COMPUTER SCIENCE

Database Management System

File Org. & Indexing

Lecture_1

Vijay Agarwal sir

TOPICS TO BE COVERED

01 File Structure

02 Indexing & its Type

1. FD & Normalization.
2. Transaction & Concurrency Control
3. ER MODEL & foreign key Concept
4. Query Language (RA, SQL, TRC)
5. File Org & Indexing. → Storage

File organization & Indexing.

Database

DB Files

Records.

Fields

Blocks

Q WHAT is **Block** ?

Disk :

Platter

↳ Surface

↳ Track

↳ Sector

⇓

'DATA'

File Structures & Indexing

Database

DB Files

Blocks

Records

Fields

- Database is collection of files.
- Each file is a collection of Records.
- Each record is a sequence of fields.

- DB is divided into number of blocks.
- Each block is divided into records.
- Record can be stored in a blocks.

Blocking factor [B_F] : Average Number of Records Per Block.

Blocking factor = 4

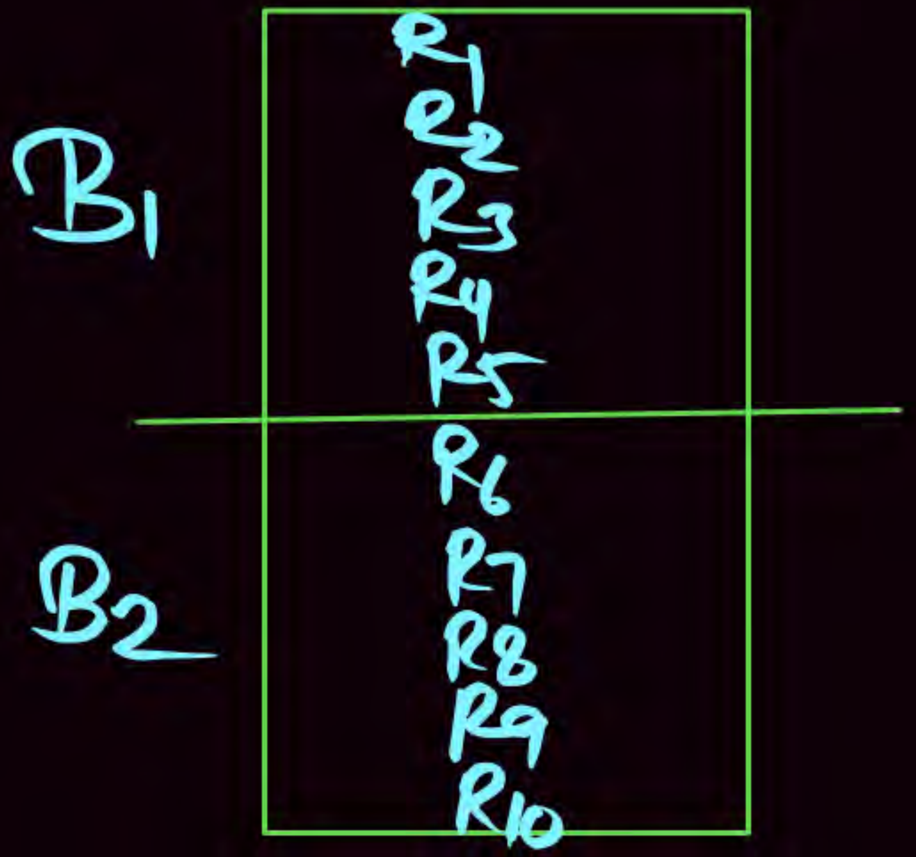
↳ 4 Records Per Block

$$\text{Block | Blocking factor } (B_F) = \frac{\text{Block Size}}{\text{Record size}}$$

