Relational Algebra Operations

Fundamental operations:

- ☆ Select (σ)
- ☆ Project (Π)
- ☆ Union (∪)
- ☆ Set Difference (-)
- ☆ Cartesian Product (X)
- ☆ Rename (ρ)

Relational Algebra Operations

Additional operations:

- ☆ Set intersection ($\cap$)
- ☆ Assignment operation ($=$)
- ☆ Natural join ($\bowtie_*$)
- ☆ Division operation ($\div$)
- ☆ Outer join
  - Left outer join (⟕)
  - Right outer join (⟖)
  - Full outer join (⟗)

1. Select (σ)

- ☆ Select tuples that satisfy a given predicate.
- ☆ It is denoted by lowercase Greek letter sigma (σ).
- ☆ Syntax: $\sigma_{\text{<selection\_condition>}}$ (Relation).
- ☆ Example: $\sigma_{D\_ID = 2}$ (Employee).
- ☆ Comparison operators: $=$, $\neq$, $<$, $\leq$, $>$, and $\geq$.
- ☆ Connectives: AND ($\wedge$), OR ($\vee$) and NOT ($\neg$).

1. Select (σ)

Example 1: Write an RA expression to find all the instructors working in "Finance" department.

Solution:

$$\sigma_{Dept\_Name\ =\ \text{"Finance"}}\ (INSTRUCTOR)$$

Output:

ID	Name	Dept_Name	Salary
26589	Yusuf	Finance	95000
12547	Neil	Finance	80000

INSTRUCTOR

ID	Name	Dept_Name	Salary
10101	John	Biology	65000
12121	Robin	Computer Science	90000
25252	Alya	Electrical	40000
26589	Yusuf	Finance	95000
54789	Ravi	Music	60000
78787	Raj	Physics	87000
87458	Jayant	History	75000
76985	Pratik	Computer Science	89000
12547	Neil	Finance	80000

1. Select (σ)

Example 3: Find all instructors who are working in "Finance" department and drawing the salary greater than $87,000.

Solution:

$\sigma_{\text{Dept\_Name} = \text{"Finance"} \land \text{Salary} > 87000}$ (INSTRUCTOR)

Output:

ID	Name	Dept_Name	Salary
26589	Yusuf	Finance	95000

INSTRUCTOR

ID	Name	Dept_Name	Salary
10101	John	Biology	65000
12121	Robin	Computer Science	90000
25252	Alya	Electrical	40000
26589	Yusuf	Finance	95000
54789	Ravi	Music	60000
78787	Raj	Physics	87000
87458	Jayant	History	75000
76985	Pratik	Computer Science	89000
12547	Neil	Finance	80000

2. Project (Π)

- ☆ It returns its argument relation with certain attributes left out.
- ☆ It is a unary operator.
- ☆ It is denoted by the uppercase Greek letter pi (Π).
- ☆ Basically a relation is a set.
- ☆ In the result, the duplicate rows are eliminated.
- ☆ Syntax: $\Pi_{\text{Attribute1, Attribute2, ...}}(\text{Relation})$.

2. Project (Π)

Example 1: List all instructors' ID, name, and salary, but do not care about the dept_name.

Solution:

$\Pi_{\text{ID, Name, Salary}}$ (INSTRUCTOR)

(

INSTRUCTOR

ID	Name	Dept_Name	Salary
10101	John	Biology	65000
12121	Robin	Computer Science	90000
25252	Alya	Electrical	40000
26589	Yusuf	Finance	95000
54789	Ravi	Music	60000
78787	Raj	Physics	87000
87458	Jayant	History	75000
76985	Pratik	Computer Science	89000
12547	Neil	Finance	80000

2. Project (Π)

Example 2: Find the name of all instructors in the Computer Science department.

Solution:

$$\Pi_{Name}(\sigma_{Dept\_Name = \text{"Computer Science"}}(INSTRUCTOR))$$

Output:

Name
Robin
Pratik

INSTRUCTOR

ID	Name	Dept_Name	Salary
10101	John	Biology	65000
12121	Robin	Computer Science	90000
25252	Alya	Electrical	40000
26589	Yusuf	Finance	95000
54789	Ravi	Music	60000
78787	Raj	Physics	87000
87458	Jayant	History	75000
76985	Pratik	Computer Science	89000
12547	Neil	Finance	80000

3. Union (U)

- ☆ Any relation is a set.
- ☆ Similar to union operation in Set Theory.
- ☆ It is a binary operator.
- ☆ It is a set of all objects that are a member of A, or B, or both.
- ☆ Like project, the duplicate rows are eliminated.
- ☆ It is denoted by U.
- ☆ Syntax: Π_{Column} (Relation_1) U Π_{Column} (Relation_2)

3. Union (∪)

R	Alphabets
	A
	B
	C
	E
	F

S	Characters
	A
	B
	$
	E
	F
	1
	#

R ∪ S

R ∪ S
A
B
C
E
F
$
1
#

3. Union (U)

Example 1: List all customer names associated with the bank either as an account holder or a loan borrower.

Solution: $\Pi_{Cu\_Name}(Depositor) \cup \Pi_{Cu\_Name}(Borrower)$

DEPOSITOR

R

Cu_Name	Acc_No
Tom	A-101
Amy	A-502
Rose	A-304
John	A-689

BORROWER

S

Cu_Name	Loan_No
John	L-201
Smith	L-658
Rose	L-254
Jack	L-547

Cu_Name
Tom
Amy
Rose
John
Smith
Jack

3. Union (U)

Example 2: Find the set of all courses taught in the Fall 2009 semester, the Spring 2010 semester, or both.

Solution:

To find the set of all courses taught in the Fall 2009 semester, we write:

$$\Pi_{Course\_id}(\sigma_{Semester = \text{"Fall"} \land year = 2009}(SECTION))$$

To find the set of all courses taught in the Spring 2010 semester, we write:

$$\Pi_{Course\_id}(\sigma_{Semester = \text{"Spring"} \land year = 2010}(SECTION))$$

To answer the query, we need the union of these two sets:

$$\Pi_{Course\_id}(\sigma_{Semester = \text{"Fall"} \land year = 2009}(SECTION)) \cup \Pi_{Course\_id}(\sigma_{Semester = \text{"Spring"} \land year = 2010}(SECTION))$$

Two important conditions

For $R \cup S$ to be valid,

1. R and S must be of same arity.
2. For all i,

   Domain of the i^{th} attribute of R = Domain of i^{th} attribute of S.

4. Set Difference (−)

- ☆ It is like the same set difference in Set Theory.
- ☆ It is a binary operation.
- ☆ To find tuples that are in one relation but are not in another.
- ☆ R - S = Tuples in R but not in S.
- ☆ It is denoted by minus (-) symbol.

4. Set Difference (−)

R

Alphabets
A
B
C
D
E
F

S

Characters
A
B
$
L
Y
1
#

R - S

R - S
C
D
E
F

4. Set Difference (−)

Example 1: List all customer names those who have a deposit account but not availed loan.

DEPOSITOR ← R

Cu_Name	Acc_No
Tom	A-101
Amy	A-502
Rose	A-304
John	A-689

BORROWER ← S

Cu_Name	Loan_No
John	L-201
Smith	L-658
Rose	L-254
Jack	L-547

R - S

Cu_Name
Tom
Amy

Solution: $\Pi_{Cu\_Name}(Depositor) - \Pi_{Cu\_Name}(Borrower)$

4. Set Difference (–)

Example 2: Find all the courses taught in the Fall 2009 semester but not in Spring 2010.

