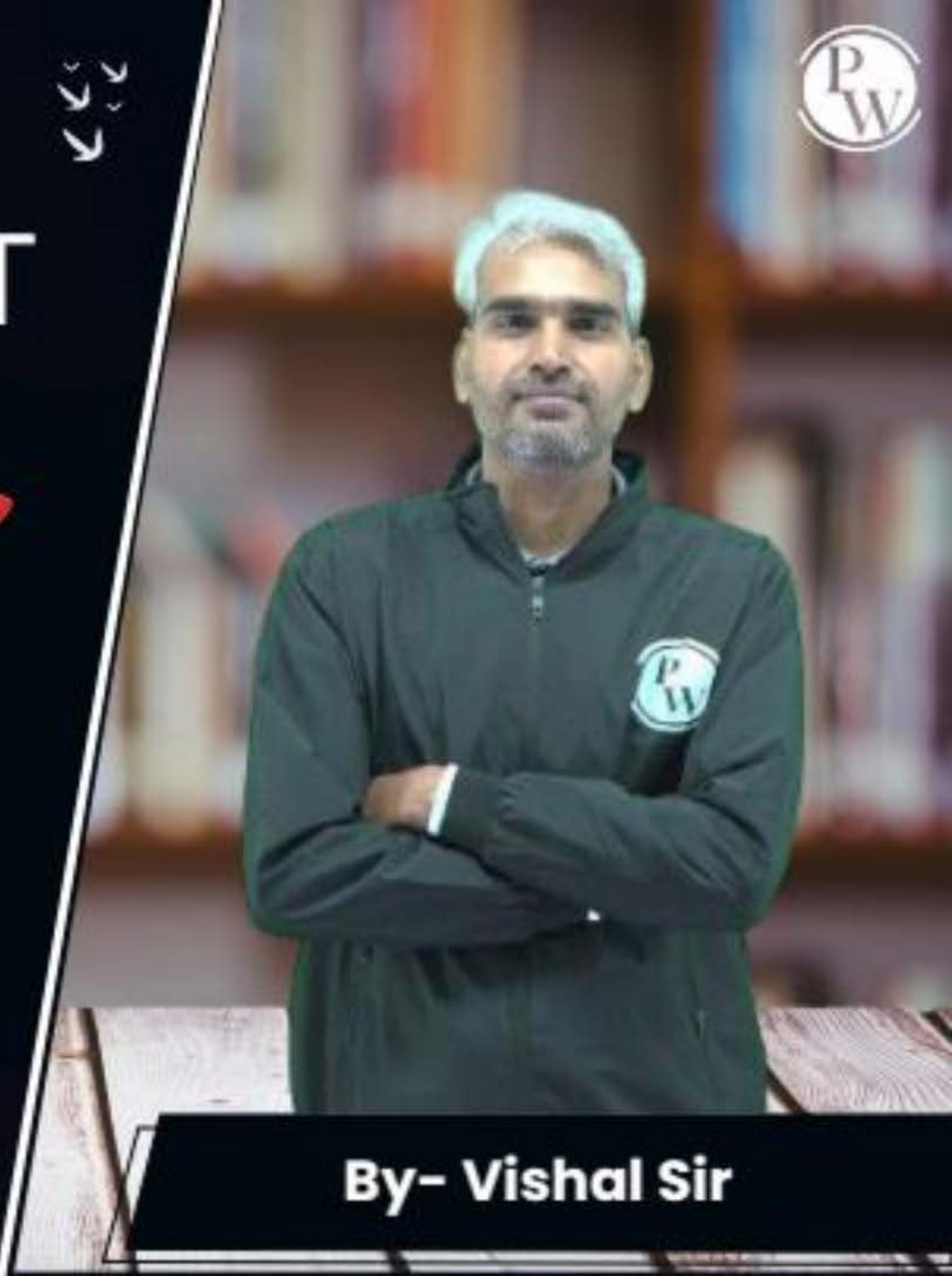
Computer Science & IT

Database Management System

File organization and indexing

Lecture No. 03



By- Vishal Sir

Recap of Previous Lecture

- Topic — Index file
- Topic — IO cost with index file
- Topic — Sparse and Dense index
- Topic — Primary, Clustering and Secondary index

