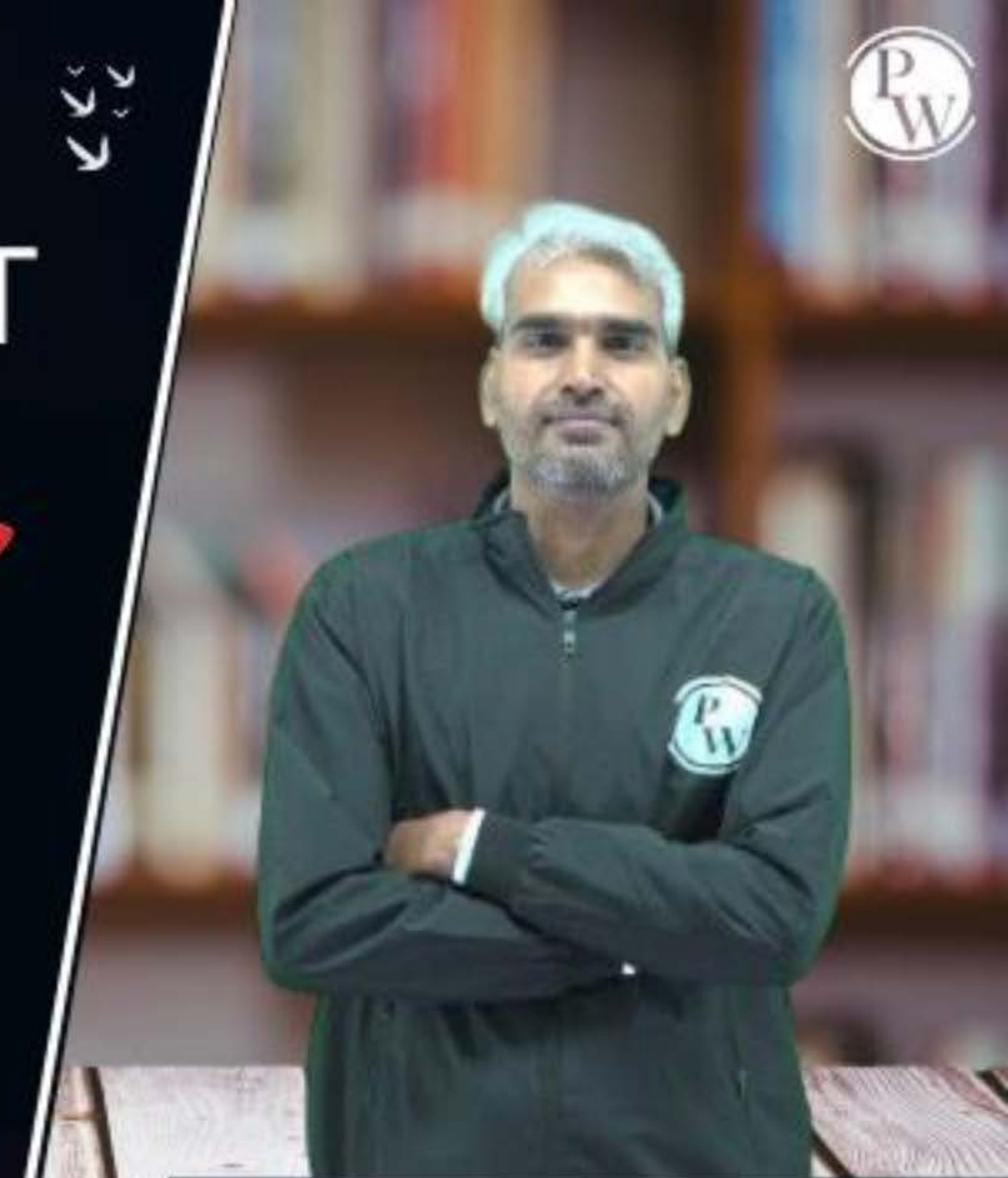
Computer Science & IT

Database Management System

File organization and indexing

Lecture No. 02



By- Vishal Sir

Recap of Previous Lecture

- **Topic** Database, Files and records
- **Topic** Organization of records
- **Topic** Blocking factor
- **Topic** IO cose without index file

Slide

Topics to be Covered

Topic Index file

Topic IO cost with index file

Topic Sparse and Dense index

Topic Primary, Clustering and Secondary index

Slide

Topic : Index file

Index file is used to reduce the IO Cost

- Each entry in the index file contains only two fields
  - * (1) Search key attribute value
  - * (2) Pointer pointing to the record / disk block corresponding to the search key value.