Solution:

To find the set of all courses taught in the Fall 2009 semester, we write:

$$\Pi_{Course\_id}(\sigma_{Semester = \text{"Fall"} \wedge year = 2009}(SECTION))$$

To find the set of all courses taught in the Spring 2010 semester, we write:

$$\Pi_{Course\_id}(\sigma_{Semester = \text{"Spring"} \wedge year = 2010}(SECTION))$$

To answer the query, we need the minus of these two sets:

$$\Pi_{Course\_id}(\sigma_{Semester = \text{"Fall"} \wedge year = 2009}(SECTION)) - \Pi_{Course\_id}(\sigma_{Semester = \text{"Spring"} \wedge year = 2010}(SECTION))$$

Two important conditions

For R - S to be valid,

1. R and S must be of same arity.
2. For all i,

   Domain of the i^{th} attribute of R = Domain of i^{th} attribute of S.

5. Cartesian Product (X)

- ☆ Cartesian product associates every tuple of R_1 with every tuple of R_2.
- ☆ It is a binary operation.
- ☆ It is denoted by cross (X) symbol.
- ☆ R_1 X R_2 = All possible pairing.
- ☆ Same attribute may appear in R_1 and R_2.
- ☆ R = Depositor X Borrower.

5. Cartesian Product (X)

Example: Perform Depositor X Borrower.

DEPOSITOR

Cu_Name	Acc_No
Tom	A-101
Rose	A-304

BORROWER

Cu_Name	Loan_No
John	L-201
Smith	L-658
Rose	L-254
Jack	L-547

DEPOSITOR X BORROWER

Cu_Name	Acc_No	Cu_Name	Loan_No
Tom	A-101	John	L-201
Tom	A-101	Smith	L-658
Tom	A-101	Rose	L-254
Tom	A-101	Jack	L-547
Rose	A-304	John	L-201
Rose	A-304	Smith	L-658
Rose	A-304	Rose	L-254
Rose	A-304	Jack	L-547

5. Cartesian Product (X)

- ☆ R = Depositor X Borrower.
- ☆ R = (Depositor.Cu_Name, Depositor.Acc_No, Borrower.Cu_Name, Borrower.Loan_No).
- ☆ R = (Depositor.Cu_Name, Acc_No, Borrower.Cu_Name, Loan_No).

5. Cartesian Product (X)

Example : Find the names of all instructors in the Physics department together with the course id of all courses they taught.

Solution:

$\sigma_{Dept\_Name = \text{"Physics"}}$ (INSTRUCTOR X TEACHES)

$\sigma_{Instructor.ID = Teaches.ID}$ ($\sigma_{Dept\_Name = \text{"Physics"}}$ (INSTRUCTOR X TEACHES))

$\Pi_{name, course\_id}$ ($\sigma_{Instructor.ID = Teaches.ID}$ ($\sigma_{Dept\_Name = \text{"Physics"}}$ (INSTRUCTOR X TEACHES)))

[or]

$\Pi_{name, course\_id}$ ($\sigma_{Instructor.ID = Teaches.ID}$ (($\sigma_{Dept\_Name = \text{"Physics"}}$ (INSTRUCTOR) X TEACHES))

6. Rename (ρ)

- ☆ Relations in the database have names.
- ☆ The results of relational-algebra expressions do not have a name.
- ☆ It is useful to be able to give them names.
- ☆ It is a unary operation.
- ☆ It is denoted by the lowercase Greek letter rho (ρ).
- ☆ Syntax: $\rho_x (E)$

6. Rename (ρ)

- ☆ A relation *r* by itself is considered a trivial relational-algebra expression.
- ☆ Thus, the same rename operation can be applied to a relation *r* to get the same relation under a new name.
- ☆ The results of relational-algebra expressions do not have a name.
- ☆ $\rho_{x(A1, A2, \ldots An)}$ (Expression)

6. Rename (ρ)

Example : Find the names of all instructors in the Physics department together with the course id of all courses they taught and name the resultant relation as Ins_Phy.

Solution:

$\sigma_{Dept\_Name\ =\ \text{"Physics"}}$ (INSTRUCTOR X TEACHES)

$\sigma_{Instructor.ID\ =\ Teaches.ID}$ ($\sigma_{Dept\_Name\ =\ \text{"Physics"}}$ (INSTRUCTOR X TEACHES))

$\Pi_{name,\ course\_id}$ ($\sigma_{Instructor.ID\ =\ Teaches.ID}$ ($\sigma_{Dept\_Name\ =\ \text{"Physics"}}$ (INSTRUCTOR X TEACHES)))

[or]

$\Pi_{name,\ course\_id}$ ($\sigma_{Instructor.ID\ =\ Teaches.ID}$ (($\sigma_{Dept\_Name\ =\ \text{"Physics"}}$ (INSTRUCTOR) X TEACHES))

ρ_{Ins_Phy} ($\Pi_{name,\ course\_id}$ ($\sigma_{Instructor.ID\ =\ Teaches.ID}$ (($\sigma_{Dept\_Name\ =\ \text{"Physics"}}$ (INSTRUCTOR) X TEACHES)))

Set Intersection (∩)

- ☆ Any relation is a set.
- ☆ Similar to intersection operation in Set Theory.
- ☆ It is a binary operator.
- ☆ It is a set of all objects that are a member of both A and B.
- ☆ It is denoted by ∩.
- ☆ Syntax: Π_{Column} (Relation_1) ∩ Π_{Column} (Relation_2)

Set Intersection (∩)

R

Alphabets
A
B
C
D
E
F

S

Characters
A
B
$
E
F
1
#

$R \cap S$

$R \cap S$
A
B
E
F

Set Intersection (∩)

Example 1: Find the names of all customers who have deposited money and also availed loan.

Solution: $\Pi_{Cu\_Name}(Depositor) \cap \Pi_{Cu\_Name}(Borrower)$

DEPOSITOR (R)

Cu_Name	Acc_No
Tom	A-101
Amy	A-502
Rose	A-304
John	A-689

BORROWER (S)

Cu_Name	Loan_No
John	L-201
Smith	L-658
Rose	L-254
Jack	L-547

R ∩ S

Cu_Name
Rose
John

Set Intersection (∩)

Example 2: Find the set of all courses taught in the Fall 2009 semester and the Spring 2010 semesters.

Solution:

To find the set of all courses taught in the Fall 2009 semester, we write:

$$\Pi_{Course\_id}(\sigma_{Semester = \text{"Fall"} \wedge year = 2009}(SECTION))$$

To find the set of all courses taught in the Spring 2010 semester, we write:

$$\Pi_{Course\_id}(\sigma_{Semester = \text{"Spring"} \wedge year = 2010}(SECTION))$$

To answer the query, we need the intersection of these two sets:

$$\Pi_{Course\_id}(\sigma_{Semester = \text{"Fall"} \wedge year = 2009}(SECTION)) \cap \Pi_{Course\_id}(\sigma_{Semester = \text{"Spring"} \wedge year = 2010}(SECTION))$$

☆ $R \cap S = R - (R - S)$.

Set Intersection (∩): $R \cap S = R - (R - S)$

☆ $R \cap S = R - (R - S)$.

R	Alphabets
	A
	B
	C
	D
	E
	F

S	Characters
	A
	B
	$
	E
	F
	1
	#

$R \cap S$
A
B
E
F

$R - S$
C
D

$R - (R - S)$
A
B
E
F

Assignment Operation (=)

- ☆ Similar to assignment operator in programming languages.
- ☆ Denoted by = or $\leftarrow$.
- ☆ It is a binary operator.
- ☆ The database might be modified if assignment to a permanent relation is made.
- ☆ Useful in the situation where it is required to write relational algebra expressions by using temporary relation variable.
- ☆ Example: $P = R \cap S$.
- ☆ Does not provide any additional power to relational algebra.

Assignment Operation (=)

☆ It provides a convenient way to express complex queries.

