GATE

CRASH COURSE

DS & AI

Database Management System

Indexing & Multilevel indexing

By – Vishal Rawtiya Sir

Lecture No. 05

Topics *to be covered*

1. Files, records and database
2. Organization of records
3. Indexing
4. Categories and types of index

Topic : WITH clause

★ {"With" is used along with "As"}

- The WITH Clause is mainly used to provide a subquery block a name that can be referenced within the main SQL query or other subquery that follows.

Syntax of WITH:

Here we will define a new schema

Here we have a sub-query

- WITH New_relⁿ_name(, , , ...) AS Sub-query block

list of new names for the attributes

The output of this subquery will be assigned a new-name by 'WITH' Clause

We may have multiple such sub-query with "WITH" clause.

In the end we will have our main query:

Select _ _ _ ... } our main query

#e.g. Consider the following database table named water_scheme

water_scheme		
scheme_no	district_name	capacity
1	Ajmer	20
1	Bikaner	10
2	Bikaner	10
3	Bikaner	20
1	Churu	10
2	Churu	20
1	Dungargarh	10

The number of tuples returned by the following SQL query is _____

```
with total(name, capacity) as
    (select district_name, sum(capacity)
     from water_schemes
     group by district_name)
with total_avg(capacity) as
    (select avg(capacity)
     from total)
select name from total, total_avg
where total.capacity >= total_avg.capacity
```

New schema

Sub-query ①

Subquery

Main query

name	total capacity
Dist.name	Sum(capacity)
Ajmer	20
Bikaner	40
Churu	30
Dungargarh	10

total_avg
capacity
Avg(capacity)
25

name	total capacity		total_avg
Ajmer	20	≱	25
Bikaner	40	≥	25
Churu	30	≥	25
Dungargarh	10	≱	25