① Index file entry = < Search key value , Pointer information >

② Index file entry size = (Search key attribute size + Pointer size)

③ Database Record Size = (Summation of the sizes required to store each attribute of the database file)

④ Disk block size is same for database file and index file

⑤ Blocking factor (w.r.t. database block) $= \dfrac{\text{Disk block size}}{\text{Record size}} =$ No. of database records in each block

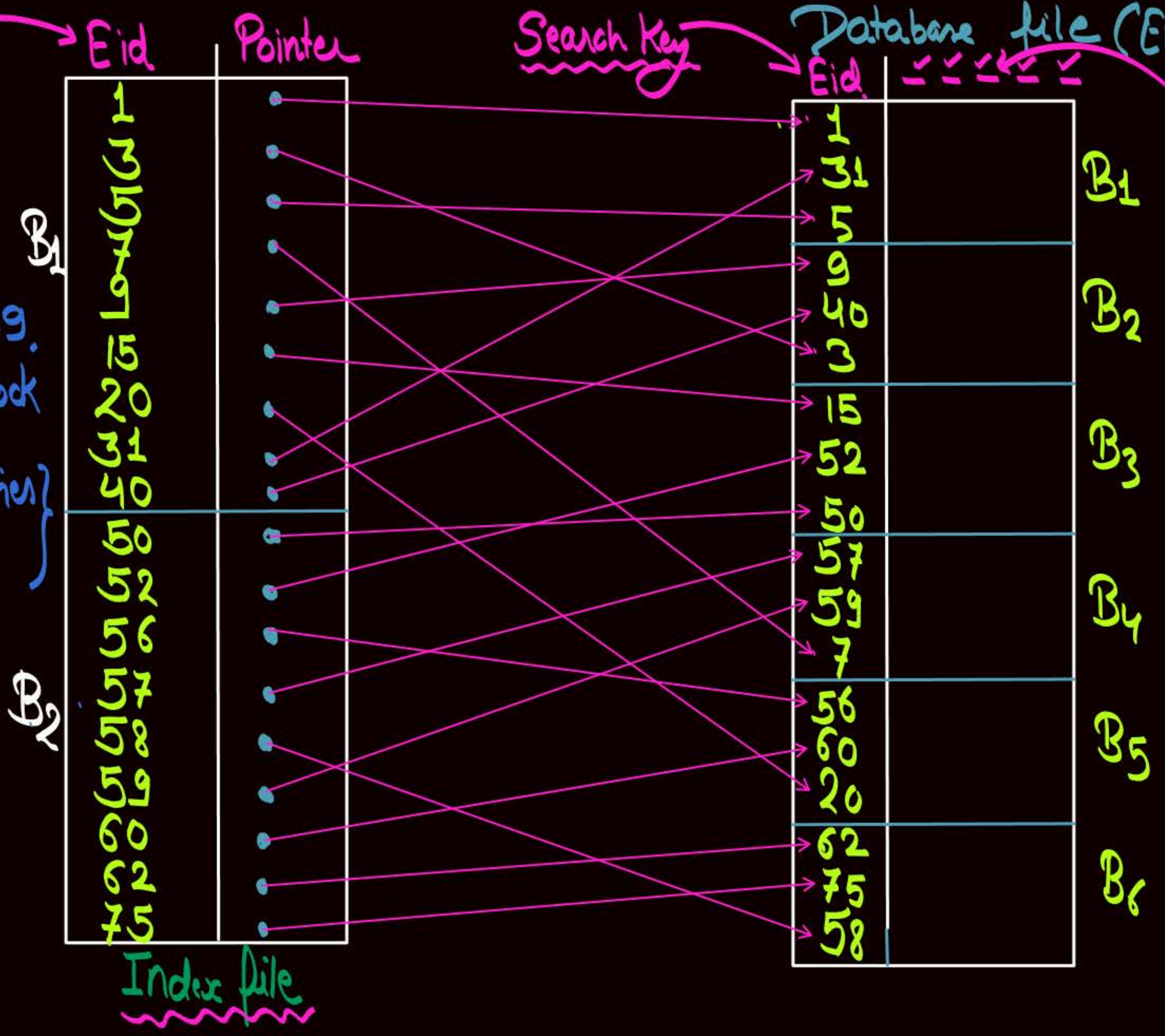
⑥ Blocking factor (w.r.t. index block) $= \dfrac{\text{Disk Block size}}{\text{Index file entry size}} =$ No. of index Entries in each block of disk