☆ Example:

$$\text{Temp1} = \text{Expression1}$$

$$\text{Temp2} = \text{Expression2}$$

$$\text{Result} = \text{Temp1} \div \text{Temp2}$$

[or]

$$\text{Temp1} \leftarrow R \times S$$

$$\text{Temp2} \leftarrow \Pi_{\text{Course\_id}}(\sigma_{\text{Semester} = \text{"Spring"} \wedge \text{year} = 2010}(\text{SECTION}))$$

$$\text{Result} \leftarrow \Pi_{R \cup S}(\text{Temp2})$$

Relational Algebra Operations – Binary

The JOIN Operation ($\bowtie$):

Syntax: $R \bowtie_{\text{<join condition>}} S$

DEPARTMENT	DName	DNo	Mgr_SSN
	Research	2	553621425
	Finance	5	996856974

EMPLOYEE	SSN	FName	LName	DNo
	123658974	Alex	Smith	2
	553621425	Fred	Scott	5
	996856974	Elsa	David	5
	859689742	Peter	Williams	2

$$\text{Dept\_Mgr} \leftarrow \text{DEPARTMENT} \bowtie_{\text{Mgr\_SSN} = \text{SSN}} \text{EMPLOYEE}$$

Relational Algebra Operations – Binary

The JOIN Operation ($\bowtie$):

DEPARTMENT	DName	DNo	Mgr_SSN
	Research	2	553621425
	Finance	5	996856974

EMPLOYEE	SSN	FName	LName	DNo
	123658974	Alex	Smith	2
	553621425	Fred	Scott	5
	996856974	Elsa	David	5
	859689742	Peter	Williams	2

$$\text{Temp} \leftarrow \text{DEPARTMENT} \times \text{EMPLOYEE}$$

$$\text{Dept\_Mgr} \leftarrow \sigma_{(\text{MgrSSN} = \text{SSN})} (\text{Temp})$$

❖ Relational Algebra Operations – Binary

★ The JOIN Operation (⋈):

DEPARTMENT

DName	DNo	Mgr_SSN
Research	2	553621425
Finance	5	996856974

EMPLOYEE

SSN	FName	LName	DNo
123658974	Alex	Smith	2
553621425	Fred	Scott	5
996856974	Elsa	David	5
859689742	Peter	Williams	2

Dept_Mgr

DName	DNo	Mgr_SSN	SSN	FName	LName	DNo
Research	2	553621425	553621425	Fred	Scott	2
Finance	5	996856974	996856974	Elsa	David	5

Result of the JOIN Operation

Relational Algebra Operations – Binary

The THETA Join (θ):

<join condition> : $A_i \; \theta \; B_j$

Syntax: $R \bowtie_{A_i \theta B_j} S$

$A_i \Rightarrow$ Attribute of R

$B_j \Rightarrow$ Attribute of S

$\theta \Rightarrow \{ =, <, <=, >, >=, \neq \}$

R	A_1	A_2
	20	25
	80	40

S	B_1
	50
	35

$R \bowtie_{A_2 = B_1} S$

$R \bowtie_{A_2 < B_1} S$

$R \bowtie_{A_2 <= B_1} S$

$R \bowtie_{A_2 > B_1} S$

$R \bowtie_{A_2 >= B_1} S$

$R \bowtie_{A_2 \neq B_1} S$

❖ Relational Algebra Operations – Binary

★ The THETA Join (θ):

$$R \bowtie_{A_2 > B_1} S$$

R	A1	A2
	20	25
	80	40

S	B1
	50
	35

A1	A2	B1
20	25	50
20	25	35
80	40	50
80	40	35

Result of "X"

A1	A2	B1
80	40	35

Result of "⋈"

Relational Algebra Operations – Binary

The EQUIJOIN Operation:

- The only comparison operator used is "=".

$$\text{Dept\_Mgr} \leftarrow \text{DEPARTMENT} \bowtie_{\text{Mgr\_SSN} = \text{SSN}} \text{EMPLOYEE}$$

The NATURAL JOIN Operation (*):

- Can be performed only if there is a common attribute in between the relations.

Relational Algebra Operations – Binary

The NATURAL JOIN Operation (*):

PROJECT

PID	PName	DNum
101	ProjectX	1
102	ProjectY	2
103	ProjectZ	2

DEPARTMENT

DNo	Mgr_SSN
1	553621425
2	996856974

$$\text{Proj\_Dept} \leftarrow \text{PROJECT} * \rho_{(\text{DNum},\ \text{Mgr\_SSN})}(\text{DEPARTMENT})$$

$$\text{DEPT} \leftarrow \rho_{(\text{DNum},\ \text{Mgr\_SSN})}(\text{DEPARTMENT})$$

$$\text{Proj\_Dept} \leftarrow \text{PROJECT} * \text{DEPT}$$

❖ Relational Algebra Operations – Binary

★ The NATURAL JOIN Operation (*):

PROJECT	PID	PName	DNum
	101	ProjectX	1
	102	ProjectY	2
	103	ProjectZ	2

DEPARTMENT	DNo	Mgr_SSN
	1	553621425
	2	996856974

Proj_Dept	PID	PName	DNum	Mgr_SSN
	101	ProjectX	1	553621425
	102	ProjectY	2	996856974
	103	ProjectZ	2	996856974

Result of " * "

❖ Relational Algebra Operations – Binary

★ The DIVISION Operation (÷):

Ex: To retrieve the Employee ID (EID) of the employees working on all projects.

EMPLOYEE

EID	PID
1001	1
1002	1
1002	2
1003	2

PROJECT

PID
1
2

Res

EID
1002

Result of the " ÷ " Operation

$$Res \leftarrow EMPLOYEE \div PROJECT$$

❖ Relational Algebra Operations – Binary

★ The DIVISION Operation (÷):

R: EMPLOYEE

EID (x)	PID (y)
1001	1
1002	1
1002	2
1003	2

S: PROJECT

PID
1
2

$(T1 \times S) - R$

EID (x)	PID
1001	2
1003	1

T1

EID
1001
1002
1003

T2

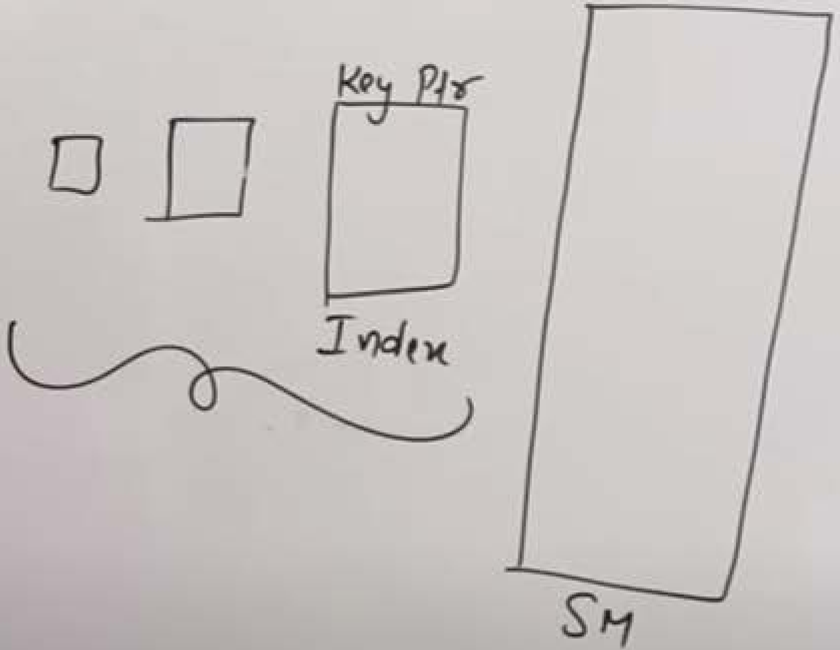
EID
1001
1003

$T1 \leftarrow \pi_x(R)$

$T2 \leftarrow \pi_x((T1 \times S) - R)$

$\boxed{Res \leftarrow T1 - T2}$

B-Tree (Dynamic Multilevel Index)