Block Size = 1000 Byte

Record Size = 200 Byte

$$\text{Block factor} = \frac{1000B}{200B} = 5 \text{ Records Pr Blocks.}$$



Block Size >>>> Record Size

Floor & Ceiling Concept

Floor

$\lfloor 3.0 \rfloor = 3$

$\lfloor 2.0 \rfloor = 2$

$\lfloor 2.9 \rfloor = 2$

$\lfloor 2.8 \rfloor = 2$

$\lfloor 2.7 \rfloor = 2$

$\lfloor 2.6 \rfloor = 2$

$\lfloor 2.5 \rfloor = 2$

$\lfloor 2.4 \rfloor = 2$

$\lfloor 2.3 \rfloor = 2$

$\lfloor 2.2 \rfloor = 2$

$\lfloor 2.1 \rfloor = 2$

$\lfloor 2.0 \rfloor = 2$

Ceiling

$\lceil 2.1 \rceil = 3$

$\lceil 2.2 \rceil = 3$

$\lceil 2.3 \rceil = 3$

$\lceil 2.4 \rceil = 3$

$\lceil 2.5 \rceil = 3$

$\lceil 2.6 \rceil = 3$

$\lceil 2.7 \rceil = 3$

$\lceil 2.8 \rceil = 3$

$\lceil 2.9 \rceil = 3$

$\lceil 2.0 \rceil = 2$

$\lceil 3.0 \rceil = 3$

① SPANNED ORG.

A Record Can be Stored More than One Block.
(Partially Can be Stored)

② UNSPANNED Strategy org.

A Record Can be Stored/belongs to a Particular Block.
[Can not Stored Partially]

① SPANNED ORG.

② UNSPANNED Strategy org.

Record Blocking and Spanned Versus Unspanned Records

- Blocking factor
  - Average number of records per block for the file

(1) Spanned organization : A Particular Record Can be Stored More than One Blocks.

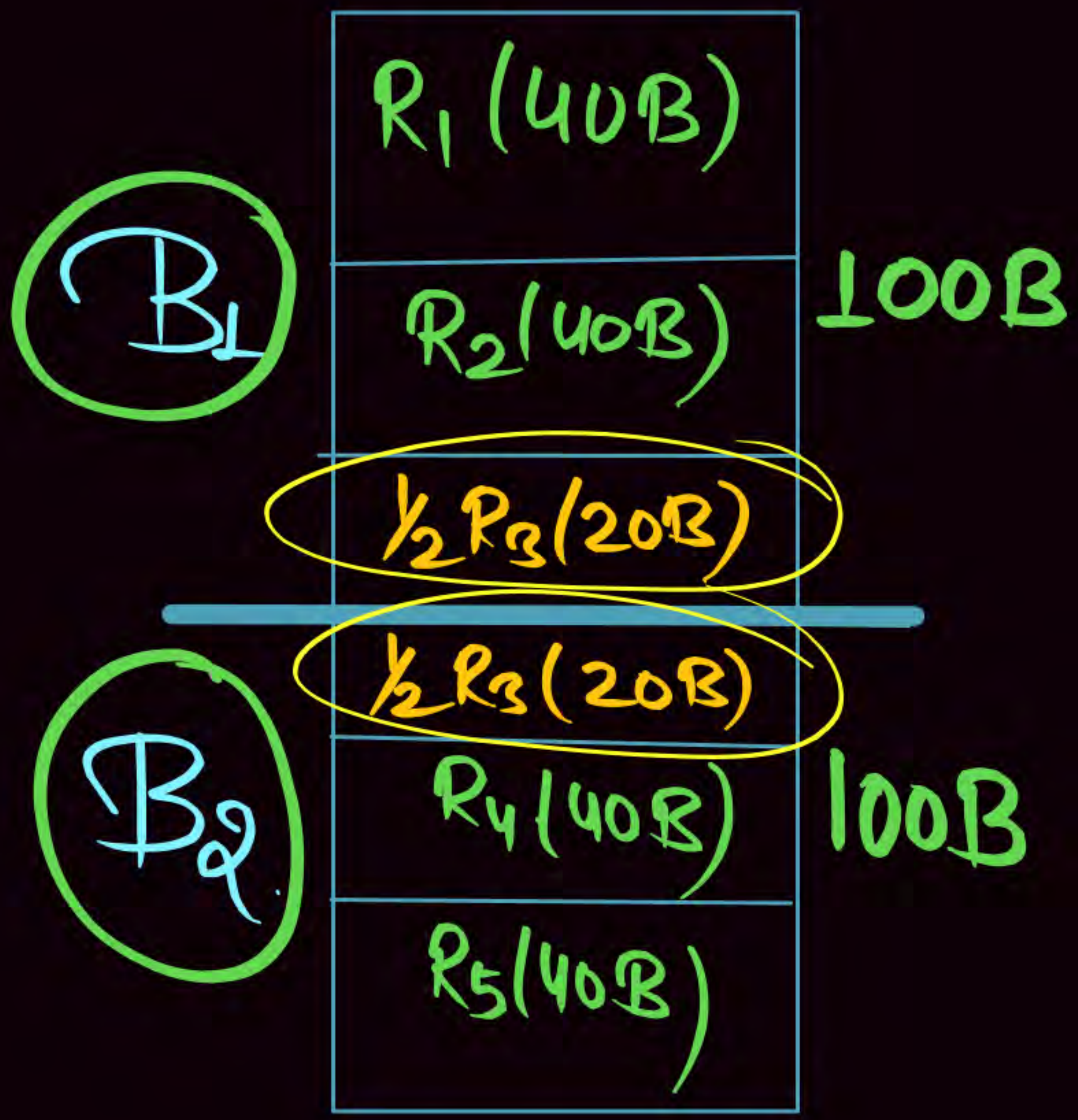
It allows Partially Part of Record Stored in a between the Block.