⑦ In general, Index file entry size < Database file record size

⑧ By ④, ⑤, ⑥ & ⑦

$$\boxed{\text{Blocking factor of index block} > \text{Blocking factor w.r.t. database block}}$$

✓ Search key



Various other attributes will be there

Consider blocking factor=9, w.r.t. index block {i.e, 9 index entries per block}

Consider blocking factor=3 w.r.t database block {i.e, 3 records per block}

Note:- Index file is always ordered based on search key values

Let,

No. of records in file = 100

No. of records per block = 5
(Blocking factor w.r.t. database block)

How many disk blocks required to store the file

$$= \frac{\text{Total no. of records in database file}}{\text{No. of records per block}}$$

$$= \frac{100}{5} = 20$$

(9) Number of disk blocks required to store database file

$$= \left\lceil \frac{\text{Total number of records in database file}}{\text{Number of records per block}} \right\rceil$$

$$= \left\lceil \frac{\text{Total number of records in database file}}{\text{Blocking factor w.r.t. database block}} \right\rceil$$

---

(10) Number of disk blocks required to store index file

$$= \left\lceil \frac{\text{Total number of entries in index file.}}{\text{Number of entries per block}} \right\rceil$$

$$= \left\lceil \frac{\text{Total number of entries in index file}}{\text{Blocking factor w.r.t index block.}} \right\rceil$$

"=" when index is dense index

(11) In general,

$$\boxed{\text{Number of entries in index file} \leq \text{Number of records in database file}}$$

"<" when index is Sparse index

(12) Index file will always be ordered based on search key attribute values.

∴ We can always perform "binary search" in index file

Topic : IO cost with indexing

- Let 'M' is the number of disk blocks required to store index file, and N is the number of disk blocks required to store the database file. $\boxed{\text{then in general, } M < N}$

$$\boxed{\text{Worst case IO Cost with index file} = \lceil \log_2 M \rceil + 1}$$

In order to search for an entry in the index file corresponding to search key value

In order to transfer the block of the database file w.r.t. address obtained from index file

Topic : Categories of Index

There are two categories of the indexes
- ① Dense index ✓
- ② Sparse index ✓

* ① **Dense index** :- If we maintain an entry in the index file for each record of the database file, then it is called dense index.

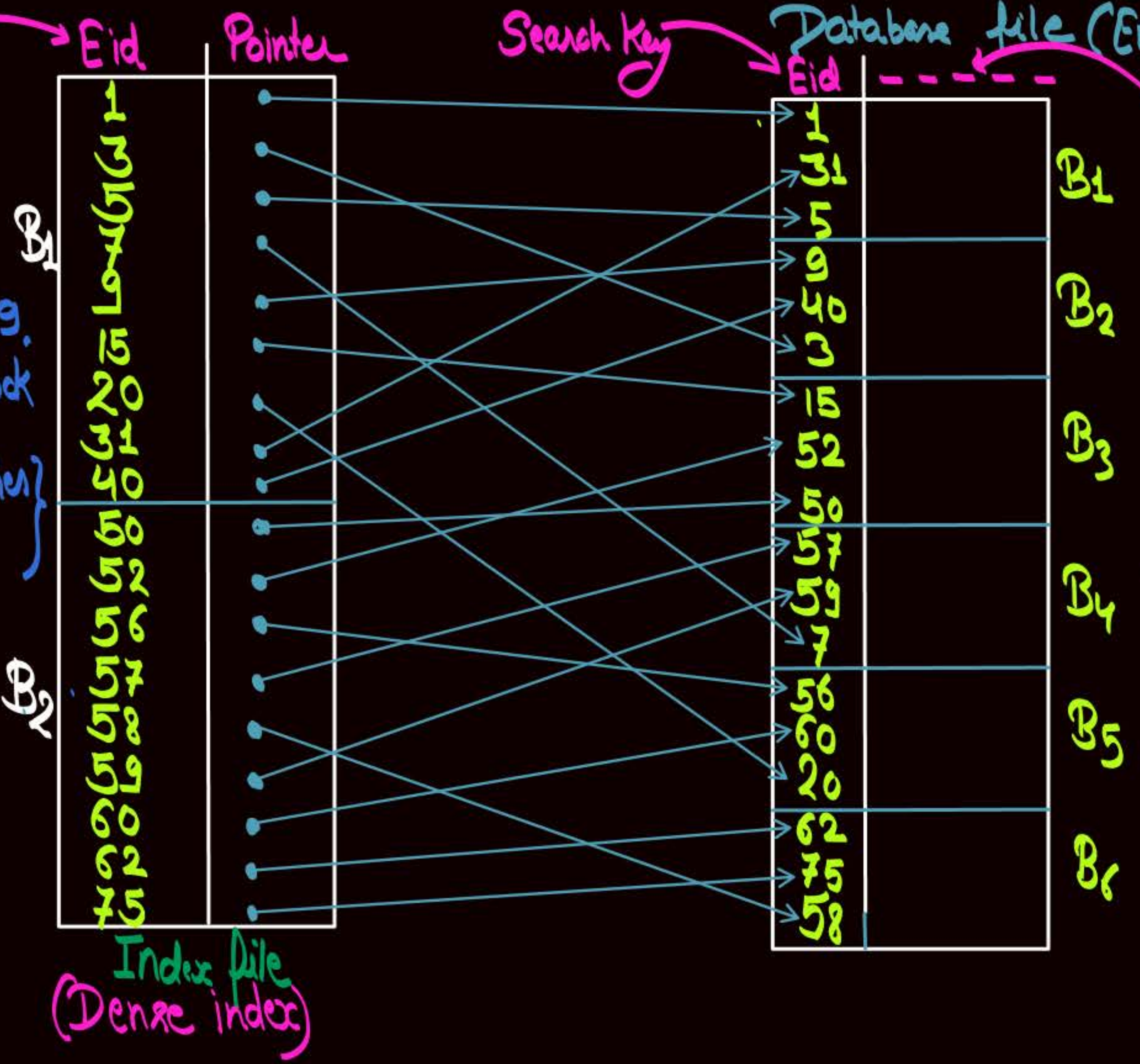
i.e; $\boxed{\text{Number of entries in index file w.r.t. dense index} = \text{No. of records in the database file}}$