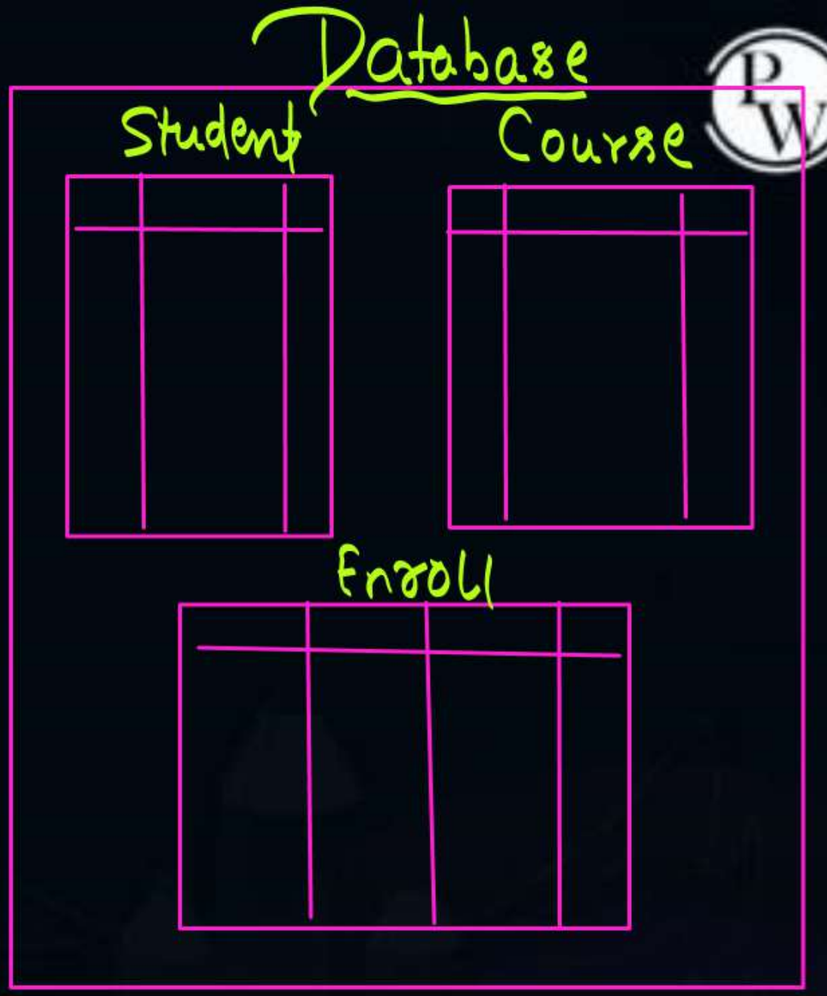
name
Bikaner
Churu

Indexing

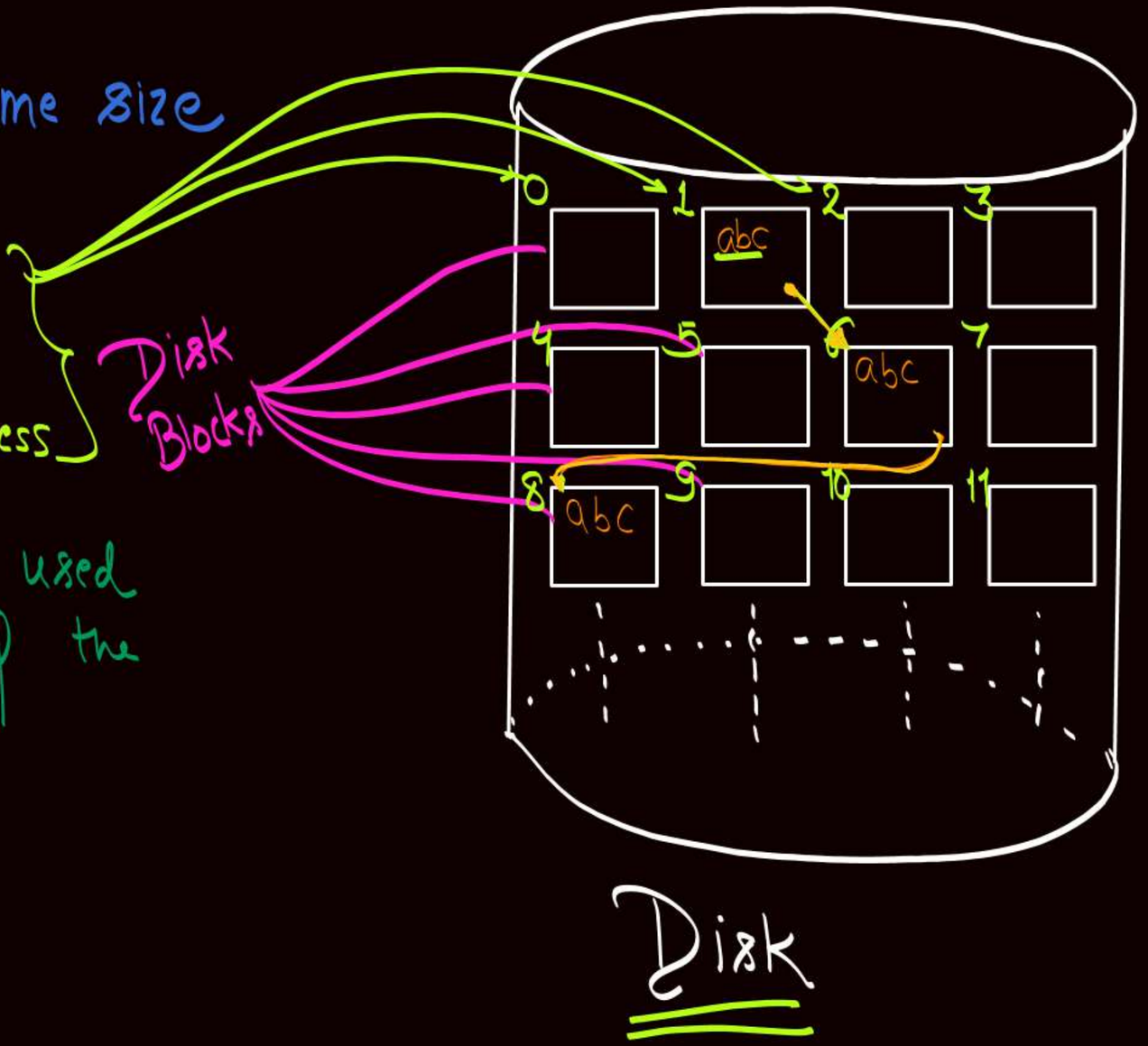
Topic : Database-File-Records

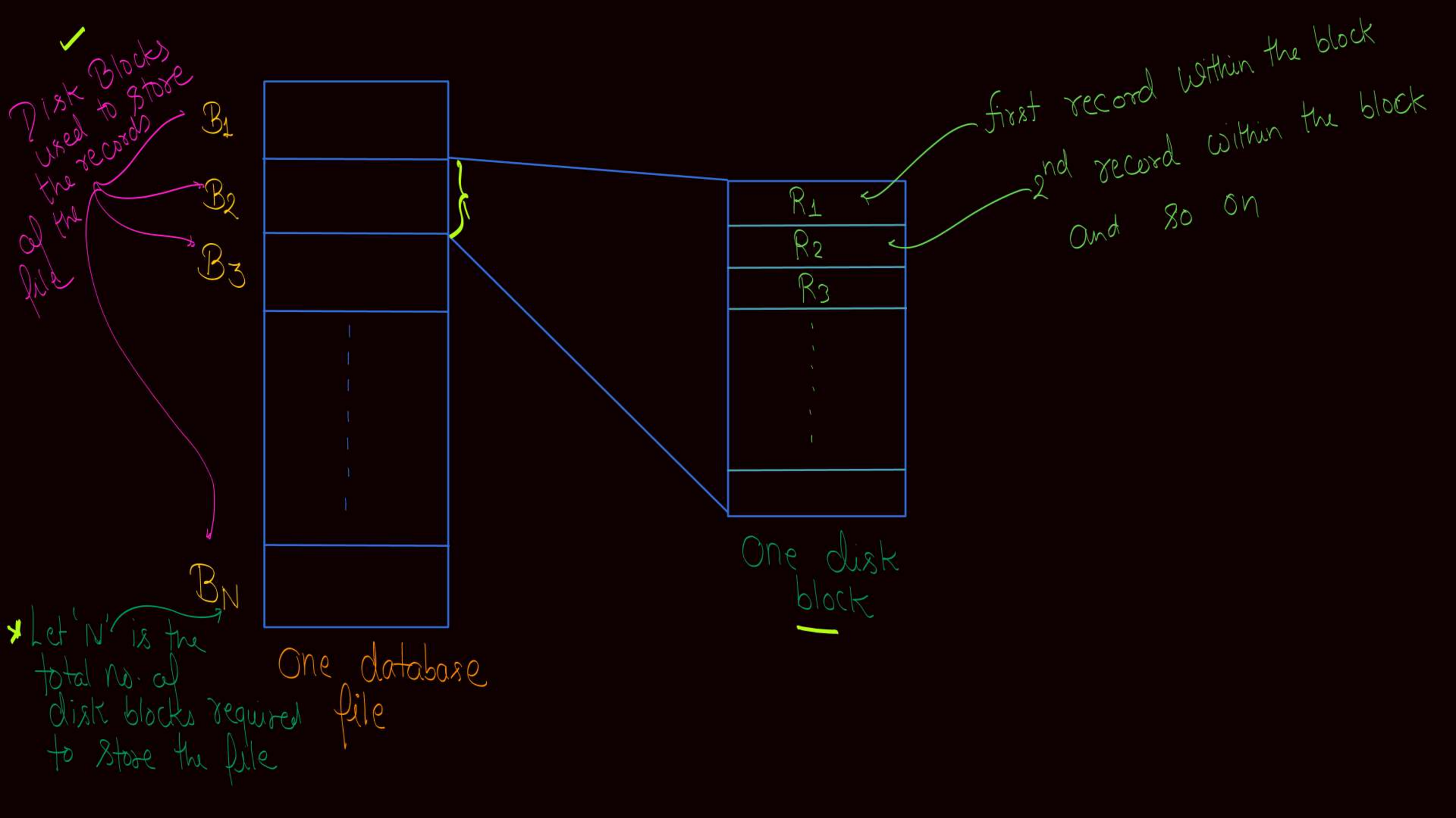
* Database is collection of files
* Files are used to store records
* Disk blocks are used to store the records of a file

Records of different files can not be stored in the same block of disk



- Each disk block within the disk will be of same size
- Each block of disk is identified by a unique no. known as disk block address
- Each disk block can be used to store multiple records of the same file




Disk Blocks used to store the records of the file
B_1
B_2
B_3
B_N
* Let 'N' is the total no. of disk blocks required to store the file
One database file
R_1
R_2
R_3
One disk block
first record within the block
2nd record within the block
And so on