eg Block Size = 100 B
Record Size = 40 B

$$\text{Block factor} = \frac{\text{Block Size}}{\text{Record Size}}$$

$$\Rightarrow \frac{100B}{40B}$$

Block factor = 2.5 Records per Block.



SPANNED org.

Advantage: No Memory Wastage.

DisAdvantage: Block Access Increase.

Suitable: For. Variable Length Record.

Record Blocking and Spanned Versus Unspanned Records

- Blocking factor
  - Average number of records per block for the file.

(2) Unspanned organization : A Record Can be Stored Not More than One Block (Stored in a paricular Block)

Record Not Stored Partially in a Block.

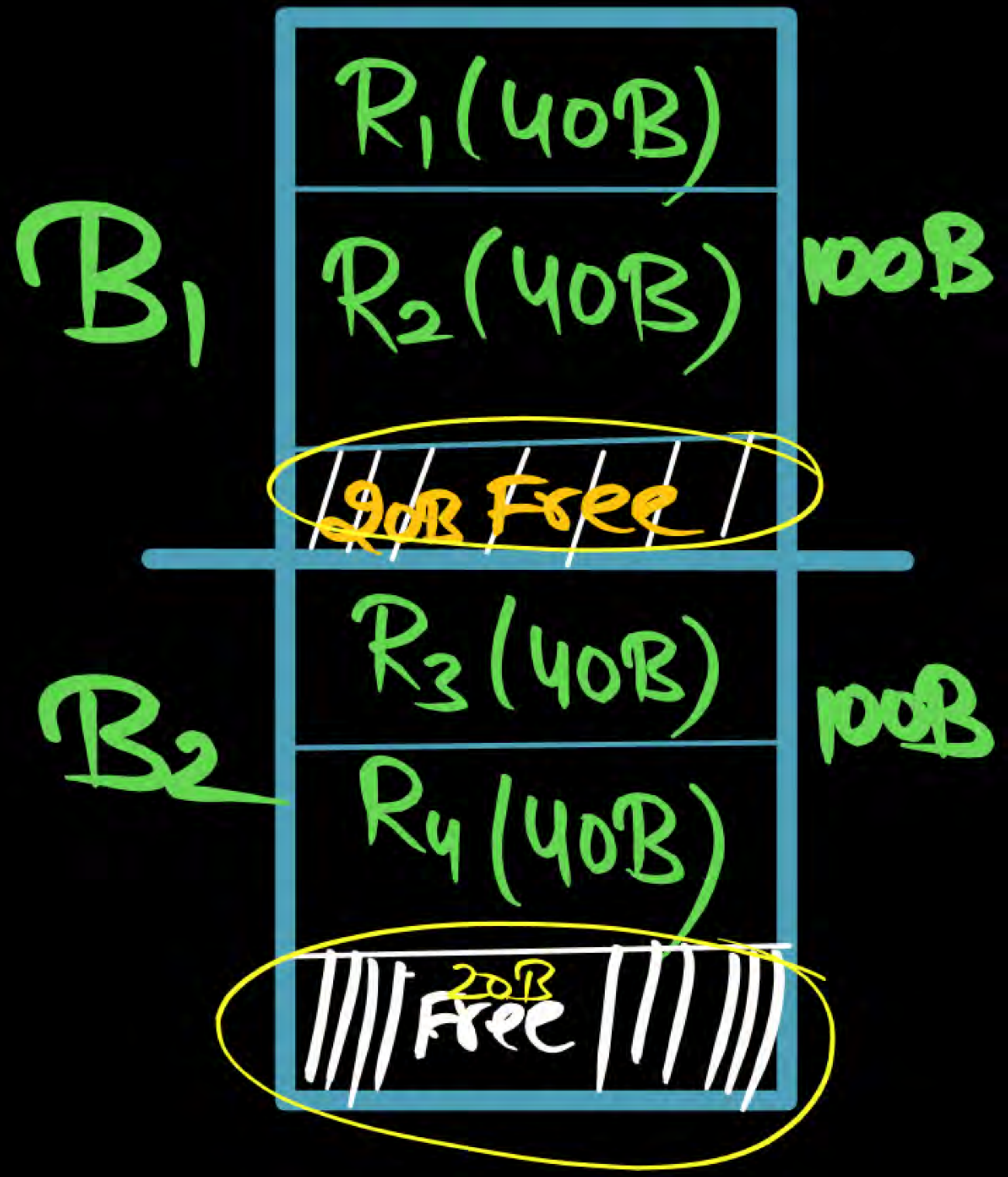
Block Size = 100B

Record Size = 40B

$$\text{Blocking factor} = \left\lfloor \frac{\text{Block Size}}{\text{Record Size}} \right\rfloor = \left\lfloor \frac{100B}{40B} \right\rfloor$$

$$= \lfloor 2.5 \rfloor$$

= 2 Records Per Block.



Unspanned Org

Advantage: Block Access Reduced.

Advantage: Wastage of Memory (Internal Fragmentation).

Suitable: For Fixed Length Record.

Note:

In spanned roganization No memory is waste but I/O cost is more (Block Access Increase).

But in unspanned organization memory is waste but input output cost is less compared to spanned organization.

Default organization is unspanned organization.

Note:

I/O Cost: Input Output cost means Number of Blocks transferred from secondary memory to Main memory in order to access some records.

Search Key: Attribute used to access the Data from DB

Organization of records in a file:
(1) ORDERED file organization