Slide

Topics to be Covered

Topic — Multi-level index tree

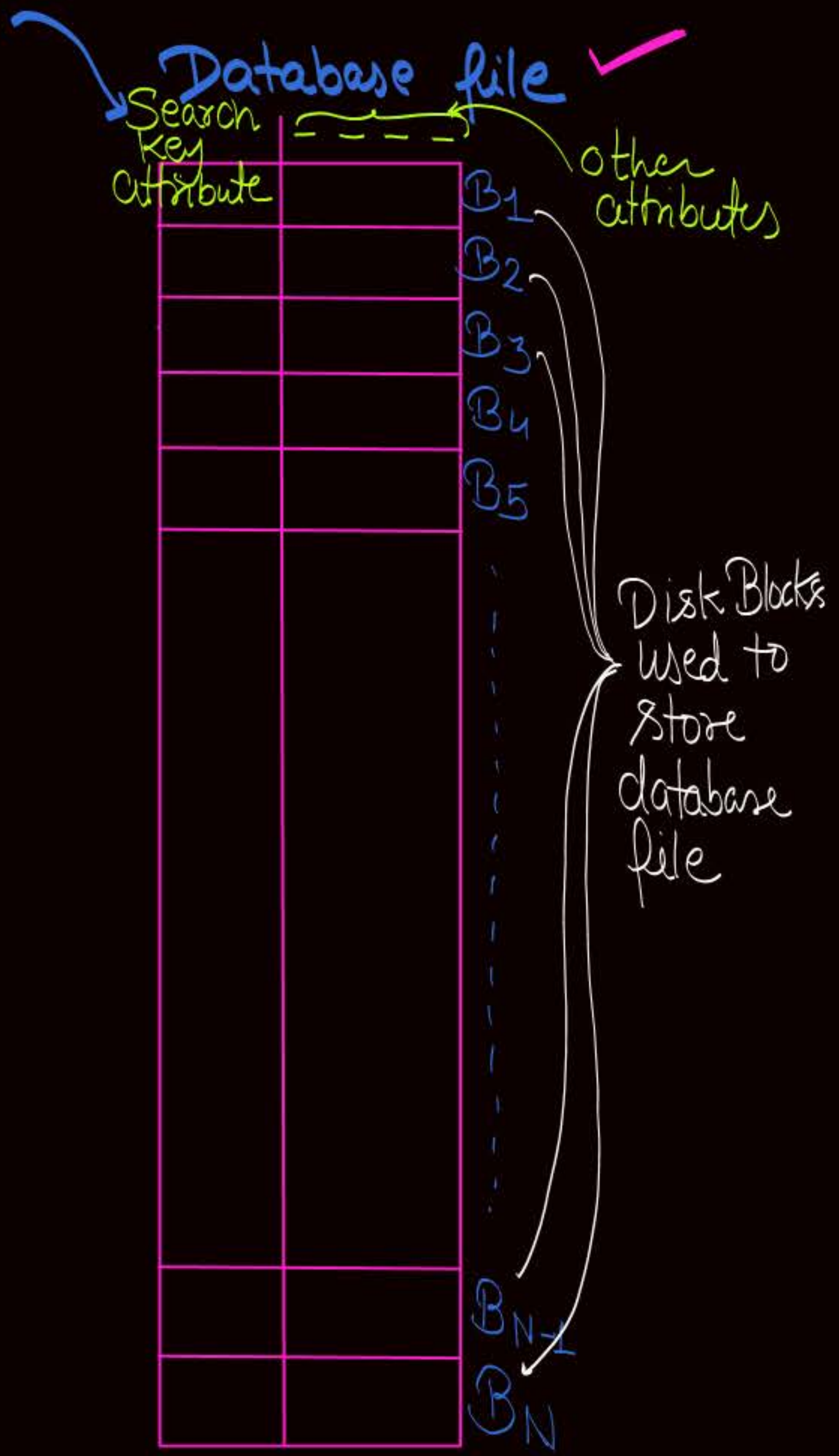
Topic — Structure of B tree

Topic — [redacted]

Slide

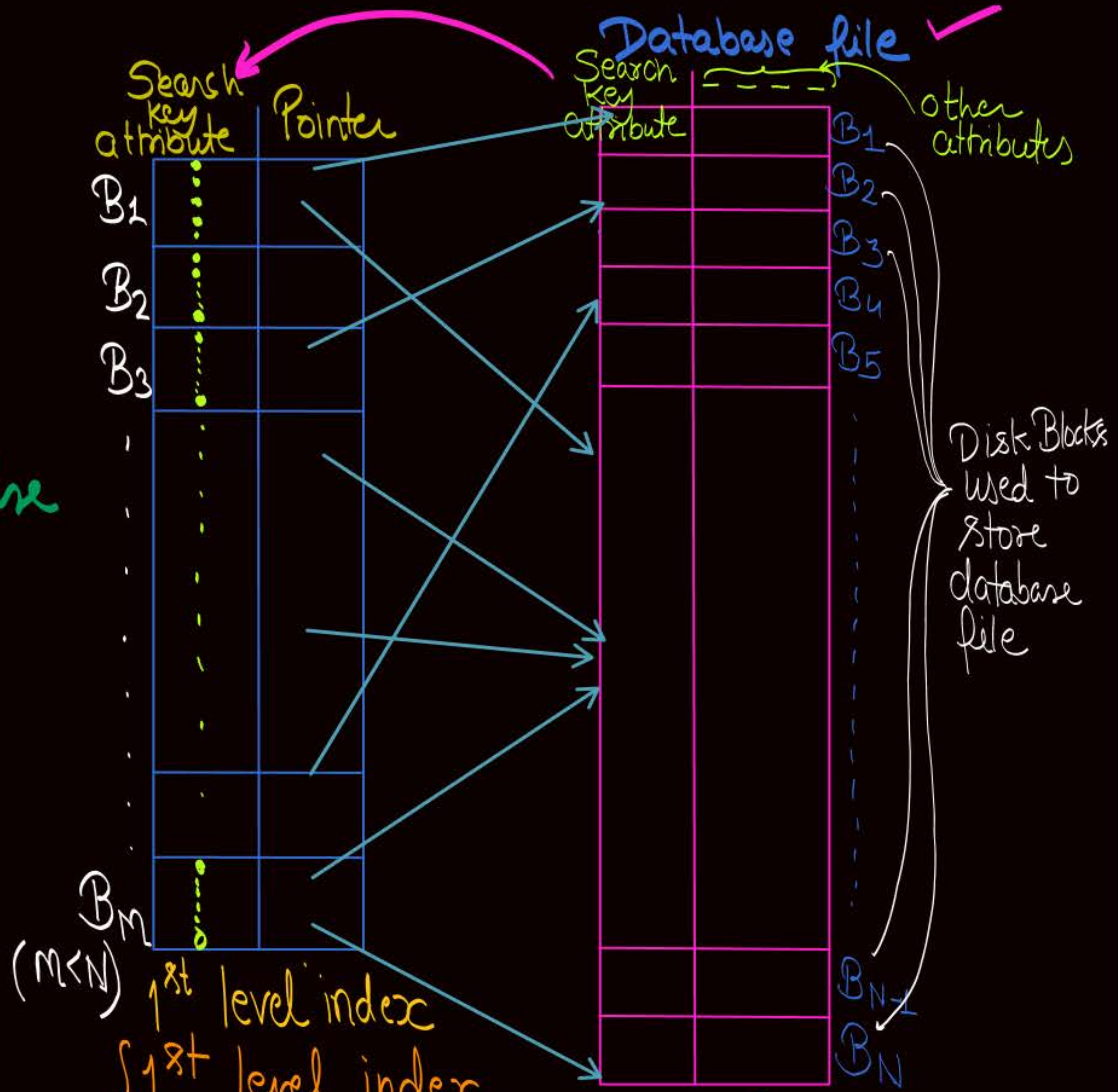
Topic : Multi-level Index

Multi-level index is used to reduce the IO Cost w.r.t. single level index


Database file
Search key attribute
other attributes
B1
B2
B3
B4
B5
BN-1
BN
Disk Blocks used to store database file