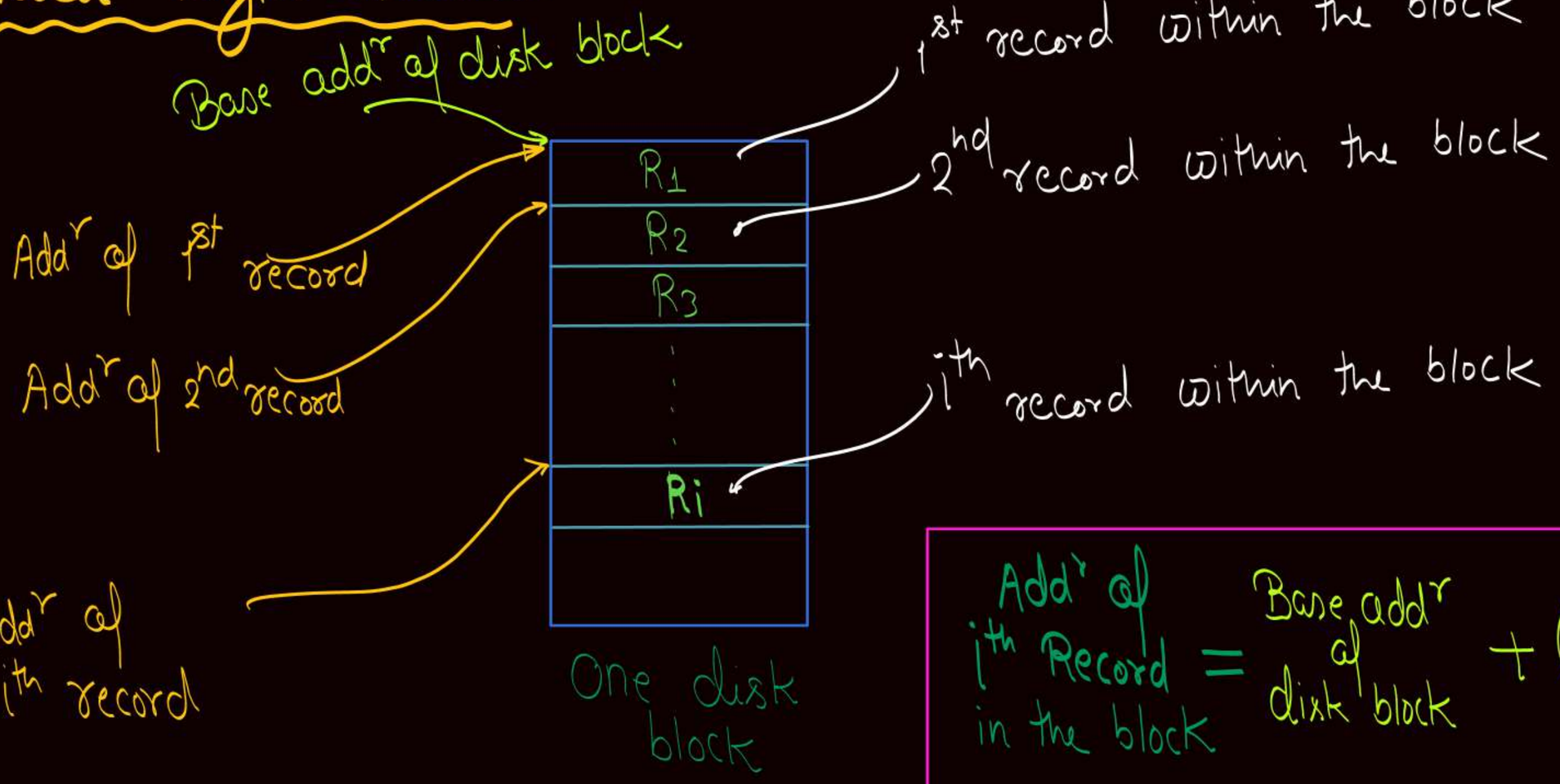
Topic : Types of Records

- There are two types of records
  - (1) Fixed length records
  - (2) Variable length records

Topic : Fixed length records

→ When each record of the file is of same size.