	Root	Intermidiate Node
Max	p	p
Min	2	$\lceil p/2 \rceil$

B-Tree (Dynamic Multilevel Index)

10, 20, 30 Block Pointer

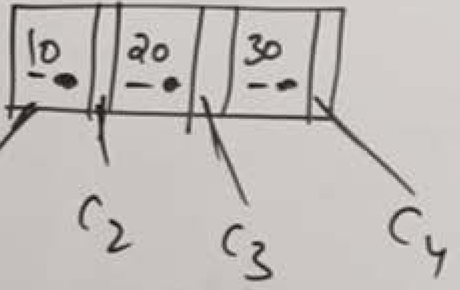
Tree Pointer

Children	Root	Intermidiate Node
Max	p	p
Min	2	$\lceil p/2 \rceil$

Keys

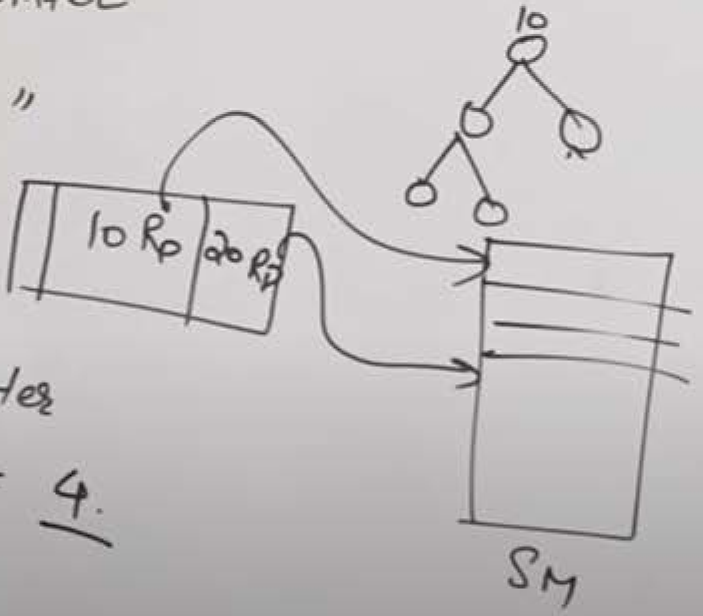
Data Pointer

Record "



$Keys = p-1$

$R_p = p-1$



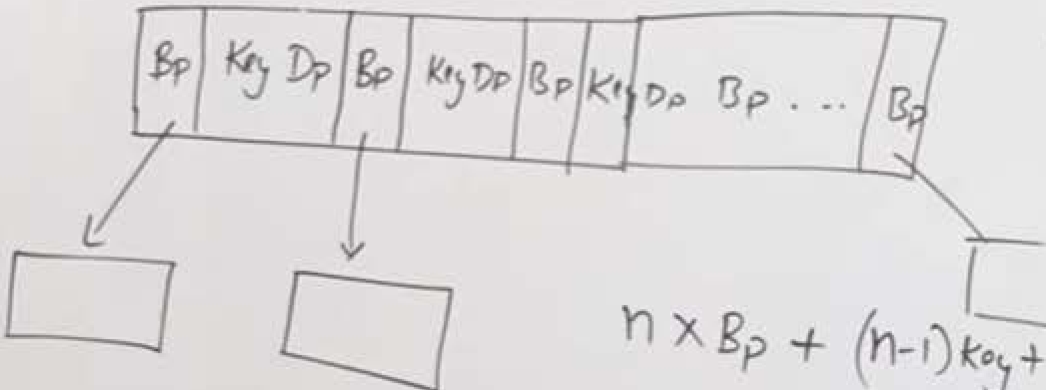
Keys = Record Pointer

Order = p = Max. no of children = 4.

$\downarrow$

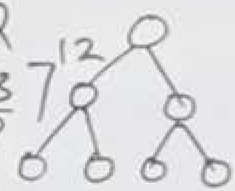
B_p

Consider a B-Tree with Key Size = 10 bytes, block size 512 bytes, data pointer is of size 8 bytes and block pointer is 5 bytes. Find the order of B-Tree?