Example of dense index :-

Generally dense index is created when database file is unordered w.r.t. search key

Search key

Consider blocking factor = 9 w.r.t. index block {i.e, 9 index entries per block}



Various other attributes will be there

Consider blocking factor = 3 w.r.t. database block {i.e, 3 records per block}

Note:- Index file is always ordered based on search key values

Note :- Dense index can be created for ordered file as well, but generall it is created for unordered file

Topic : Categories of Index

(2) Sparse index :- For a Collection of records in the database file we maintain only one entry in index file, then it is sparse index

i.e; $\boxed{\text{Number of entries in index file w.r.t. Sparse index} < \text{No. of records in the database file}}$

If index is sparse index and nothing else is given in the question about the number of Entries in the sparse index, then by default for each block of the database file we will maintain one entry in the index file

∴ By default,

$\boxed{\text{No. of Entries in Sparse index file} = \text{No. of disk blocks required to store database file}}$

Consider the following database file :

Database file (Employee)

Eid	- - -	
1, 3		B_1
4, 7		B_2
9, 10		B_3
12, 14		B_4
15, 18		B_5
20, 24		B_6
27, 30		B_7
35, 40		B_8

Search Key

File is ordered w.r.t. Search key

∴ Both, Sparse as well as dense index are Possible

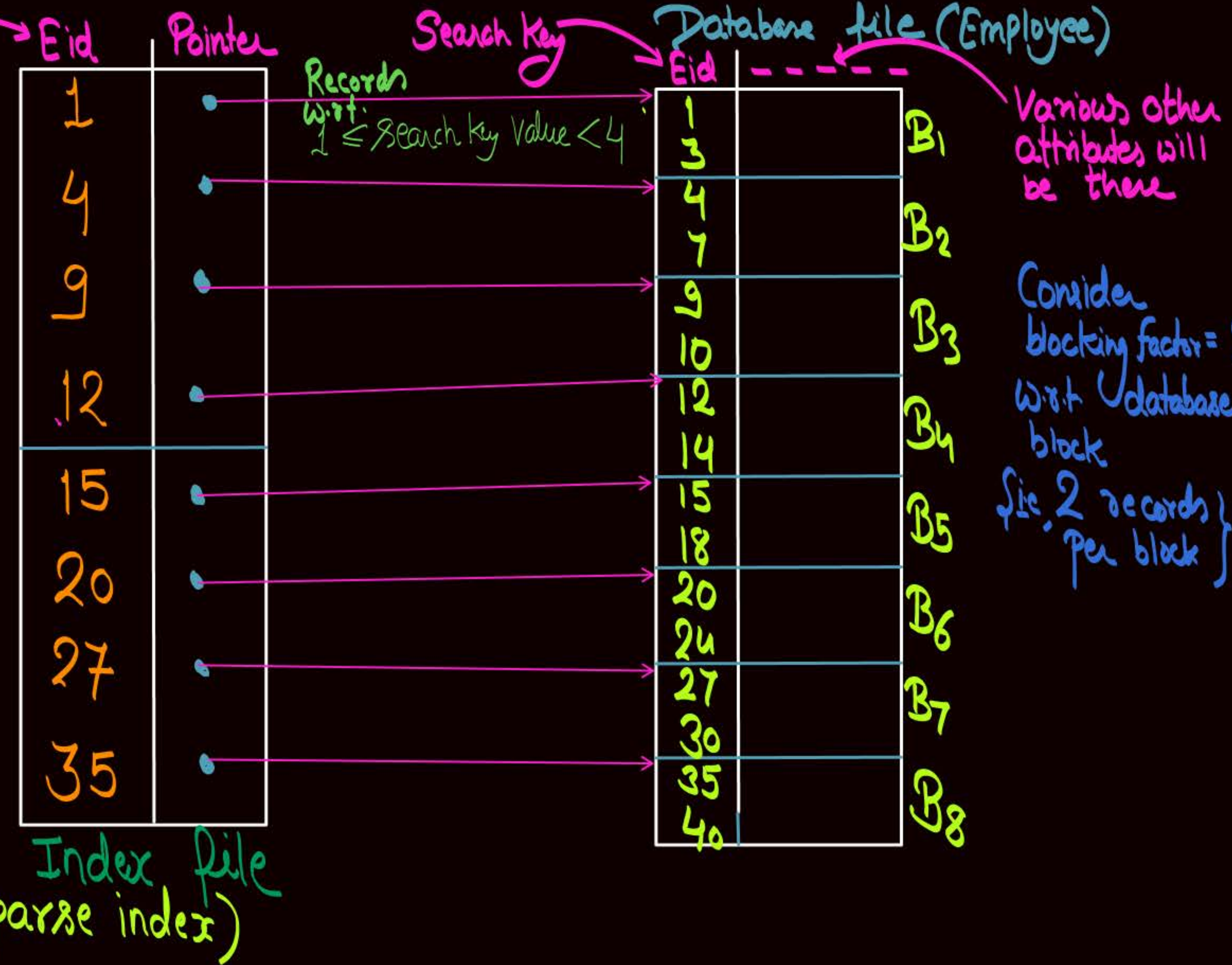
Various other attributes will be there

Consider blocking factor = 2 w.r.t database block

{i.e, 2 records per block}

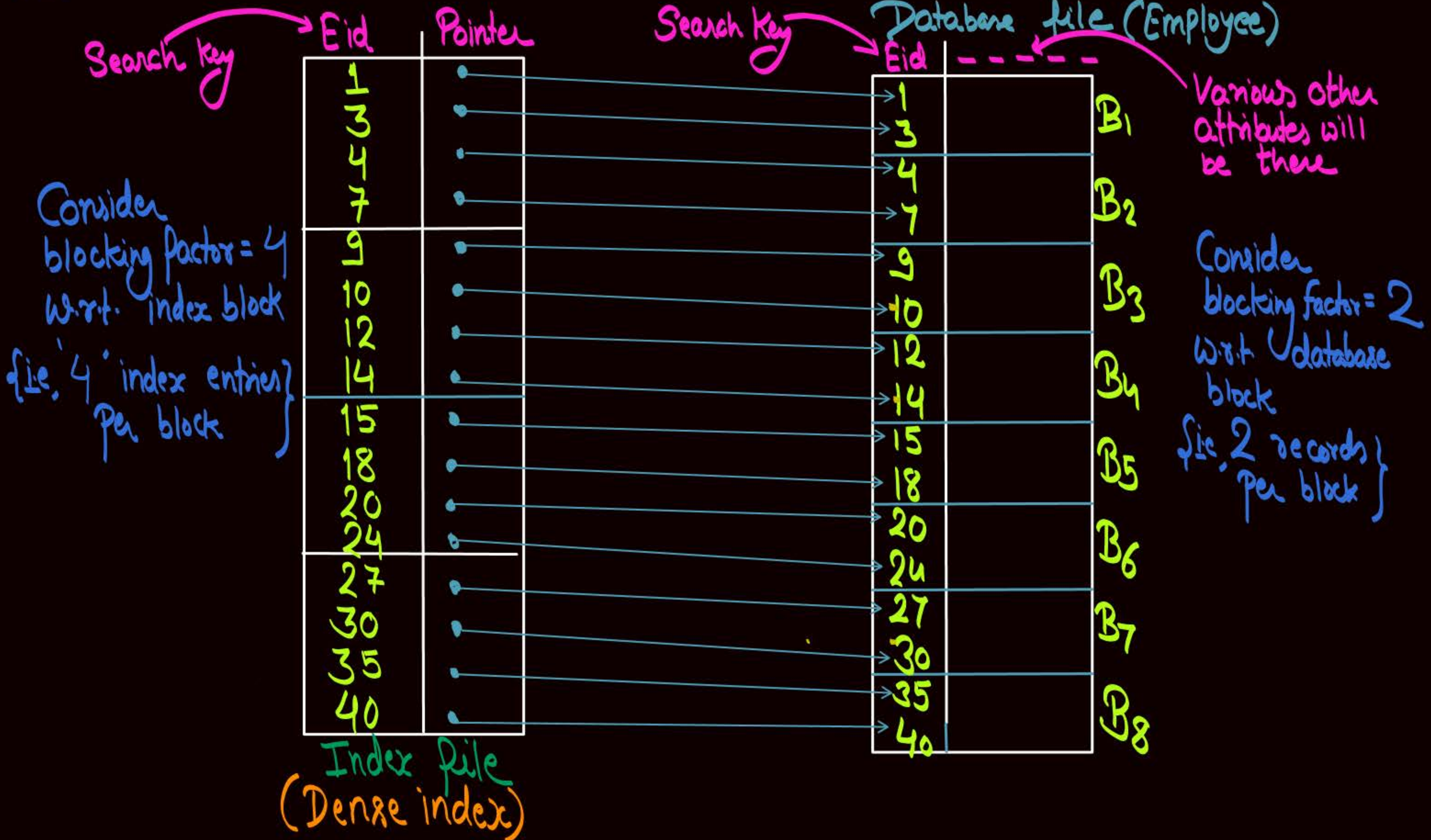
* Example of Sparse index file :-

Consider,
blocking factor = 4
w.r.t. index block
{ i.e. '4' entries per block



Consider
blocking factor = 2
w.r.t. database block
{ i.e. 2 records per block }

* Example of Dense index file w.r.t. ordered file :-



Note :-

1. If file is ordered based on search key attribute values, then both sparse as well as dense index are possible

2. If file is unordered based on search key attribute values, then only dense index is possible on that search key
{ Sparse index is never possible for an unordered file }

Note: ① Generally, we use record pointers in dense index

$\therefore$ Index file entry size w.r.t. dense index = (Search key size + Record pointer size)

② Generally, we use block pointers in sparse index

$\therefore$ Index file entry size w.r.t. sparse index = (Search key size + Block pointer size)