Fixed length Records: –

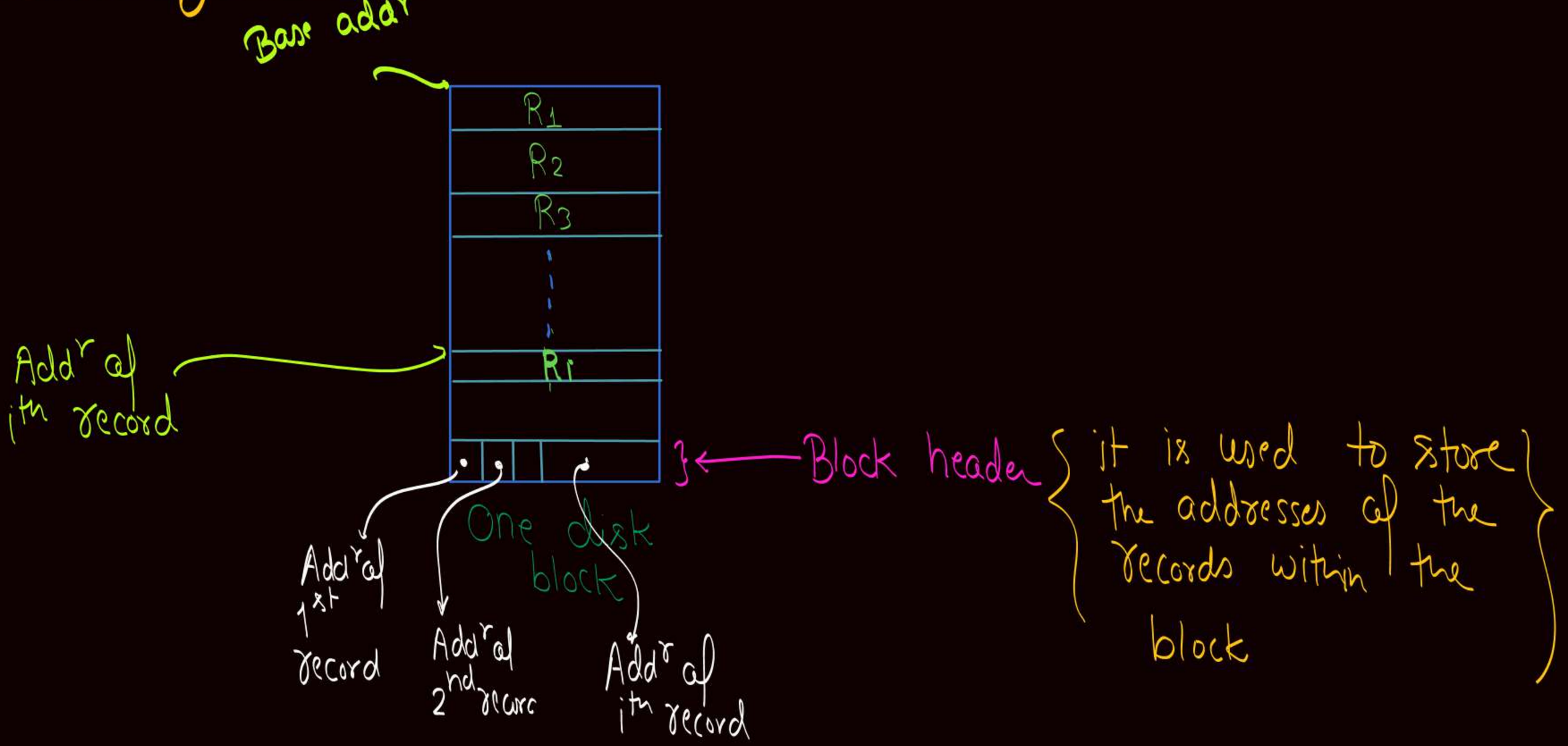


$$\text{Addr of } i^{th} \text{ Record in the block} = \text{Base addr of disk block} + (i-1) * \text{Size of each record}$$

Topic : Variable length records

→ When two records of the same file may be of different sizes, then records are called Variable length records.

Variable length Records: -



Note:- → Block header may be required with "fixed length records" as well in order to store the addr of the next disk block used to store the consequtive records of the file & it will be required if "link allocation" is used to allocate the blocks of the disk}

The pointer used to store address of the next disk block of the file is called "Block Anchor".

Note :- ① If nothing is given in the question about Block header size, then it is considered "Zero". i.e, the Complete disk block size will be used to store the records of the file

② If Block header size is given in the question, then Effective space available to store the records of the file will be :- (Disk block size − Block header size)

Topic : Blocking Factor (Bf)

Blocking factor (Bf) of a disk block w.r.t. database file is defined as average number of records stored per block of the disk.

$$\text{Blocking factor } (B_f) = \frac{\text{Disk block size}}{\text{Avg record size}}$$ { When block header size is not given }

$$\text{Blocking factor } (B_f) = \frac{\text{Disk block size} - \text{Block header size}}{\text{Avg record size}}$$ { When block header size is given }

Topic : Organization of Records

There are two ways in which records of the file can be organized within the disk block

- (1) Unspanned organization
- (2) Spanned organization

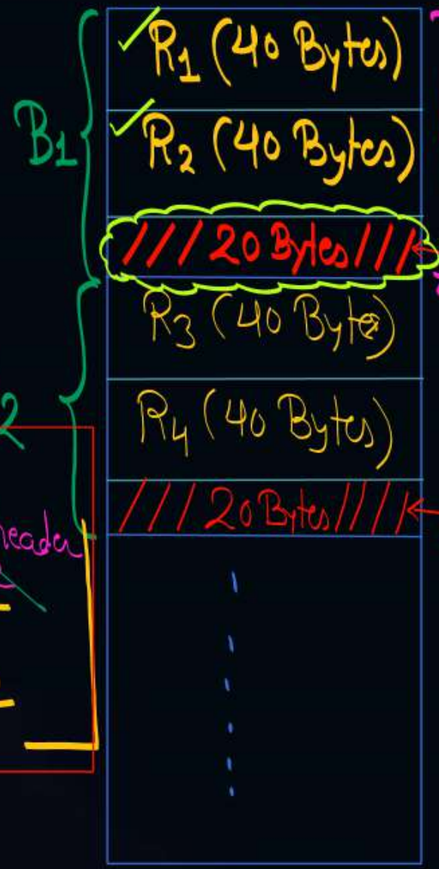
Topic : Un-spanned Organization

When a single record of the file is not allowed to span in more than one block

i.e, Complete record must be stored within the same block

Consider,

- Disk block size = 100 Bytes
- Record size = 40 Bytes



Available space is not sufficient to store the record completely so this space is not used

Hence Unspanned organization can result in internal fragmentation

$$\text{Blocking factor}(B_f) \text{ w.r.t. Unspanned-organization} = \left\lfloor \frac{\text{Block size} - \text{Block header size}}{\text{Record size}} \right\rfloor$$

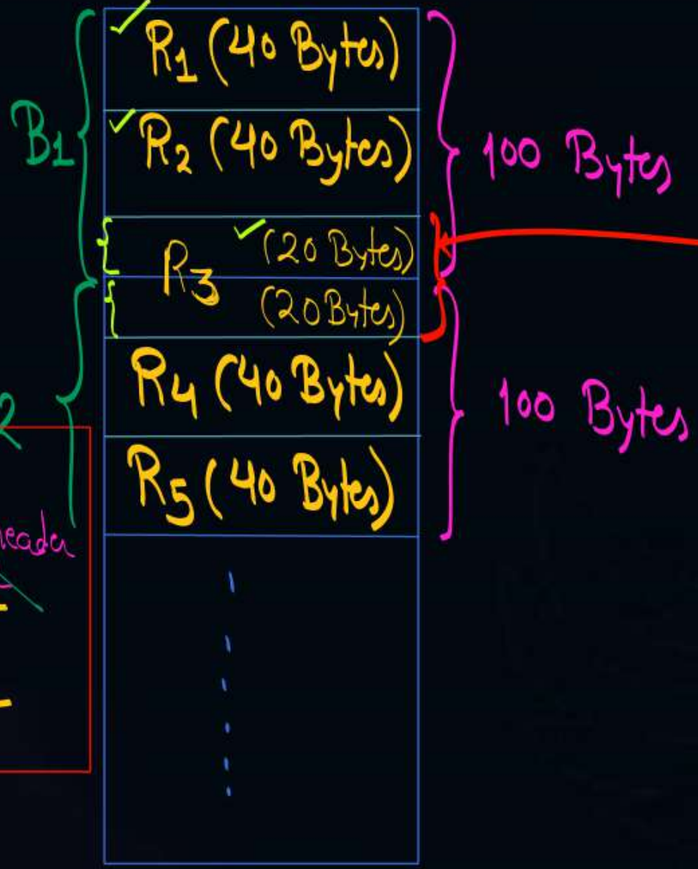
$$\text{Blocking factor } (B_f) = \left\lfloor \frac{100 - 0}{40} \right\rfloor = 2$$