23, 2

$23, \lceil \frac{23}{2} \rceil$ 12



Order - 1

$n \times B_p + (n-1) \text{key} + (n-1) R_p \leq \text{Block size}$

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

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

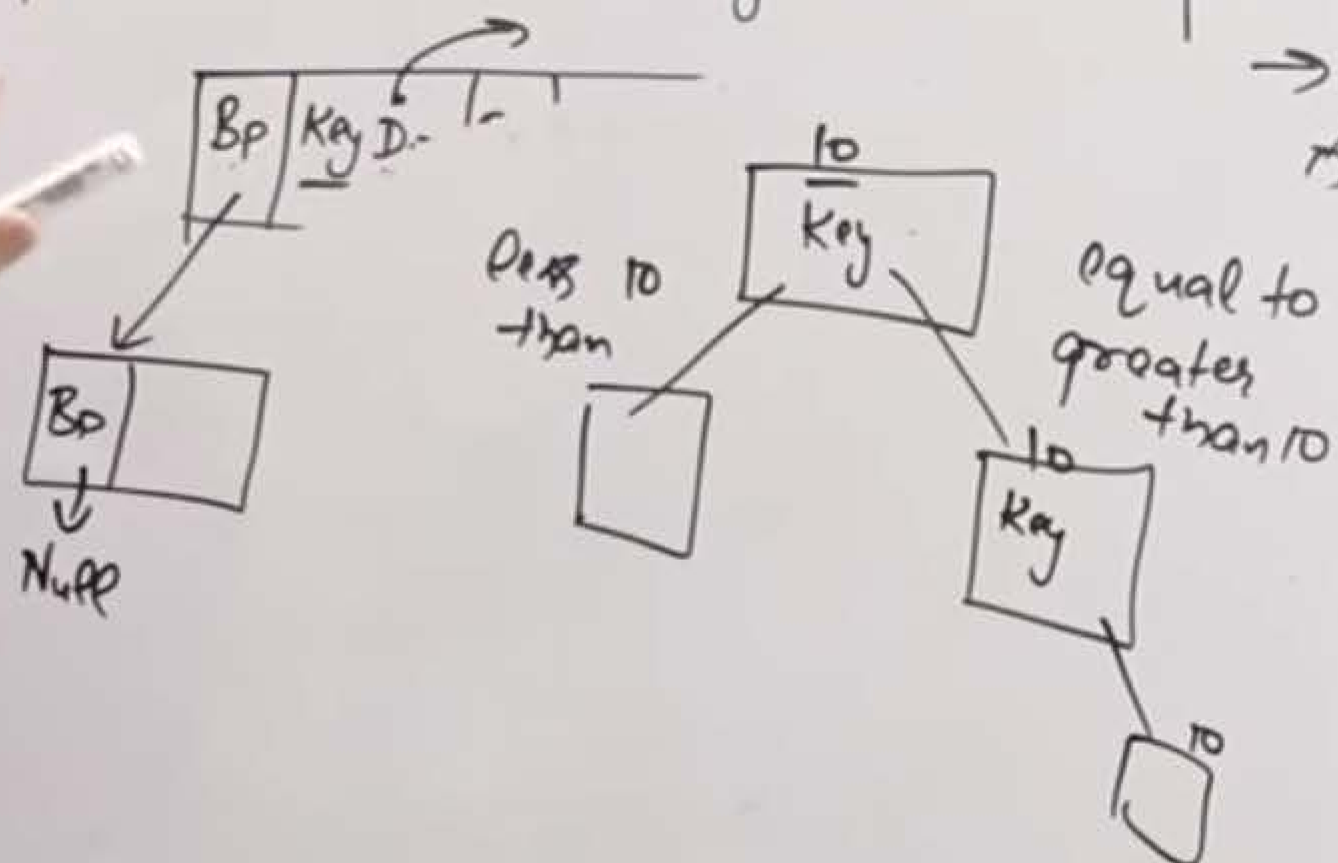
$23n - 18 \leq 512$

$23n \leq 530$

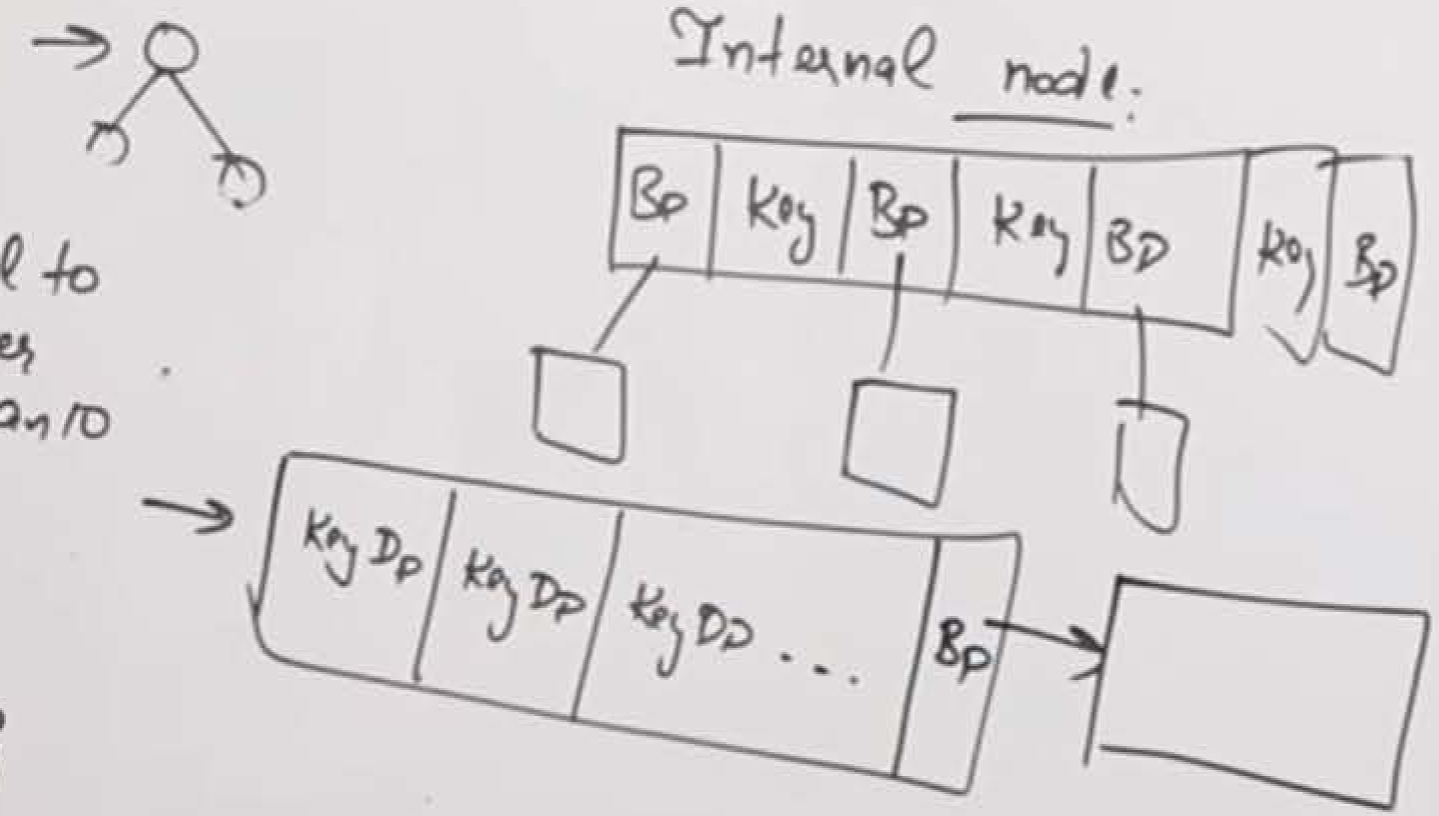
$n \leq \frac{530}{23} = 23.04$

23

B-Tree	B^+ Tree
1) Data is stored in leaf as well as internal nodes.	1) Data is stored only in leaf nodes.
2) Searching is slower, deletion complex	2) Searching is faster, deletion easy (directly from leaf node)
3) No Redundant search key Present	3) Redundant Keys may present.
4) Leaf nodes not linked together	4) Linked together like Linked list.

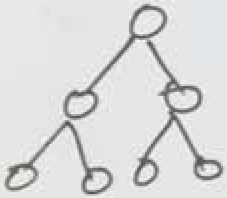


Internal node:



Consider a B^+ Tree with Key Size = 10 bytes, block size = 512 bytes data pointer = 8 bytes and block pointer = 5 bytes. What is the order of leaf and non leaf node?

Key Dp pair.



Non leaf.

Bp	Key	Bp	Key	Bp ...	Key ...	Bp

$$n \times Bp + (n-1) \times Key \leq \text{Block size}$$
$$n \times 5 + (n-1)\, 10 \leq 512$$
$$5n + 10n - 10 \leq 512$$
$$15n \leq 522$$
$$n \leq \frac{522}{15} = 34.8$$

34

Leaf:

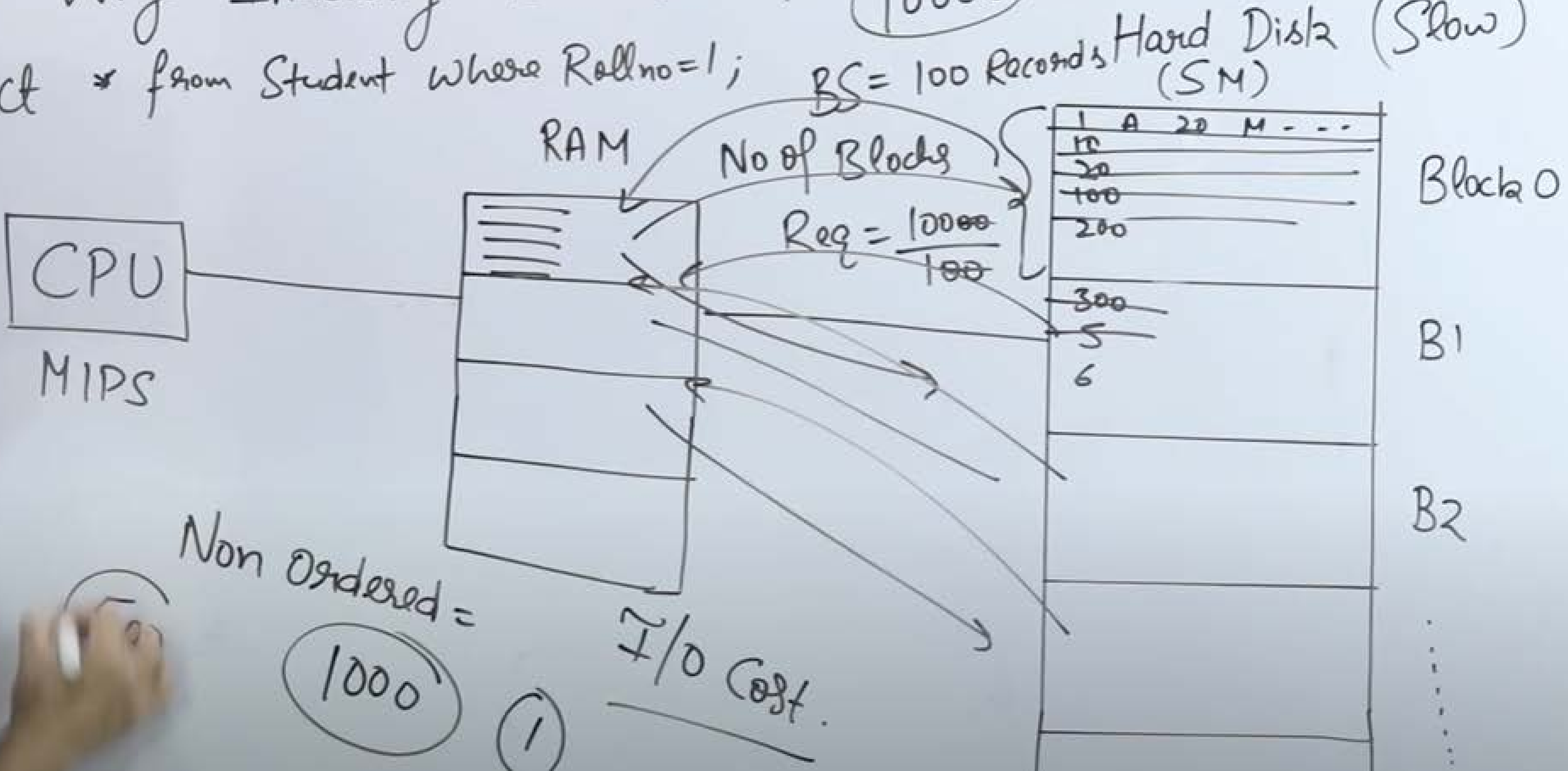
Key Dp, key Dp, key Dp,	Bp

$$x(Key + Dp) + Bp \leq \text{Block size}$$
$$x(10 + 8) + 5 \leq 512$$
$$18x \leq 507$$
$$x \leq \frac{507}{18} = 28.$$

28

Why Indexing is used? 10000 Records

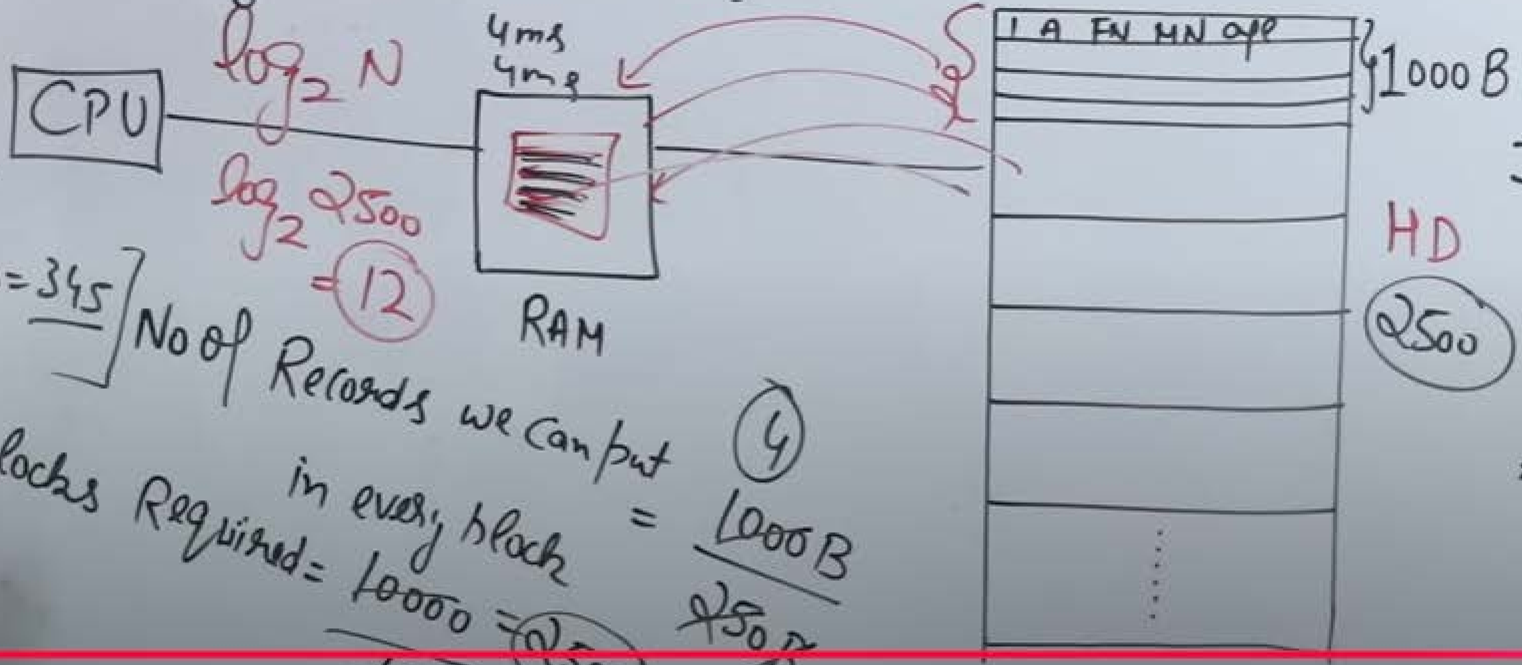
Select * from Student where Rollno=1;



Without Indexing.

Consider a Hard Disk in which Block Size = 1000 Bytes, each record is of Size = 250 Bytes. If total no of records are 10000 and the data entered in Hard disk without any order (Unordered) (Ordered) what is avg time Complexity to search a record from HD?

Select * from Student where Rollno = 345



No of Records we can put in every block $= \frac{1000B}{250B} = 4$

No of blocks Required $= \frac{10000}{4} = 2500$

$O(N)$

I/O Cost =

Best case = 1

Worst " = 2500 = N

Avg $= \frac{2500}{2} = \frac{N}{2} = 1250$

"Types of Indexes"

1) Primary
2) Clustered
Secondary

Ordered file	Primary Index	Clustered Index
Unordered file	Secondary Index	Secondary Index
	Key	Non Key

10
4
6
12
2
1

1
2
...

10
7
6

1
2
3
4
5
5
7
...

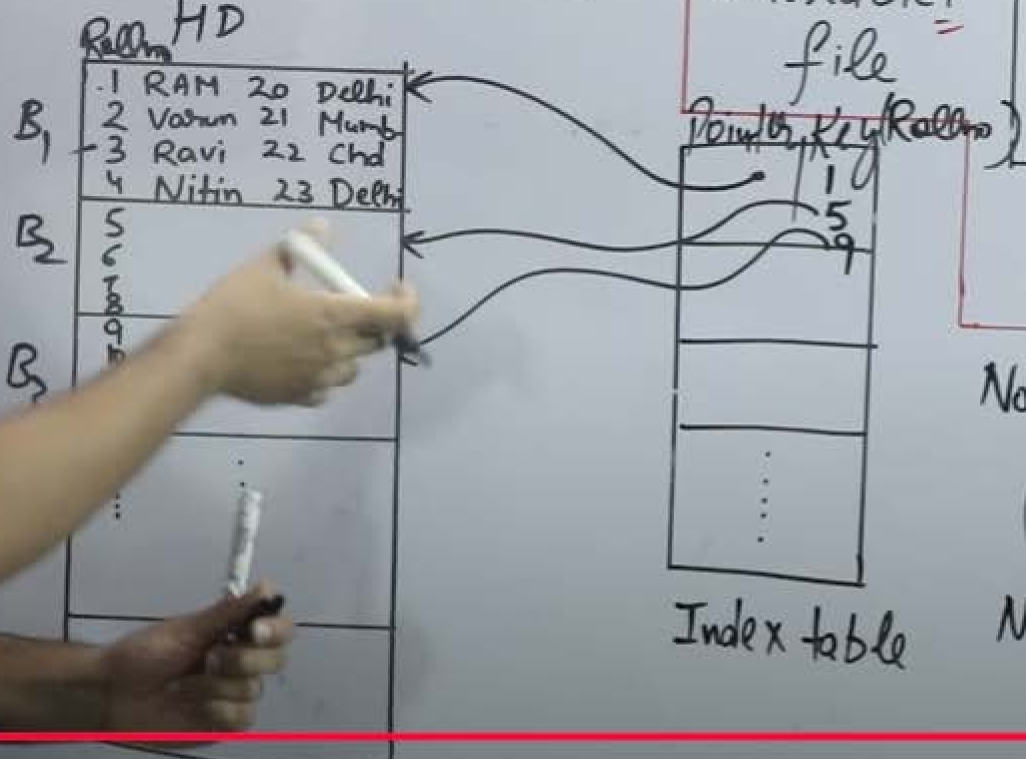
1
1
2
2
2
3
3
4
4
4
5

7
6
2
1
9
3

2
5
2
3
9
8
2
5
6
7

"Primary Index"

file Sparse



Index table

	Key	Non Key
Ordered file	Primary Index (Sparse)	Clustered Index
Unordered file	Secondary Index	Secondary Index