(2) Unordered file organization

1. Ordered File → ✓

2. Unordered File (Heap)

ORDERED File	Unordered (Heap) File
Searching Easy	Insertion Easy
Insertion Expensive (Reorg the file)	Searching Expensive
Binary Search	**Linear Search**
To Access a Record Avg No of Block Access = $\lceil \log_2 B \rceil$ (B: # DATA Block)	Average Case = $\frac{B}{2}$ Worst Case = B

Files of Ordered Records (Sorted Files)

- Ordered (sequential) file
  - Records sorted by ordering field
    - Called ordering key if ordering field is a key field
- Advantages
  - Reading records in order of ordering key value is extremely efficient
  - Finding next record
  - Binary search technique

NOTE:
To Access a record the average number of
Block Access = $\log_2 B$
(B: Data Blocks)

Files of Unordered Records (Heap Files)

- Heap (or pile) file
  - Records placed in file in order of insertion
- Inserting a new record is very efficient
- Searching for a record requires linear search
- Deletion techniques
  - Rewrite the block
  - Use deletion marker

NOTE:
To Access a record the average number of

Block Access = $\frac{B}{2}$

(B: Data Blocks)

Worst Case = B

Access Times for Various File Organizations

Type of Organization	Access/Search Method	Average Blocks to Access a Specific Record
Heap (unordered)	Sequential scan (linear search)	$b/2$
Ordered	Sequential scan	$b/2$
Ordered	Binary search	$\log_2 b$

Average access times for a file of b blocks under basic file organizations

Indexing :

Indexing are Used to Improve the Searching Efficiency.