Topic : spanned Organization

{ A single record may be allowed to span in more than one block }

Consider,

- Disk block size = 100 Bytes
- Record size = 40 Bytes



Using spanned org^{n} there is no internal fragmentation { Except for the last block of the file }

Blocking factor (B_f) w.r.t. Spanned organization $= \dfrac{\text{Block size} - \text{Block header size}}{\text{Record size}}$

$$\text{Blocking factor } (B_f) = \frac{100 - 0}{40} = 2.5$$

Topic : IO cost

- IO Cost of a record access can be defined as number of disk blocks that needs to be transferred from secondary memory to main memory in order to access that record

Topic : Search-key & Type of File

Search key :- Attribute / field used to search for a record in a database file is called search key

Retrieve Record of the Employee with Eid = E_6

Search for record will be performed based on "Eid"

∴ Eid Will be the Search Key

Employee database file

Eid	--other attributes--::	Dept_id	
E_3		2	B_1
E_1		1	
E_4		2	B_2
E_6		3	
E_2		4	B_3
E_5		1	
⋮	⋮	⋮	

Retrieve records of all the employees w.r.t. dept_id = 2

Search will be Performed based on "dept_id"

∴ Dept_id will be the Search Key

Topic : Search-key & Type of File

Need not have unique value

Search key :- Attribute / field used to search for a record in a database file is called search key

Types of file :- There are two types of files

(1) **Unordered file** :- If records of the file are not physically ordered (not sorted) based on search key, then file is Unordered file

(2) **ordered file** :- If records of the file are physically ordere (sorted) based on search key, then file is ordered file

Retrieve Record of the Employee with Eid = E6

↓

Search for record Will be performed based on "Eid"

∴ Eid Will be the Search Key

⇓

Records of the file are not Ordered based on "Eid"

∴ Unordered file

Employee database file

Eid	- -other attributes- - -	Dept_id	
E3		2	B_1
E1		1	
E4		2	B_2
E6		3	
E2		4	B_3
E5		1	
⋮	⋮	⋮	

Retrieve records of all the employees w.r.t. dept_id = 2

↓

Search Will be Performed based on "dept_id"

∴ Dept_id Will be the Search Key

⇓

Records of the file are Not ordered based on "Dept_id"

∴ Unordered file

Retrieve Record of the Employee of Eid = E6

⇓

Search will be performed based on Eid,
∴ Eid is the search key

⇓

Record of the file are Physically ordered based on search key (i.e. Eid)
∴ file is ordered file w.r.t. given situation

Employee database file

Eid	--other attributes---	Dept_id	
E1		1	B_1
E2		1	
E3		2	B_2
E4		2	
E5		3	B_3
E6		3	
⋮	⋮	⋮	

Records are still Unordered w.r.t. "dept_id"

⇓

∴ If search key is the dept_id, then file will be Unordered file

Topic : IO cost without indexing

* Let "N" is the total number of disk blocks required to store the given database file,

(i) Worst case IO Cost = N {Search will be linear Search}
(When file is un-ordered)

(ii) Worst Case IO Cost = $\lceil \log_2 N \rceil$ {If file is ordered then Binary Search is possible}
(When file is ordered)

* Note:- If database is too large, then "N" will also be a large value and in that case $\lceil \log_2 N \rceil$ will also be a significant value
∴ We must do something to reduce the IO Cost (Hence Concept of indexing will be used)

Ordered file

① Binary search is possible

② Insertion of a new record in an ordered file will be a costly operation. (because record must be stored at appropriate position in file)