$\lceil \log_2 M \rceil + 1$

Will be Worst Case IO Cost using single level index

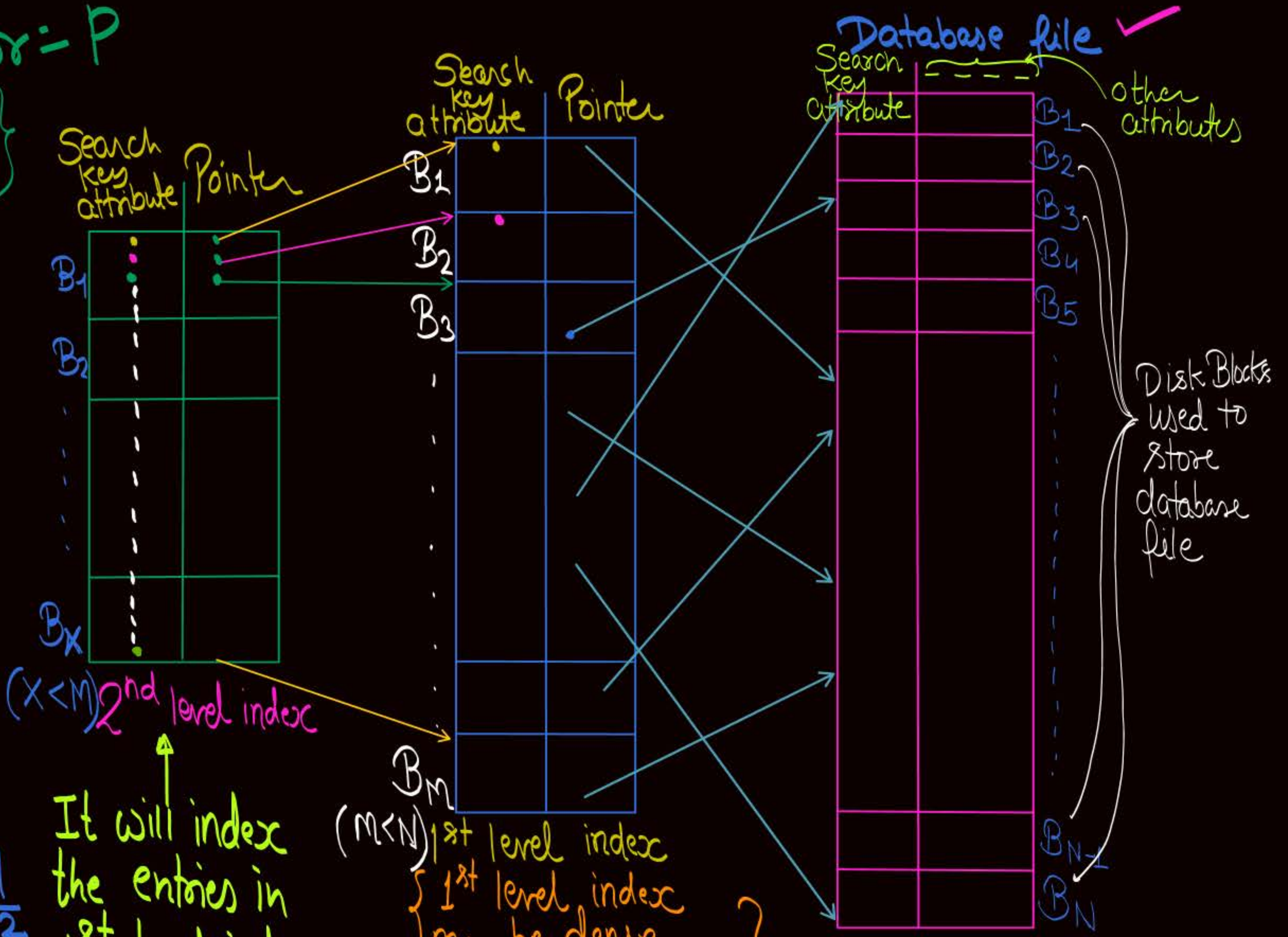


{1st level index
May be dense or May be Sparse}

let Blocking Factor = P
{w.r.t. blocks of 2nd level index}

then

$$\#\text{ disk blocks required for 2nd level index} = \left\lceil \frac{M}{P} \right\rceil$$



$$\#\text{ Disk block required for 3rd level index} = \left\lceil \frac{\left\lceil \frac{M}{P} \right\rceil}{P} \right\rceil \approx \frac{M}{P^2}$$

It will index the entries in 1st level index
(it will be sparse index)

{1st level index may be dense or may be sparse}

(Key size + Pointer size)