→ Search in a faster Ways.

→ To Reduce the I/O Cost.

One Record of Index file Contain 2 fields.

Search key	Pointer (Block Pointer)

→ Denote to a Block where key is available.

Indexing (Basic Concepts)

- Indexing mechanisms used to speed up access to desired data.
  - E.g., author catalog in library
- **Search Key** - attribute to set of attributes used to look up records in a file.
- An **index file** consists of records (called **index entries**) of the form

search-key	pointer

- Index files are typically much smaller than the original file.
- Two basic kinds of indices:
  - **Ordered indices:** search keys are stored in sorted order
  - **Hash indices:** search keys are distributed uniformly across "buckets" using a "hash function".

Index file block size is same as DB file Block Size

$\boxed{\text{Block Size of Index File = Block Size of DB file}}$

$\boxed{\text{One Index Record Size = Size of Search Key + Size of Block Pointer}}$

NOTE: To Access a Record Average number of block access

$$= \underbrace{\log_2 B_i}_{\text{Index Block access}} + \underbrace{1}_{\text{Data Block access}}$$

Index Block access

Data Block access

[Bi : Index Block]

Example:

- Suppose that:
  - record size $\boxed{R = 150 \text{ bytes}}$ block size $B = 512$ bytes,

    $r = 30000$ records
- Then, we get:
  - blocking factor $Bfr = \left\lfloor \frac{\text{Block size}}{\text{Record Size}} \right\rfloor = \left\lfloor \frac{512B}{150B} \right\rfloor = 3$ Record Per Block.
  - number of file blocks $b =$

$$\frac{30{,}000}{3} = 10{,}000 \text{ Data Block}$$

$$\boxed{B = 10{,}000}$$

ORERDED file

To Access a Record Avg No. of Block Access $= \lceil \log_2 B \rceil$

$2^{10} = 1024$

$2^{11} = 2048$

$2^{12} = 4096$

$2^{13} = 8192$

$\lceil 13.x \rceil$ $2^{14} =$

$\Rightarrow \lceil \log_2 10000 \rceil$

$= 14$ Ans

Unoderd file

To Access a Record Avg Number of Block Access $= B/2$.

$\Rightarrow \frac{10000}{2} = 5000$ Ans

Worst $= 10{,}000$ Ans

Example:

- Suppose that:
  - record size R = 150 bytes, block size B = 512 bytes,

    r = 30000 records
- Then, we get:
  - blocking factor Bfr = $\boxed{\text{B div R} = \text{512 div 150} = \text{3 records/block}}$
  - number of file blocks b = $\boxed{(r/\text{Bfr}) = (30000/3) = 10000 \text{ blocks}}$

Example:

Given the following data file

EMPLOYEE (NAME, SSN, ADDRESS, JOB, SAL, ...)

Suppose that:

- record size $R=150$ bytes, block size $B=512$ bytes $r=30000$ records
- For an index on the SSN field, assume the field size $V_{SSN}=9$ bytes, assume the record pointer size $P_R=7$ bytes. Then:

Example:

Given the following data file

EMPLOYEE (NAME, SSN, ADDRESS, JOB, SAL, ...)

Suppose that:

- record size $R=150$ bytes, block size $B=512$ bytes $r=30000$ records
- For an index on the SSN field, assume the field size $V_{SSN}=9$ bytes, assume the record pointer size $P_R=7$ bytes. Then:
  - index entry size $R_1=(V_{SSN}+P_R)=$
  - index blocking factor $Bfr_1=B \text{ div } R_1=$ entries / block
  - number of index blocks $b=(r/Bfr_1)=$ = blocks
  - binary search needs $\log_2 bI=\log_2 938=$ block accesses

Example:

Given the following data file

EMPLOYEE (NAME, SSN, ADDRESS, JOB, SAL, ...)