Vs

Unorder file
{Also known as Heap file}

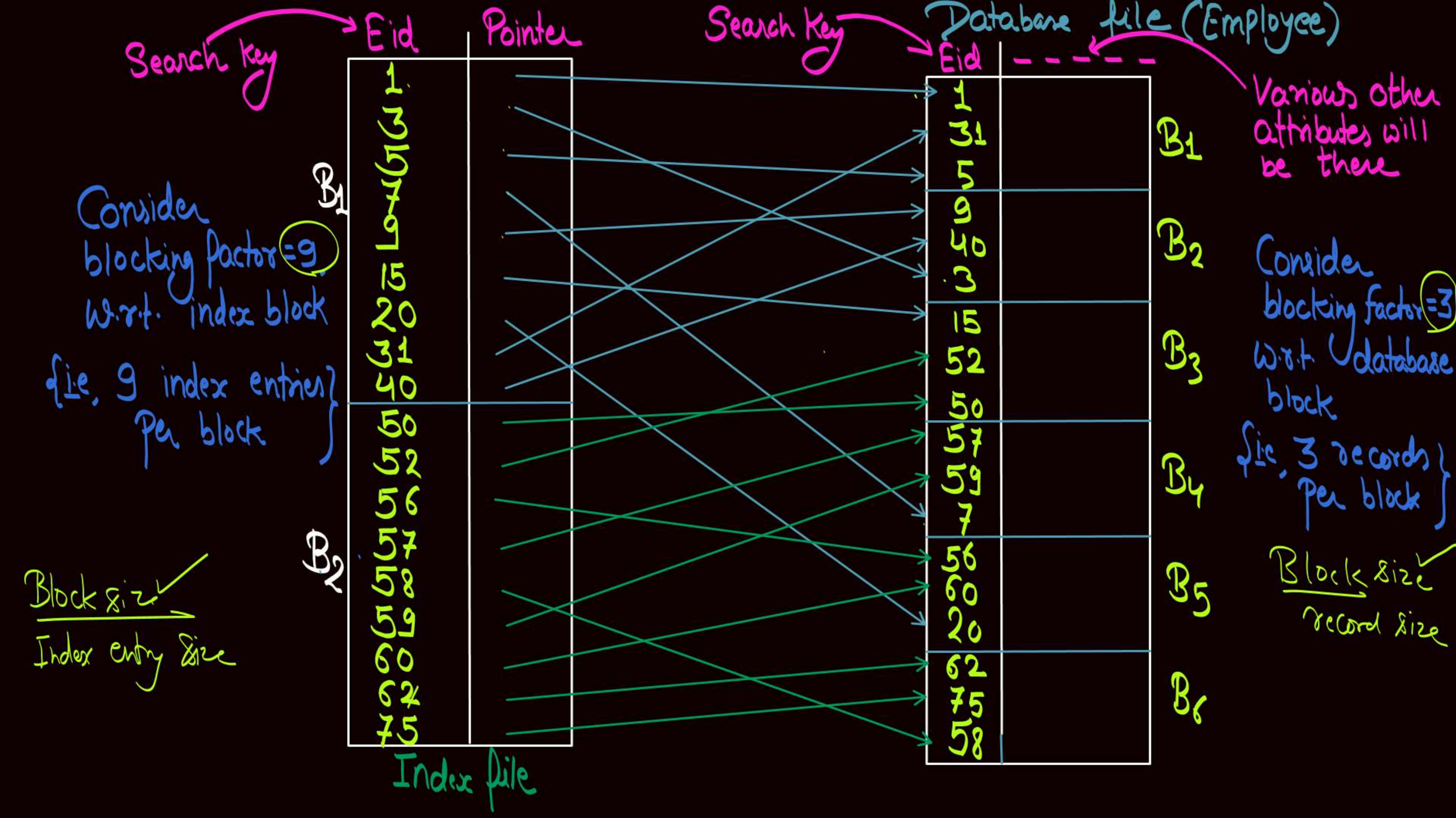
① Binary search is not possible

② Insertion of a new record will be easy {we can simply insert the record in the last block of the file}

Topic : Index file

* Index files are 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.


Search key
Eid
Pointer
Search key
Database file (Employee)
Eid
Various other attributes will be there
B1
B2
B3
B4
B5
B6
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}
Block size / Index entry size
Block size / record size
Index file

Topic : Index file

Index files are 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.

Index file Entry = < Search key value, Pointer information >

- In general, Size of index file entry < Size of each record in DB file

  (Contains only two fields) (Contains multiple fields in each record)

- Index file is also stored in the blocks of the same disk.

  i.e. Size of disk blocks used to store index file = Size of disk blocks used to store the records of the database file

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 index block) = $\frac{\text{Disk block size}}{\text{Index file entry size}}$ ↓

⑥ Blocking factor (w.r.t. database block) = $\frac{\text{Disk block size}}{\text{Record size}}$ ↑

⑦ In general, Index file Entry size < database file record size

i.e By ④, ⑤, ⑥, ⑦

⑧ → Blocking factor (w.r.t index block) > Blocking factor (w.r.t. database block)

⑨ No. of disk blocks required to store index file

$$= \frac{\text{Total no. of entries in the index file}}{\text{Number of index entries per block}}$$

$$= \frac{\text{Total no. of entries in the index file}}{\downarrow \text{Blocking factor of index block}}$$

---

⑩ No. of disk blocks required to store database file

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

$$= \frac{\text{Total no. of records in the database file}}{\downarrow \text{Blocking factor of database block}}$$

(11) In general,

No. of entries in index file $\leq$ No. of records in database file

$<$ — w.r.t. Spare index

$=$ — w.r.t. dense index.