Entry size : 50B

$$Bf = \frac{1\,KB = \text{Block size}}{50\,B = \text{Entry size}}$$

$Bf \approx 20$

leaf → Previous → Previous → Previous - - -

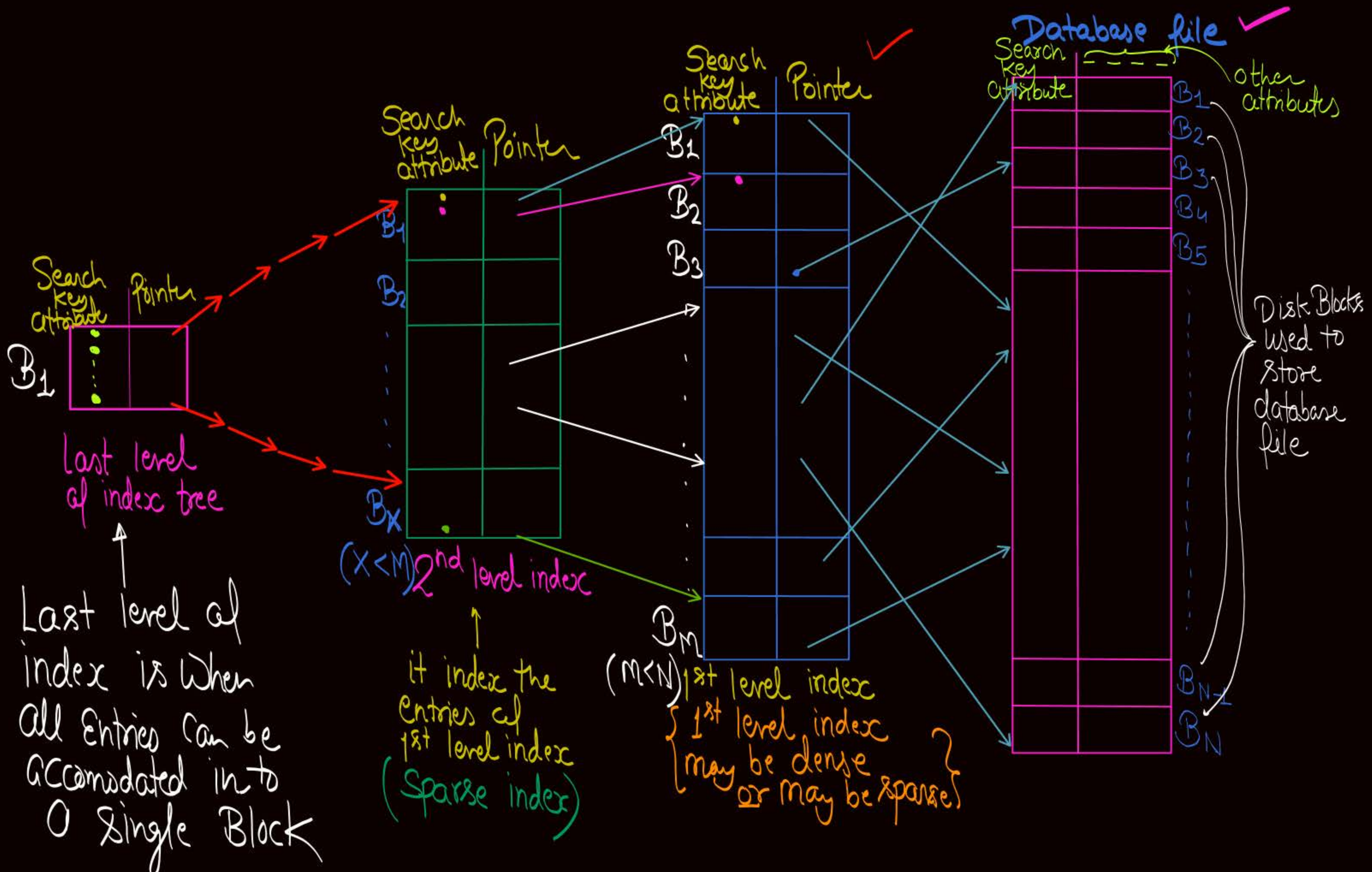
$M \qquad \frac{M}{2} \qquad \frac{M}{2^2} \qquad \frac{M}{2^3}$