Suppose that:

- record size $R=150$ bytes, block size $B=512$ bytes $r=30000$ records
- For an index on the SSN field, assume the field size $V_{SSN}=9$ bytes, assume the record pointer size $P_R=7$ bytes. Then:
  - index entry size $R_1=(V_{SSN}+P_R)=(9+7)=16$ bytes
  - index blocking factor $Bfr_1=B \text{ div } R_1=512 \text{ div} 16=32$ entries / block
  - number of index blocks $b=(r/Bfr_1)=(30000/32)=938$ blocks
  - binary search needs $\log_2 bI=\log_2 938=10$ block accesses

Category of Index

1) Dense Index Files

Number of Index entries = Number of DB Records

2) Sparse Index Files

Number of Index entries = Number of Blocks

Dense Index Files

- Dense Index - Index record appears for every search-key values in the file.
- Example - index on *ID* attribute of *instructor* relation

10101
12121
15151
22222
32343
33456
45565
58583
76543
76766
83821
98345

10101	Srinivasan	Comp. Sci.	65000	
12121	Wu	Finance	90000	
15151	Mozart	Music	40000	
22222	Einstein	Physics	95000	
32343	El Said	History	60000	
33456	Gold	Physics	87000	
45565	Katz	Comp. Sci.	75000	
58583	Califieri	History	62000	
76543	Singh	Finance	80000	
76766	Crick	Biology	72000	
83821	Brandt	Comp. Sci.	92000	
98345	Kim	Elec. Eng.	80000	

Sparse Index Files

- Sparse Index: contains index records for only some search-key values.
  - Applicable when records are sequentially ordered on search-key
- To locate a record with search-key value K we:
  - Find index record with largest search-key value $< K$
  - Search file sequentially starting at the record to which the index record points

10101	
32343	
76766	

10101	Srinivasan	Comp. Sci.	65000	
12121	Wu	Finance	90000	
15151	Mozart	Music	40000	
22222	Einstein	Physics	95000	
32343	El Said	History	60000	
33456	Gold	Physics	87000	
45565	Katz	Comp. Sci.	75000	
58583	Califieri	History	62000	
76543	Singh	Finance	80000	
76766	Crick	Biology	72000	
83821	Brandt	Comp. Sci.	92000	
98345	Kim	Elec. Eng.	80000	

Types of Index

Single-level Ordered Indexes

- Primary indexes
- Clustering indexes
- Secondary indexes

Multilevel Indexes

Dynamic multilevel indexes Using B-Tress and B^+ Trees.

Q.1

Assume a relational database system that holds relation: C(colleges) with the following characteristics

- Records are stored as fixed length, fixed format records, length is 256 bytes.
- There are 16384 records.
- Records contains key attribute CollegeNumber (C.N), length 22 bytes and other fields.
- Unspanned organization is used to store the information or record.

Let's suppose we want to build a sparse primary index on C.N then how many numbers of 4096-byte blocks are needed to store the primary index when block pointer size is 10 bytes _____?

A. 7

B. 8

C. 9

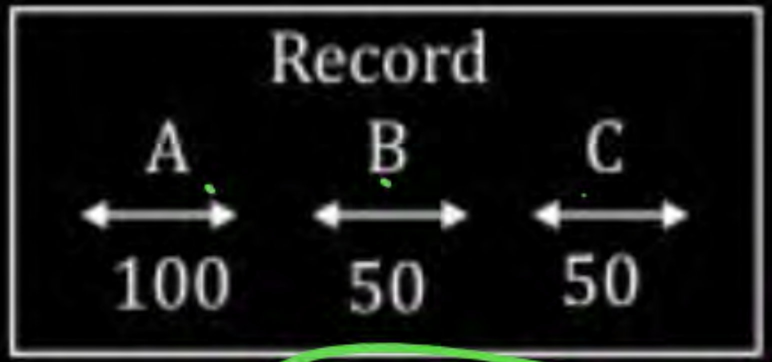
D. 10

File Organization:

- Data Records can be :