(12) By (8), (9), (10) & (11)

Number of disk blocks required to store index file $<<$ Number of disk blocks required to store database file

Note:- In general, Index file is ordered based on search key

→ ∴ 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.

then in general, $M << N$

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

(to search for an entry in the index file w.r.t. given search key)

To access the block of database file, that contains the record corresponding to the search key value

Topic : Categories of Index

There are two categories of the indexes

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; Number of entries in index file w.r.t. dense index = No. of records in the database file

Topic : Categories of Index

There are two categories of the indexes

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

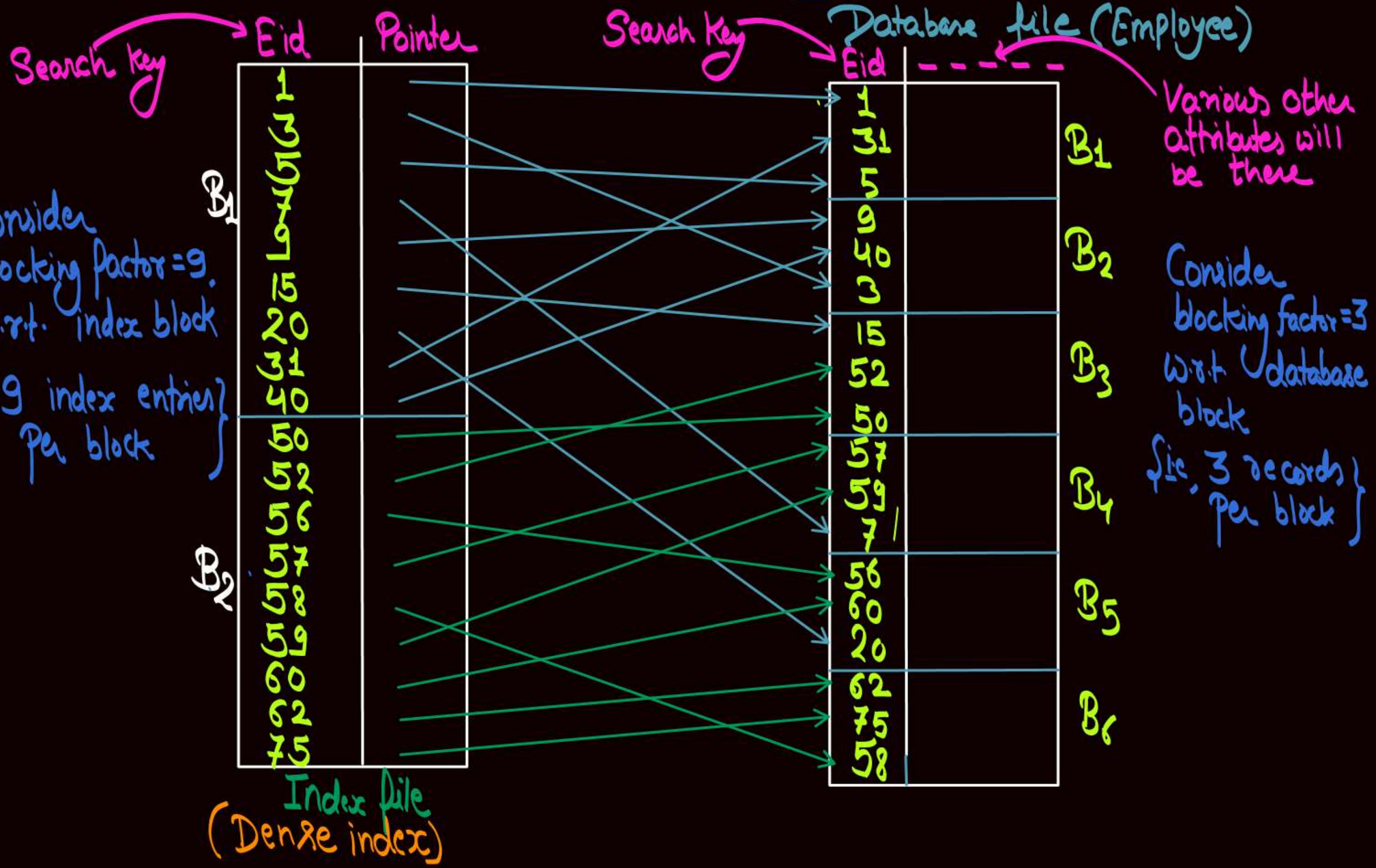
i.e; Number of entries in index file w.r.t. Sparse index < No. of records in the database file

* If index is sparse index, and nothing else is specified in the question about the no. of entries in sparse index, then in general 1 index file entry per block of database file

i.e, No. of entries in Sparse index file = No. of disk blocks required to store database file

* Example of Dense index file:-

In general, dense index is created for unordered file



- We can create dense index for ordered file aswell, but generally it is created for unordered file

* Example of Sparse index file :-

Sparse index can be created only for ordered files (Not possible for unordered 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 }


Thank
THANK
Keep Hustling!