$P \qquad P^2 \qquad P^3$

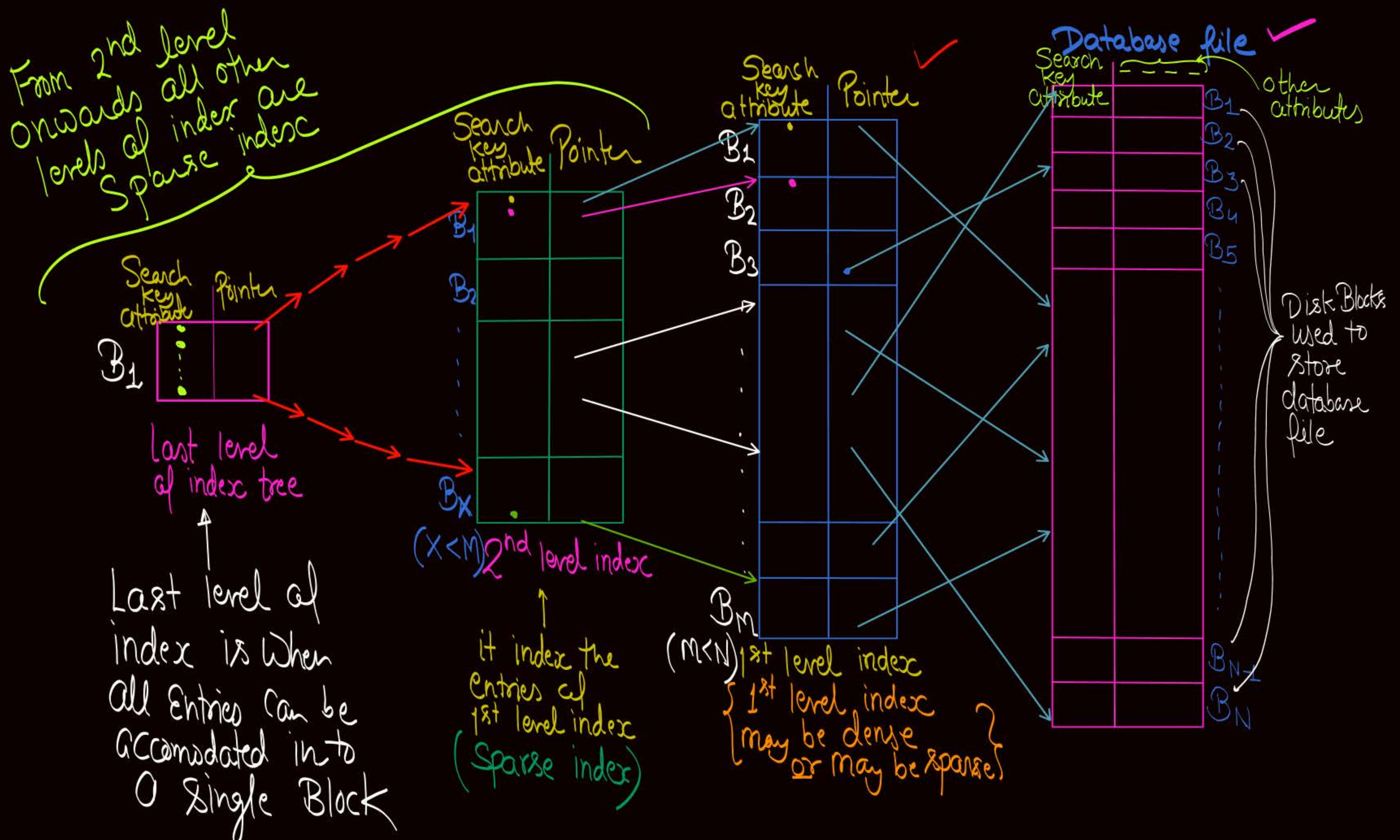
$20^2 \qquad 20^3$

$$\lceil \log_{P \approx 30} M \rceil$$

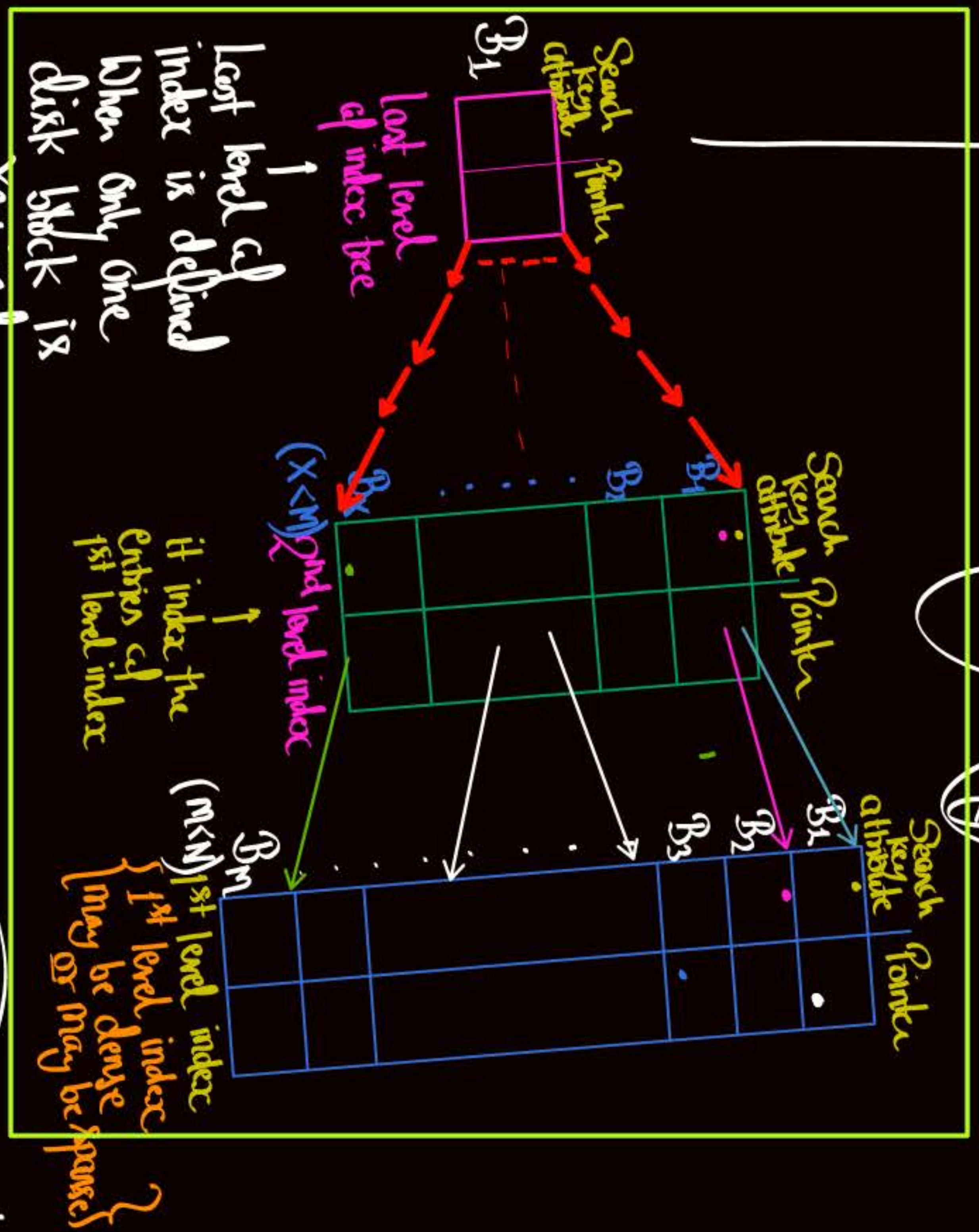
$$\boxed{\log_2 M \gg \log_{30} M}$$

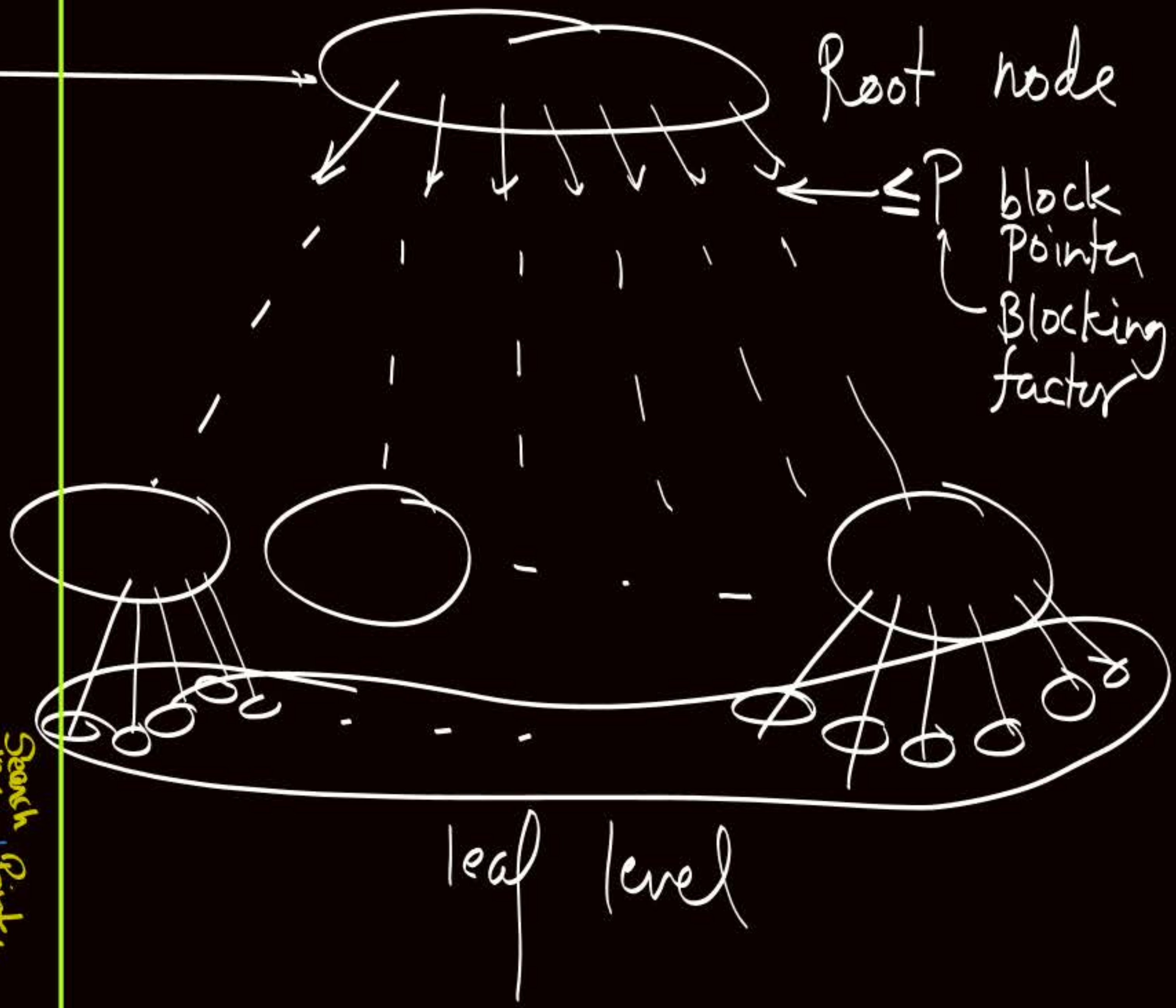


Database file
Search key attribute
other attributes
Search key attribute
Pointer
Search key attribute
Pointer
Search key attribute
Pointer
B1
last level of index tree
B1
B2
Bx
(X<M) 2nd level index
it index the entries of 1st level index
(Sparse index)
B1
B2
B3
BM
(M<N) 1st level index
{1st level index may be dense or may be sparse}
B1
B2
B3
B4
B5
BN-1
BN
Disk Blocks used to store database file
Last level of index is when all Entries can be accomodated into a single Block


From 2nd level onwards all other levels of index are Sparse index
Search key attribute
Pointer
B1
last level of index tree
Last level of index is when all Entries can be accomodated into a single Block
B1
B2
Bx
(X<M) 2nd level index
it index the entries of 1st level index
(Sparse index)
Search key attribute
Pointer
B1
B2
B3
BM
(M<N) 1st level index
{1st level index may be dense or may be sparse}
Database file
Search key attribute
other attributes
B1
B2
B3
B4
B5
BN-1
BN
Disk Blocks used to store database file