No of entries in IT = No of blocks in HD

$(\log_2 N+1)$

N = No of blocks in Index table

Disk Architecture

Platters → Surface → Track → Sectors → Data

$8 \times 2 \times 256 \times 512 \times 512\,KB$

$2^3 \times 2^1 \times 2^8 \times 2^9 \times 2^9 \times 2^{10}\,B$

Disk Size $= P \times S \times T \times S \times D$

$= 2^{40}\,B$

$= \boxed{1TB.}$

$1K = 2^{10}$

$1M = 2^{20}$

$1G = 2^{30}$

$1T = 2^{40}$

No. of bits required to represent Disk size.

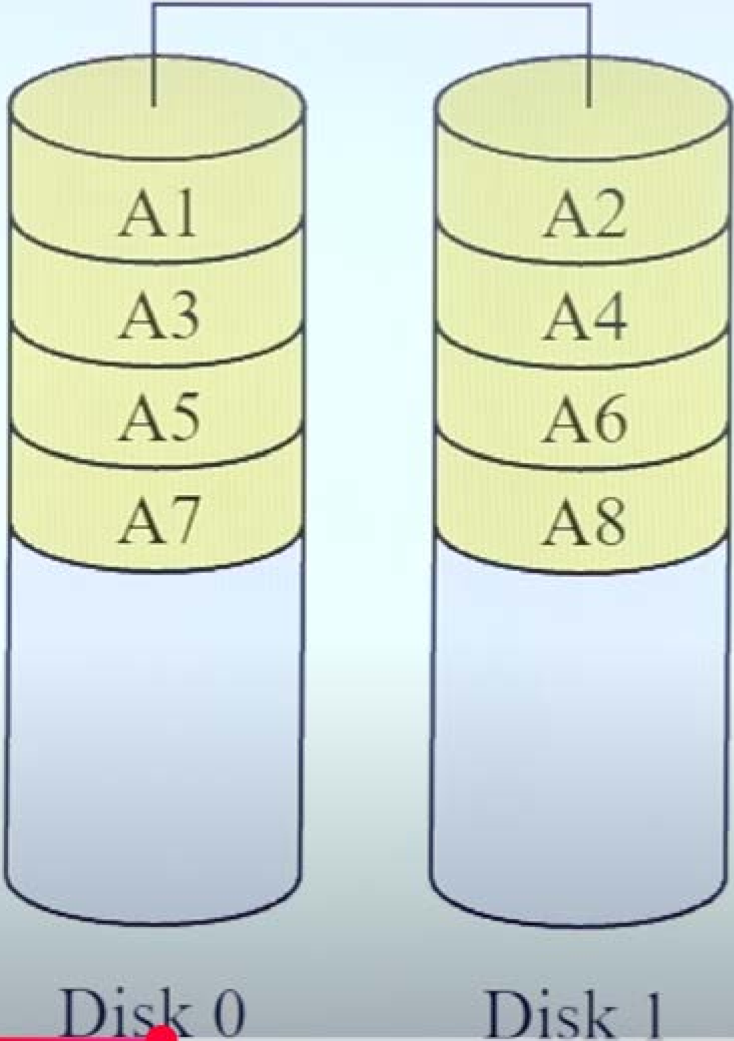
40.

0-15.

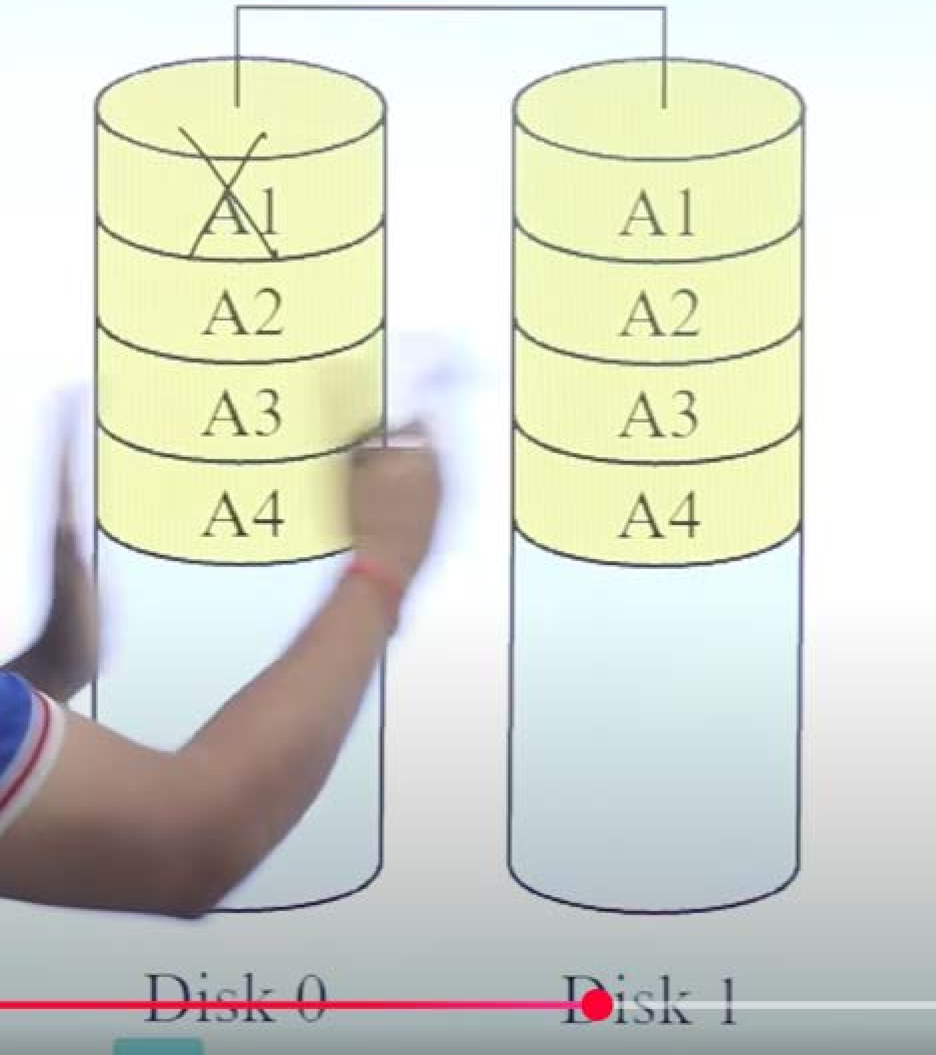
0000
1111] 16.