Note: If only one pointer is given, then use same for both dense index as well as sparse index

Q:- Consider the following specifications.

Disk block size = 1000 Bytes

Record size = 100 Bytes

Search Key attribute size : 12 Bytes

Record/Block pointer size = 8 Bytes

let total no. of records in the database file = 10,000

(1) How many disk blocks are required to store the dense index file on the above database file Ans = ? = 200

(2) How many disk blocks are required to store the sparse index file on the above database file Ans = ? = 20

(1) No. of disk blocks required to store dense index file $= \left\lceil \dfrac{\text{No. Entries in dense index}}{\text{No. of Entries per block}} \right\rceil$

$$= \left\lceil \frac{\text{No. of records in database file}}{\text{Blocking factor w.r.t. dense index block}} \right\rceil$$

$$= \left\lceil \frac{10,000}{\left(\dfrac{\text{Block size}}{\text{Index Entry size w.r.t. dense index}}\right)} \right\rceil = \left\lceil \frac{10000}{\left(\dfrac{1000}{\text{Key size} + \text{Record ptr size}}\right)} \right\rceil$$

$$= \left\lceil \frac{10,000}{\left(\dfrac{1000}{12+8}\right)} \right\rceil = \left\lceil \frac{10000}{50} \right\rceil = \boxed{200} = \underline{\text{Ans}}$$