Multi-level index tree



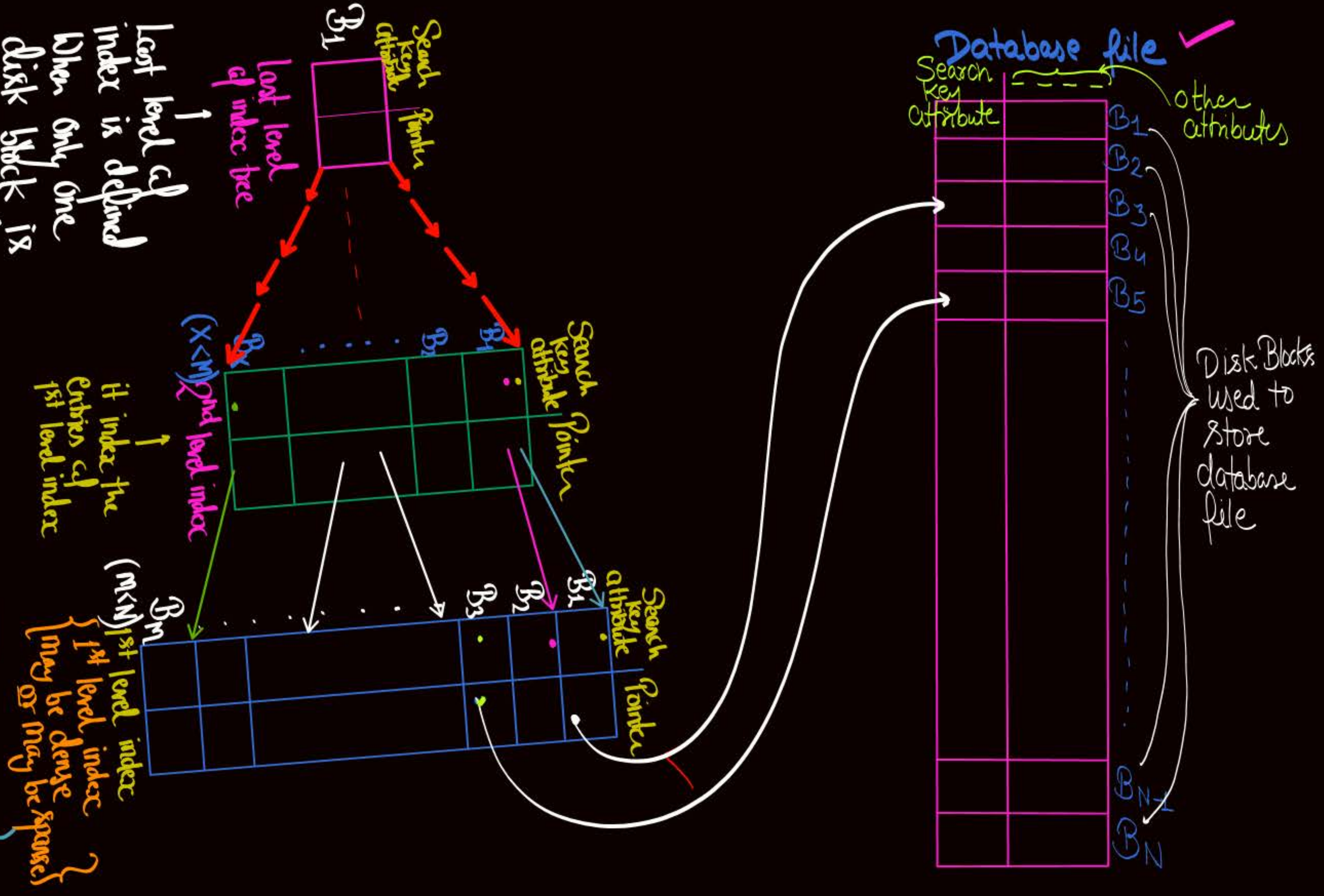


(leaf level of Multi-level index tree)

No. of levels in Multi-level index =

Last level of index is defined when only one disk block is required

(leaf level of Multi-level index tree)



Note :→

IO Cost using multi-level index

IO Cost using multi-level Index = (Number of levels in multi-level index tree) + 1

Number of levels in multi-level index tree $\approx \lceil \log_{Bf} M \rceil$

Bf: Blocking factor w.r.t. index block (2nd level onwards)

M: No. of disk blocks w.r.t. 1st level index

To locate an entry in the 1st level (leaf level) index. Starting from root level (from each level One block will be transferred)

1: To access the record from the database file based on the address obtained from the Entry of first level index

Note :→

$\boxed{\text{IO Cost using multi-level index}}$

$$\text{IO Cost using multi-level Index} = \left(\text{Number of levels in multi-level index tree} + 1\right)$$

it will be same for all access

i.e, Same IO Cost for all three cases,

Best Case, Worst Case, average Case

Topic : Multi-level Index

In multi-level index,

(1) 1^{st} level of index may be dense or may be sparse

(2) Except first level, all other levels of index are always sparse index

- Multi-level index is suitable only for static database i.e. the database in which insertion & deletion are very rare

- But if database is dynamic {i.e. insert and delete opⁿ are frequent} then after every insertion and/or deletion we may have to restructure all the levels of multi-level index. and it will be a time consuming operation

* ∴ for dynamic database we use

→ $\underbrace{\text{B tree or B+ tree}}_{\text{these are self Balancing multi-way search tree}}$

* $\underbrace{\text{Self-balancing multi-way search tree}}$

B tree
B+ tree

These are balanced ← i.e all leaf nodes are at same level

$\underbrace{\text{multi-way search tree}}$

m-ary search tree

it may be any value ≥ 2

Structure of B tree

Topic : B Tree

* $\boxed{\text{Order of a node of B tree}}$

In general, Order of a node of B tree is defined as the maximum number of Child pointer a node can have

Consider a B tree of order = P,

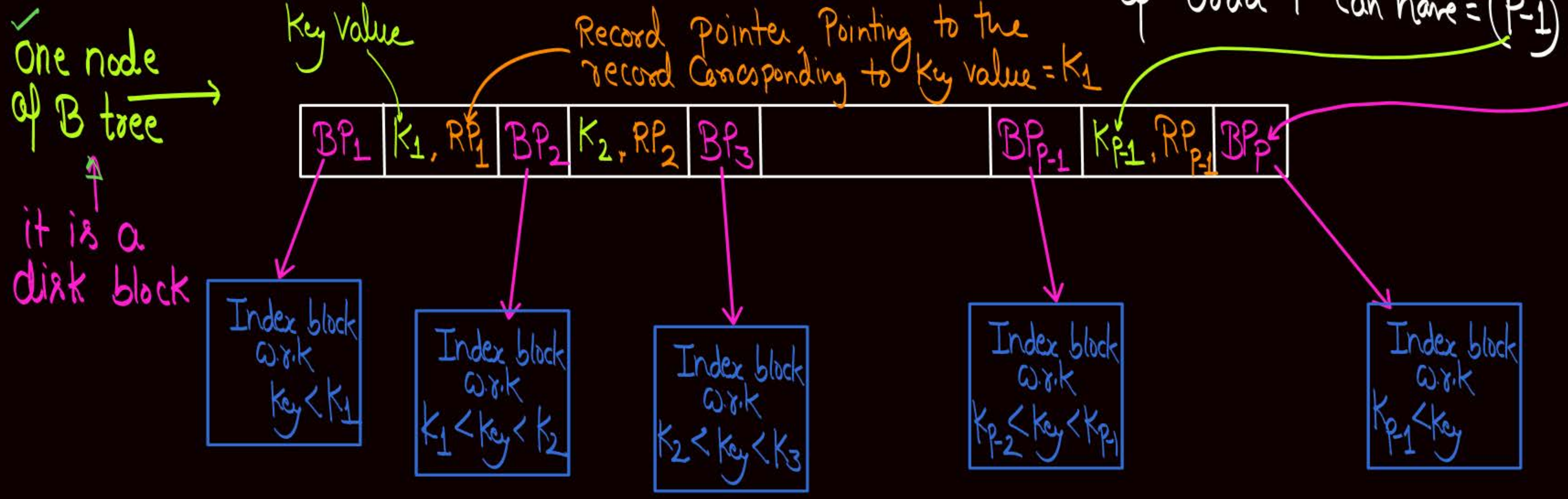
└ i.e. maximum number of Child pointer a node can have = P

