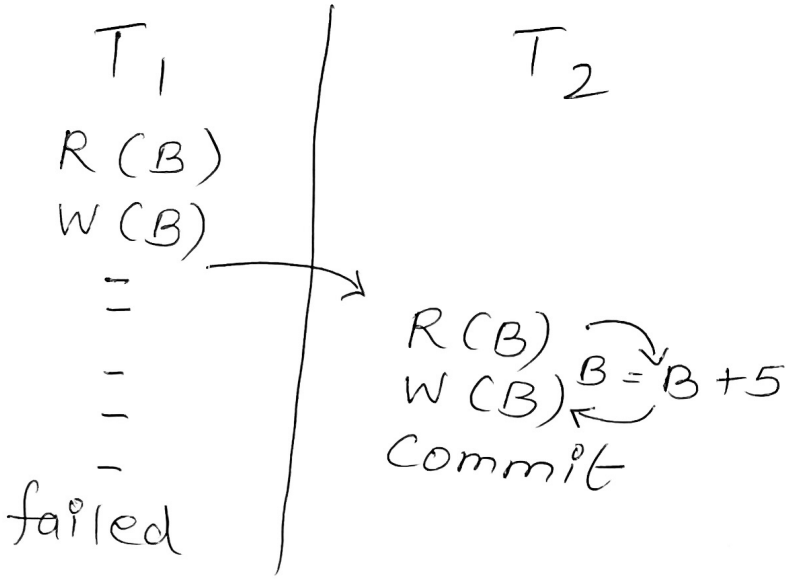
T_1	T_2
R(A)	
W(A)	
R(B)	
W(B)	
	R(B)
	W(B)
	R(A)
	W(A)

Parallel →

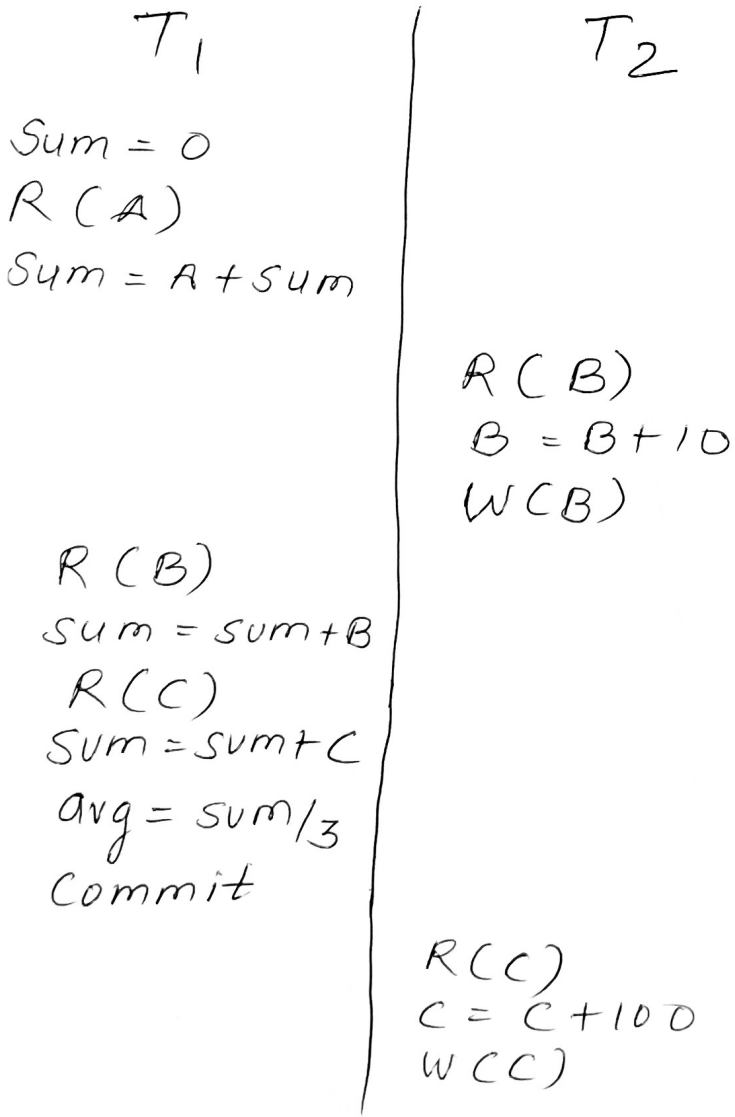
T_1	T_2
R(A)	
	R(A)
	W(A)
R(B)	
	R(B)
	W(B)