② No. of disk blocks required to store Sparse index file

$$= \left\lceil \frac{\text{No. of Entries in Sparse index}}{\text{Blocking factor w.r.t Sparse index block}} \right\rceil$$

$$= \left\lceil \frac{\text{No. of disk blocks required for database file}}{\left(\dfrac{\text{Block size}}{\text{Index file Entry size w.r.t. sparse index}}\right)} \right\rceil$$

$$\#\text{ disk blocks for database file} = \left\lceil \frac{\text{Total no. of records}}{\text{Blocking factor w.r.t database block}} \right\rceil$$

$$= \left\lceil \frac{10{,}000}{\left(\dfrac{\text{Block size}}{\text{Record size}}\right)} \right\rceil$$

$$= \left\lceil \frac{10000}{\left(\dfrac{1000}{100}\right)} \right\rceil = 1000$$

$$= \left\lceil \frac{1000}{\left(\dfrac{\text{Block size}}{\text{Key size} + \text{Block pointer size}}\right)} \right\rceil = \left\lceil \frac{1000}{\left(\dfrac{1000}{12+8}\right)} \right\rceil$$

$$= \left\lceil \frac{1000}{50} \right\rceil = \boxed{20} = \text{Ans}$$

Topic : Types of Index

- Single level index
  - Primary index
  - Clustering index
  - Secondary index
- Multi-level index
- Dynamic Balanced Search tree (B tree & B+ tree)

Topic : Primary Index

If ① Search key attribute is a key {i.e. Unique values in file}
and ② File is ordered based on that search key attribute,

then Index file created on such search key is called primary index.

Topic : Primary Index

* Eid is a key attribute in database file (i.e. unique values of Eid)
&
* Records of EMP database file are ordered based on Eid

∴ $\boxed{\text{Index is Primary index}}$ ✓

Search key + Unique value

Search key (Eid)	Pointer	Block
1		B_1
9		B_1
13		B_1
20		B_2
27		B_2

Index file.
Primary index
{It can be dense or sparse}

E.M.P.

Eid		Block
1		B_1
7		B_1
9		B_2
10		B_2
13		B_3
16		B_3
20		B_4
24		B_4
27		B_5
30		B_5

{In general, Primary index is sparse index}

Topic : Clustering Index

* If (1) Search key attribute is non-key attribute {i.e. not unique values in database file}
- and (2) File is ordered on search key attribute,

then index file created on such search key is called a Clustering index

Topic : Clustering Index

Dept-id values are not unique in database file
&
Records of EMP database file are Ordered based on Dept-id

∴ Index will be Clustering index

Search key Dept-id	Pointer
1	
2	
4	
5	
7	

Index file
(Clustering Index)

Search key values are not unique
+
File is ordered w.r.t. Dept-id

EMP

	Dept-id	Eid	- - - -
B_1	1	7	
	2	13	
B_2	2	1	
	2	5	
B_3	4	4	
	5	9	
B_4	5	15	
	7	6	
B_5	7	20	
	7	12	

Block Anchor

Topic : Clustering Index

Dept-id values are not unique in database file
&
Records of EMP database file are Ordered based on Dept-id

∴ Index will be Clustering index

Index file (Clustering Index)

Search key Dept-id	Pointer
1	
2	
4	
5	
7	

It is the addr of the first record in the database file Corresponding to the Dept-id = 2

* If record in the database block is the last record of the block, then we don't know whether it is the last record Corresponding to the search key value or we have more records in the next block of the file.

Search key + values are not unique + File is ordered w.r.t. Dept-id

EMP

Dept-id	Eid	- - - -
1	7	
2	13	
2	1	
2	5	
4	4	
5	9	
5	15	
7	6	
7	20	
7	12	

B_1, B_2, B_3, B_4, B_5

Block Anchor

∴ We will use the block anchor to access the next block of the file to Check if that was the last record Corresponding to the search key value or not

Note :→

1. Primary index can be sparse index or dense index {generally, Primary index is sparse}

2. Clustering index is always a Sparse index.

3. At most one primary index is possible on a database file {because file must be ordered based on search key}

4. At most one Clustering index is possible on a database file. {because file must be ordered based on search key}

5. On a database file we can either have a primary index or we can have a Clustering index but not both.
Because at a time file can be ordered w.r.t. at most one attribute

Topic : Secondary Index

* If file is unordered based on search key, {Search key attribute values may or may not be unique in the file}
* then index file created on such search key is called secondary index

Topic : Secondary Index

* File is unordered based on Pass-no.