∴ Maximum number of Keys a node of B tree of order 'P' can have = (P-1)

Structure of an internal node of B tree

- All keys within the node will always be stored in ascending order

  └ i.e. $K_1 < K_2 < K_3 - - - < K_{P-1}$



Consider a B tree of order = P,

↳ i.e. maximum number of Child pointer a node can have = P

∴ Maximum number of Keys a node of B tree of order 'P' can have = (P-1)

Structure of leaf node of B tree

All keys within the node will always be stored in ascending order

↳ i.e. $K_1 < K_2 < K_3 - - - < K_{P-1}$

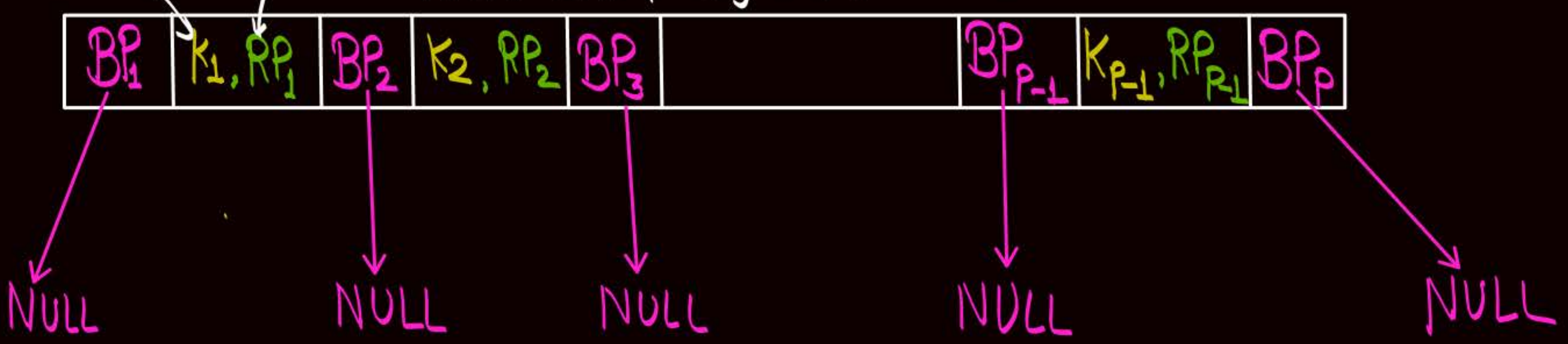
Key value = K_1

Record Pointer, pointing to the record corresponding to Key value = K_1

A node of B tree

↑

it is a block of disk



- BP (Block Pointer) : It is a pointer that Points to the index block at lower level of B tree

{ Child Pointer
{ node Pointer
{ Tree Pointer }

- RP (Record Pointer) : It Points to the record in the database file corresponding the associated key value

* **Note :-** In B tree, record pointers are present in internal node as well as in leaf node.

∴ Structure of B tree is different from the structure of multi-level index

* In multi-level index record pointers are present only at leaf level {i.e, 1^{st} level index}

Note:- Each node of B tree is a disk block

∴ whatever information we want to store in one node of B tree must fit in a single disk block.

Consider a B tree of order = P,

↳ i.e. maximum number of Child pointer a node can have = P

∴ Maximum number of Keys a node of B tree of order 'P' can have = (P-1)

Key value = K_1

Record Pointer, pointing to the record corresponding to Key value = K_1

One node of B tree

↑

One disk block

BP_1	K_1, RP_1	BP_2	K_2, RP_2	BP_3		BP_{P-1}	K_{P-1}, RP_{P-1}	BP_P

$$\left(\left(P * \text{Block Pointer size}\right) + (P-1) * \left(\text{Key field size} + \text{Record Pointer size}\right)\right) \le \text{Disk Block size}$$

Using this Equation we can determine the maximum order possible for a node of B tree

Topic : B Tree

Consider a relations R with a key filed K. A B-tree of order P is used to access structures on K. Where 'P' denotes the maximum Number of tree pointers in a B-tree index node. Assume that key (k) is 30 bytes long disk block size 1024 byte. Each data pointer size P_D is 12 bytes long & each block pointer P_B is 10 byte long. In order for each B-Tree Node to fit in a single disk block, the maximum value of P is-

Record

$$\left[\left(P * \text{Block Pointer Size}\right) + (P-1)\left(\text{Key field size} + \text{Record Pointer Size}\right)\right] \leq \text{Disk block Size}$$

(A) 19

(B) 20

(C) 21

(D) 22

$$(P * 10) + (P-1)(30+12) \leq 1024$$

$$52P \leq 1024 + 42$$

$$P \leq \frac{1066}{52} \Rightarrow \boxed{P \leq 20.5} \Rightarrow P_{max} = 20$$

Order = Max. no. of child pointer a node of B tree can have

Consider a B tree of order = p

Each non-root node of B tree must be at least half full.

- Maximum number of child pointer a node can have is = p
- Maximum number of keys a node can have is = (p-1)
- Minimum number of child pointer a non-root node must have is = $\lceil p/2 \rceil$
- Minimum number of keys a non-root node must have is = $\lceil p/2 \rceil$-1
- Minimum number of child pointer a root node must have is = 2
- Minimum number of keys root node must have is = 1

2 mins Summary

- **Topic** Multi-level index tree
- **Topic** Structure of B tree
- **Topic** [redacted]

Slide

THANK - YOU

Slide