Concurrency Problems

1) Dirty Read



2) Incorrect Summary



- Lost update Problem

T_1	T_2
R (A)	
	R (A)
A = A + 5	
	A = A + 10
	W (A)
	Commit

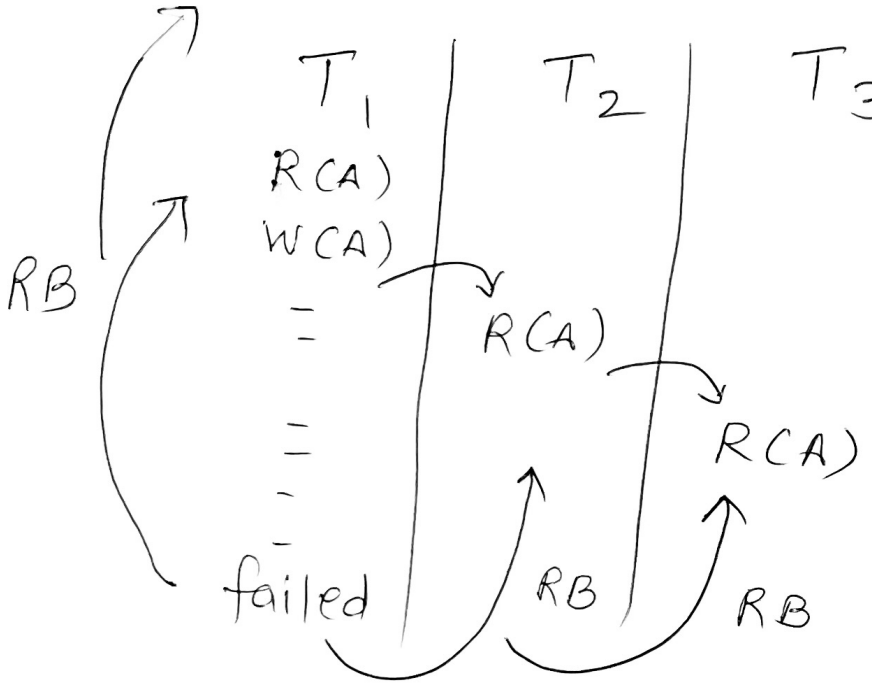
- Unrepeatable Read Problem

T_1	T_2
R (A)	
	R (A)
W (A)	
	R (A)

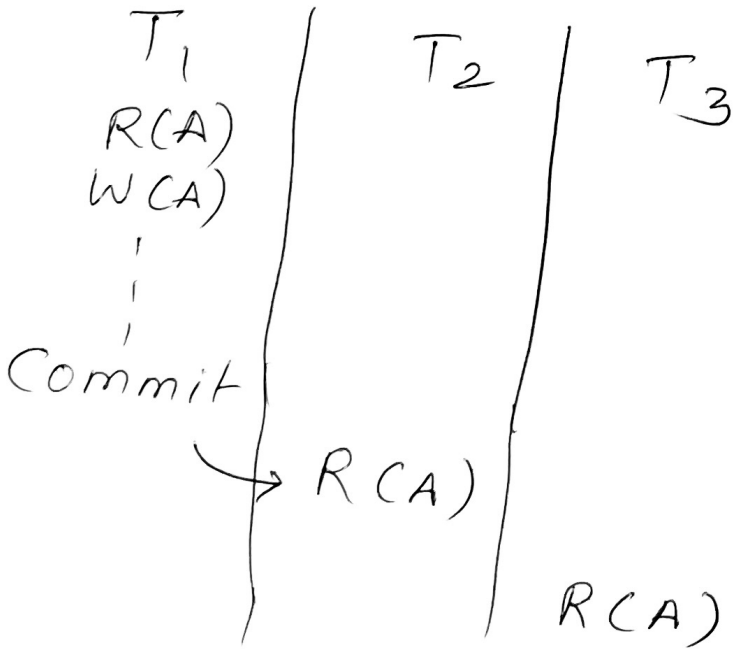
- Phantom Read Problem :

T_1	T_2
R (A)	
	R (A)
Delete (A)	
	R (A)

Cascading Schedule



Cascadeless Schedule



Recoverable Vs Non-Recoverable

T_1	T_2
R(A)	
W(A)	
	R(A)
Commit	
	commit

T_1	T_2
R(A)	
W(A)	
	R(A)
→ failed ←	
Commit	
	commit

↓

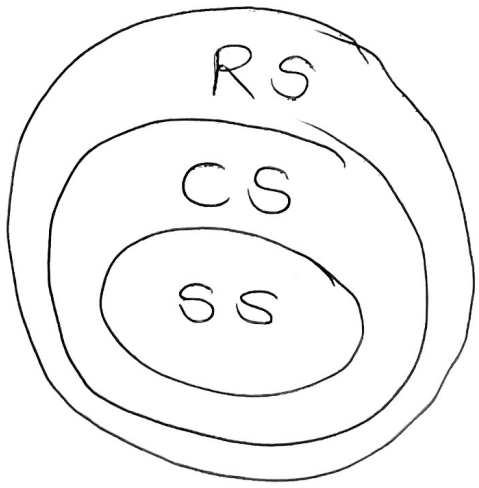
It can't be RB ←

Strict Schedule

T_1
R(A)
W(A) — Don't perform until Commit

R(A)
W(A)

T_1	T_2
R(A)	
W(A)	
	R(B)
	W(B)
Commit	
	R(A)
	commit



- Serializability
  - Conflict
  - view

Given: Parallel schedule → Convert to its equivalent → Serial copy

- Conflict. serializability

Conflicts (x)