Actuator Arm

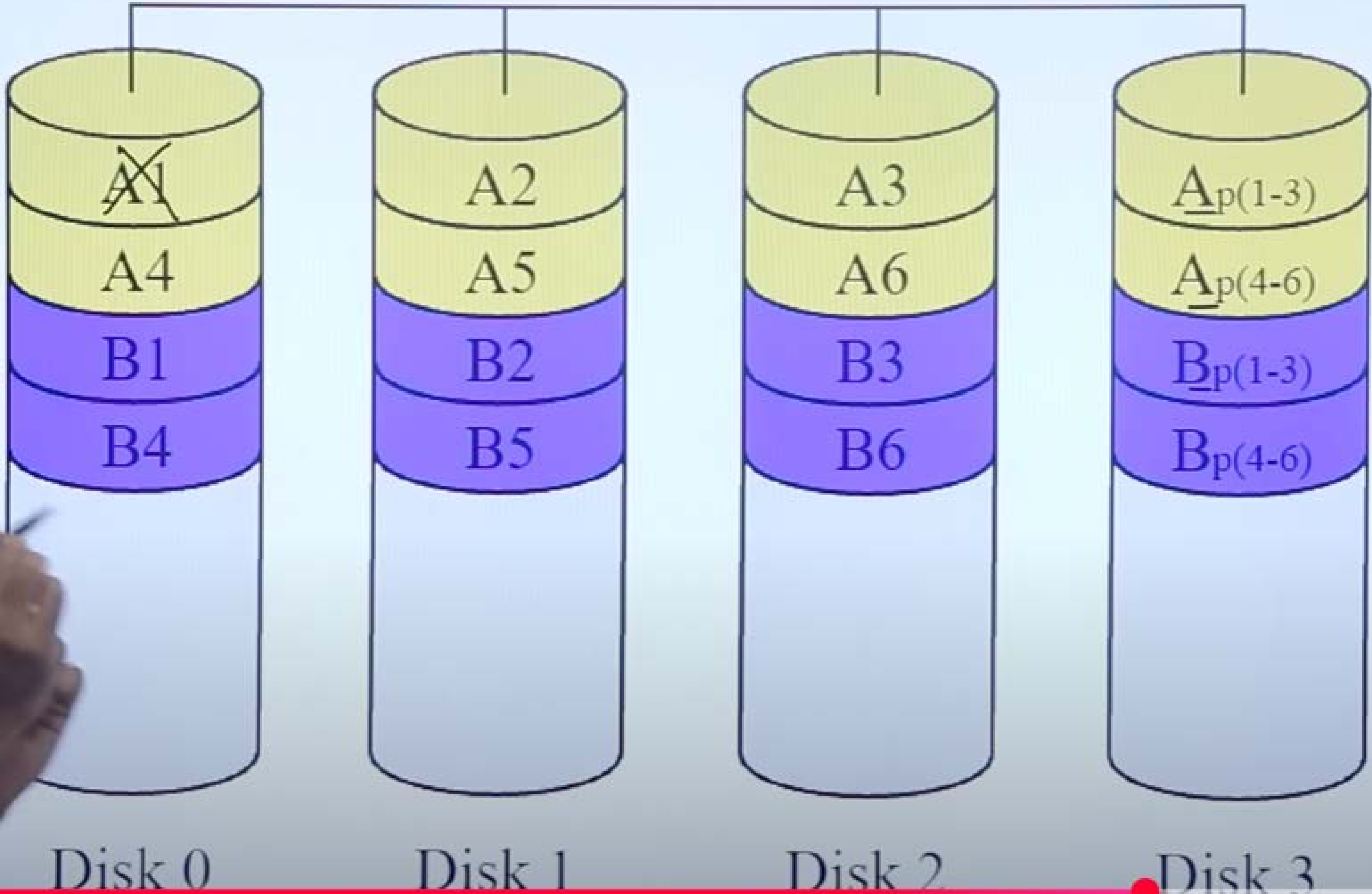
RAID 0



RAID 1

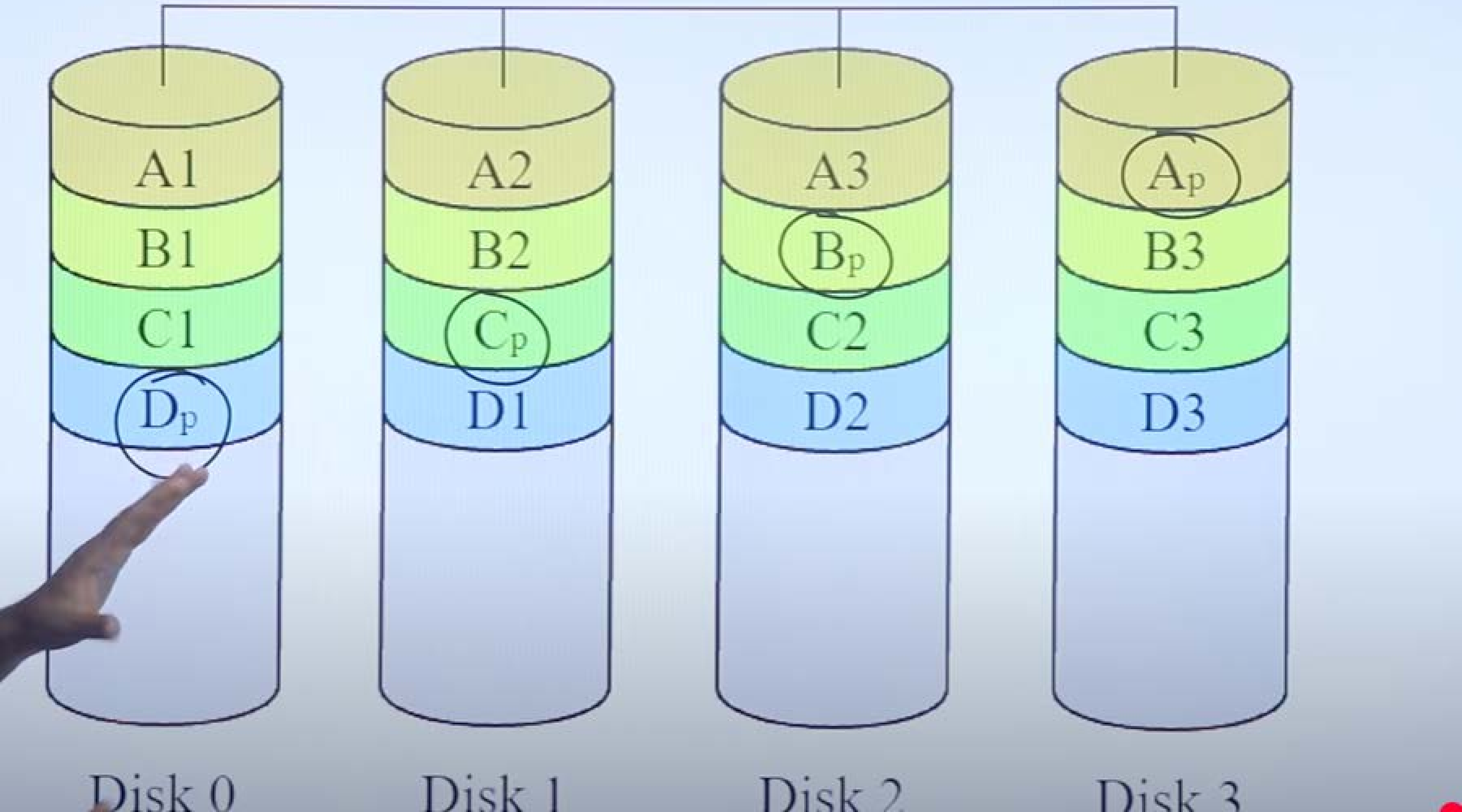


RAID 3



A1
A2
A3
Ap(1-3)
A4
A5
A6
Ap(4-6)
B1
B2
B3
Bp(1-3)
B4
B5
B6
Bp(4-6)
Disk 0
Disk 1
Disk 2
Disk 3

RAID 5



A1
A2
A3
Ap
B1
B2
Bp
B3
C1
Cp
C2
C3
Dp
D1
D2
D3
Disk 0
Disk 1
Disk 2
Disk 3