o: Secondary index

Secondary index Is always dense index

Pass-No	Pointer
1	
2	
4	
7	
9	
10	
12	
13	

Rating	Pointer

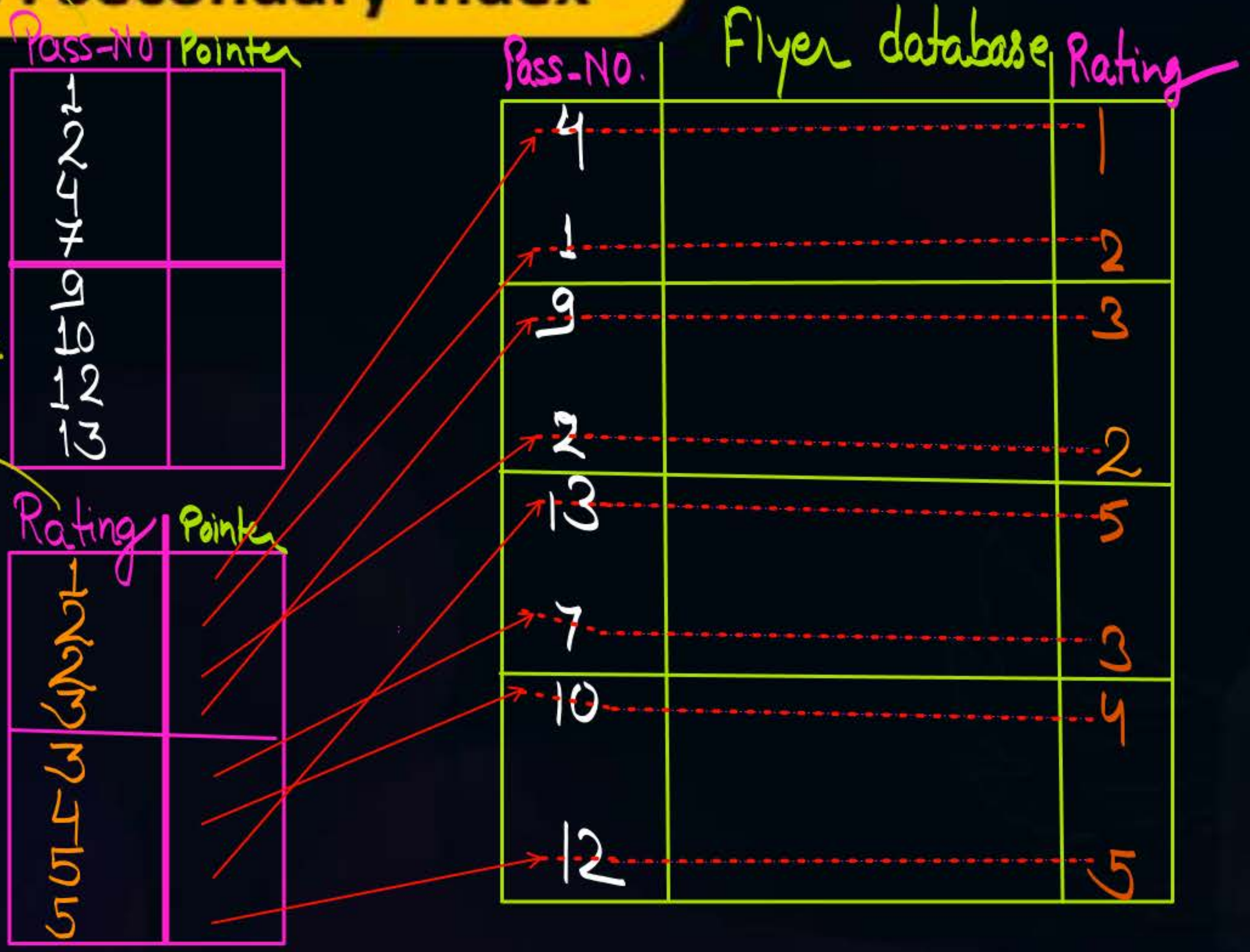
Flyer database

Pass-NO.		Rating
4		1
1		2
9		3
2		2
13		5
7		3
10		4
12		5

Topic : Secondary Index

File is unordered based on Pass-no.
∴ Secondary index

Database file is unordered based on "Rating"
∴ Secondary index



Note:- When file is unordered based on search key, then our first objective will be to provide an order using index file so that binary search can be applied

∴ Secondary index is always a dense index.

Note:- Any number of secondary index are possible on a database file { Because file don't need to be ordered }

2 mins Summary

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

Slide

THANK - YOU

Slide