T_1	T_2	
R(A)		
	R(B)	①
R(A)		
	W(B)	②
R(A)		
	R(A)	③

Conflicts (v)

T_1	T_2	
R(A)		
	W(A)	①
W(A)		
	W(A)	②
W(A)		
	R(A)	③

- Conflict equivalent

↳ $S_1 \rightarrow S_2$ (swap Non Conflicting Operations)

S_1

T_1	T_2
R(A)	
	R(B)
R(B)	
	R(A)

S_2

T_1	T_2
	R(B)
R(A)	
R(B)	
	R(A)

S_1

T_1	T_2
R(A)	
W(A)	
	R(A)
	W(A)
R(B)	
R(B)	
	R(B)
	W(B)

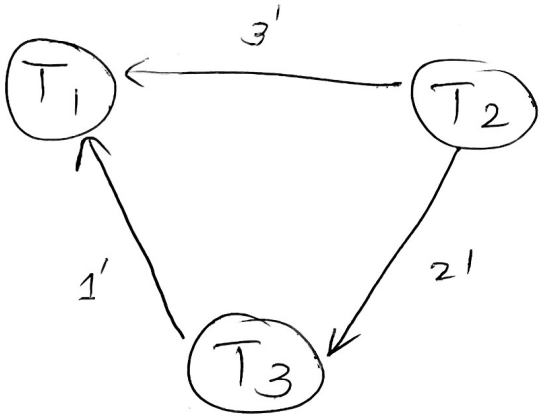
S_2

T_1	T_2
R(A)	
W(A)	
R(B)	
W(B)	
	R(A)
	W(A)
	R(B)
	W(A)

Precedence Graph

Conflict pairs
(RW, WR, WW)

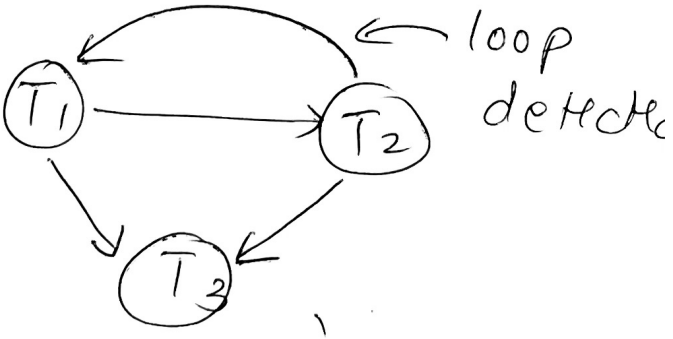
T_1	T_2	T_3
		R(A)
		R(B)
	R(A)	
	R(C)	
		W(A)
	W(C)	
R(C)		
W(B)		
W(C)		



If loop/cycle exist then Not `CSér. also it has a serial schedule (serializable)

Eg ⇒

T_1	T_2	T_3
R(x)		
	W(x)	
W(x)		
		W(x)



Its Not `CSer'

Now we use view serializability.

T_1	T_2	T_3
R(x)		
W(x)		
	W(x)	
		W(x)

- In both T_1 is Reading (x)
- In both T_3 is final write (x)

- **Concurrency Control**
  - maintain Consistency
  - \+
  - Concurrency (Many Transactions)

① Locking method
- shared (Read only)
- Exclusive (Read & write both)

T
Shared(x)
R(x)
Unlock(x)

	S	X
S	Y	N
X	N	N

T
Exclusive(x)
R(x)
W(x)
Unlock(x)

2PL (2 phase Locking)

Grow → Shrink

$T \rightarrow S(B)\ X(A)\ R(A)\ W(B)\ R(B)\ U(A)\ U(B)$

Grow (lock) | Shrink (unlock)

	T_1	T_2
G	X(P)	
	R(P)	
	W(P)	
		R(P)] wait till the shrink phase of T_1 starts & releases the lock on 'P'
	X(Q)	
	R(Q)	
	W(Q)	
S	U(P)	
	U(Q)	

(Always serializable).

- Time Stamp Ordering

$\overset{5}{T_1}$	$\overset{10}{T_2}$
R(A)	
	R(A)
	W(A)
W(A)	

RTS = 10
WTS = 5

- T_i request Read (A)

→ If $TS(T_i) < WTS(A)$ (Roll Back)

5 T_1	10 T_2
	W(A)
R(A)	

$5 < 10$

→ If $TS(T_i) \geq WTS(A)$ (Allowed)

5 T_1	10 T_2
W(A)	
	R(A)

set & $RTS_{(A)} = max(RTS(A), TS(T_i))$

- T_i request Write (A)

Not Allowed

$TS(T_i) < RTS(A)$

$TS(T_i) < WTS(A)$

(Roll back)

Allowed.

$TS(T_i) \geq RTS(A)$

$TS(T_i) \geq WTS(A)$

set

$WTS_{(A)} = max(WTS(A), TS($