  1) Fixed Length Records

  2) Variable Length Records



200 bytes = for each Records

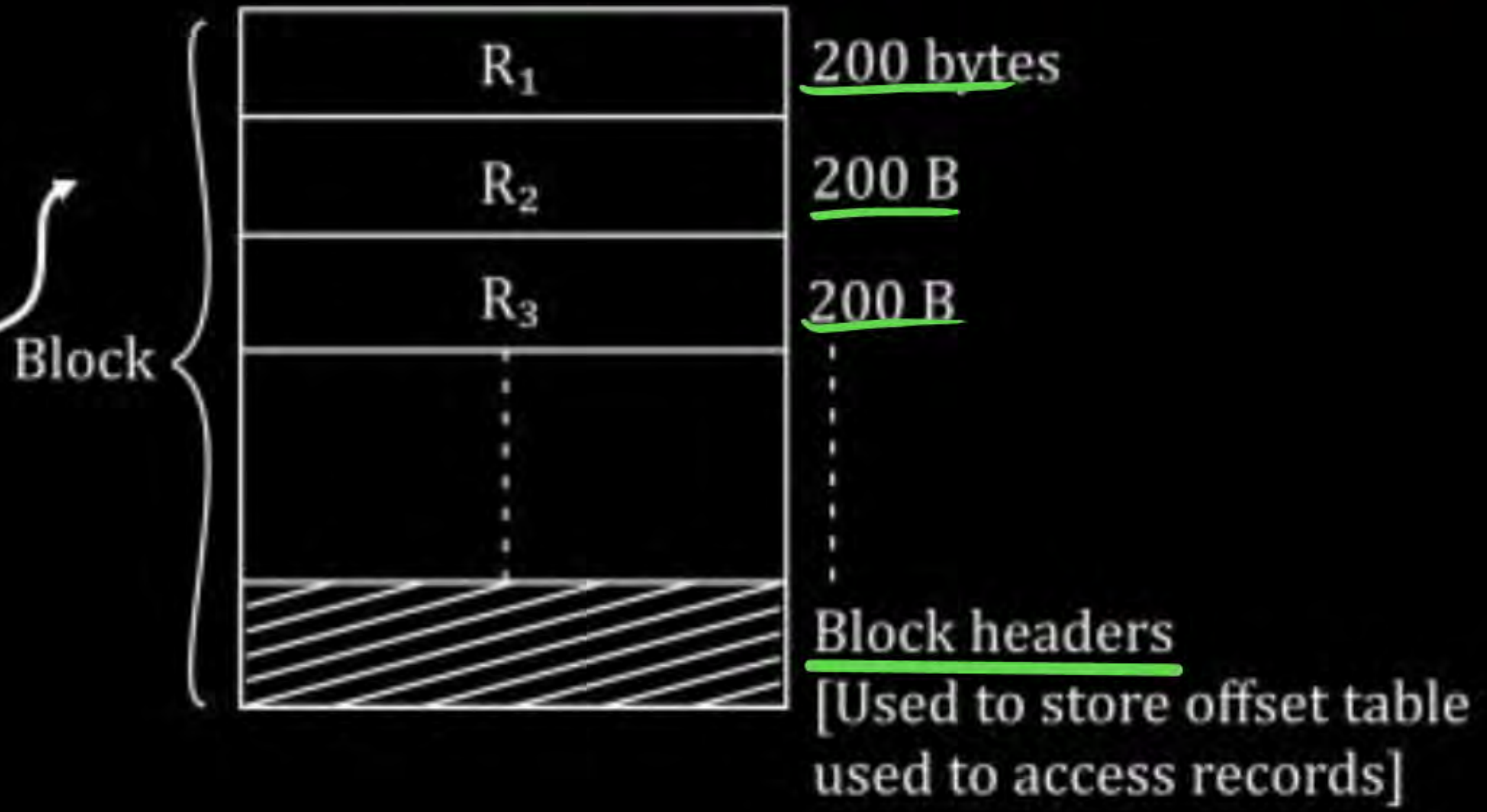
1) Fixed Length Records

Create table R

(A char (100),

B char (50),

C char (50));



File Organization:

2) Variable Length Records

Create table S

(D char (100),

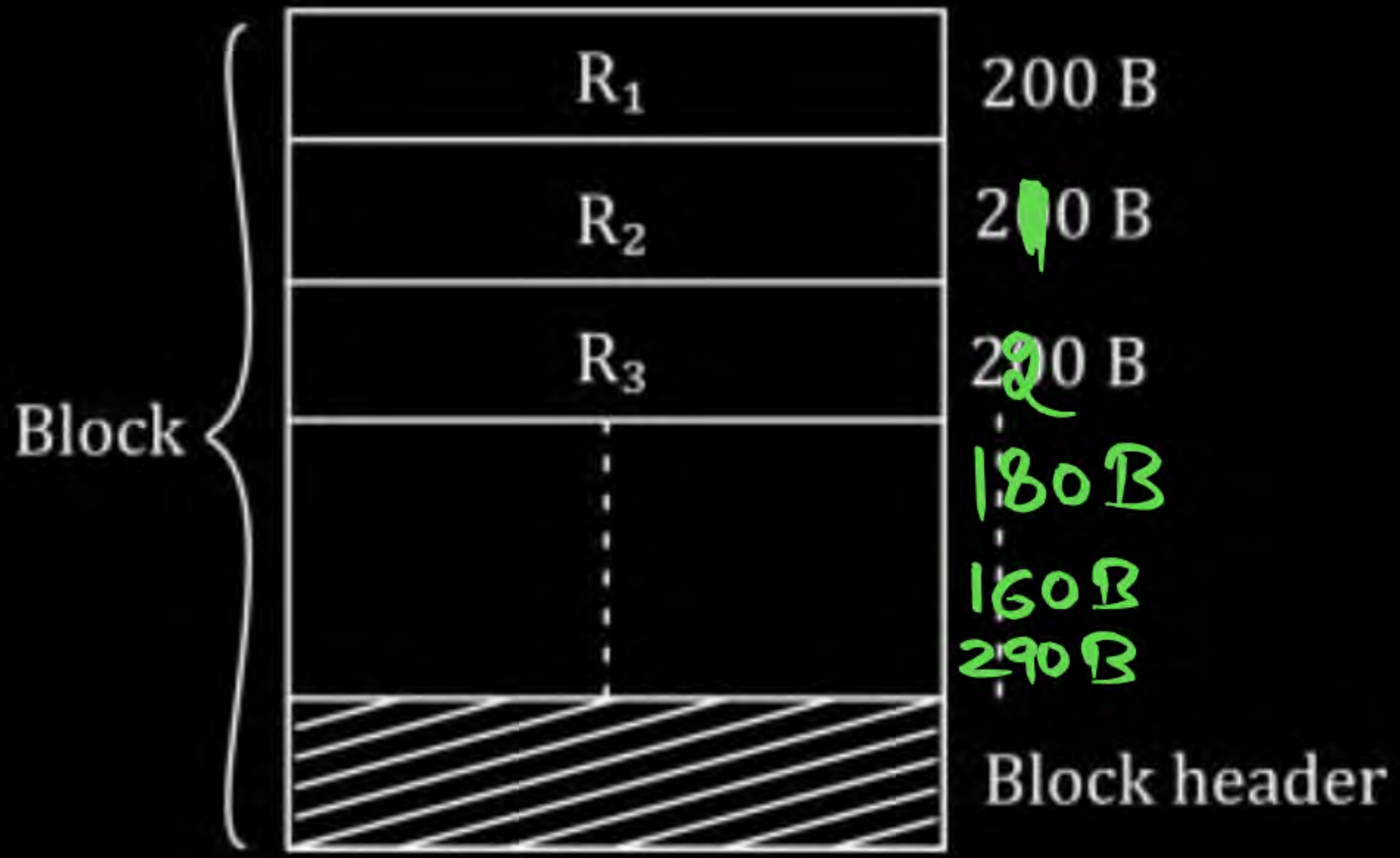
E char (50),

F text

);

$\therefore$ size (D) + Size (E) + size(F)

100   50   not fixed



File Organization:

- DB File with all records fixed length
- DB file with variable length records

$\Rightarrow$ both are possible in RDBMS

Record Organization:

(a) Spanned Organization

Record allow to span in more than one block

Example: Block size 1000B Record size 400B

Advantage:

⇒ Possible to allocate file with no internal fragmentation.

Record Organization:

$\frac{B-H}{R}$

R: Record
B: Block.

(a) Spanned Organization

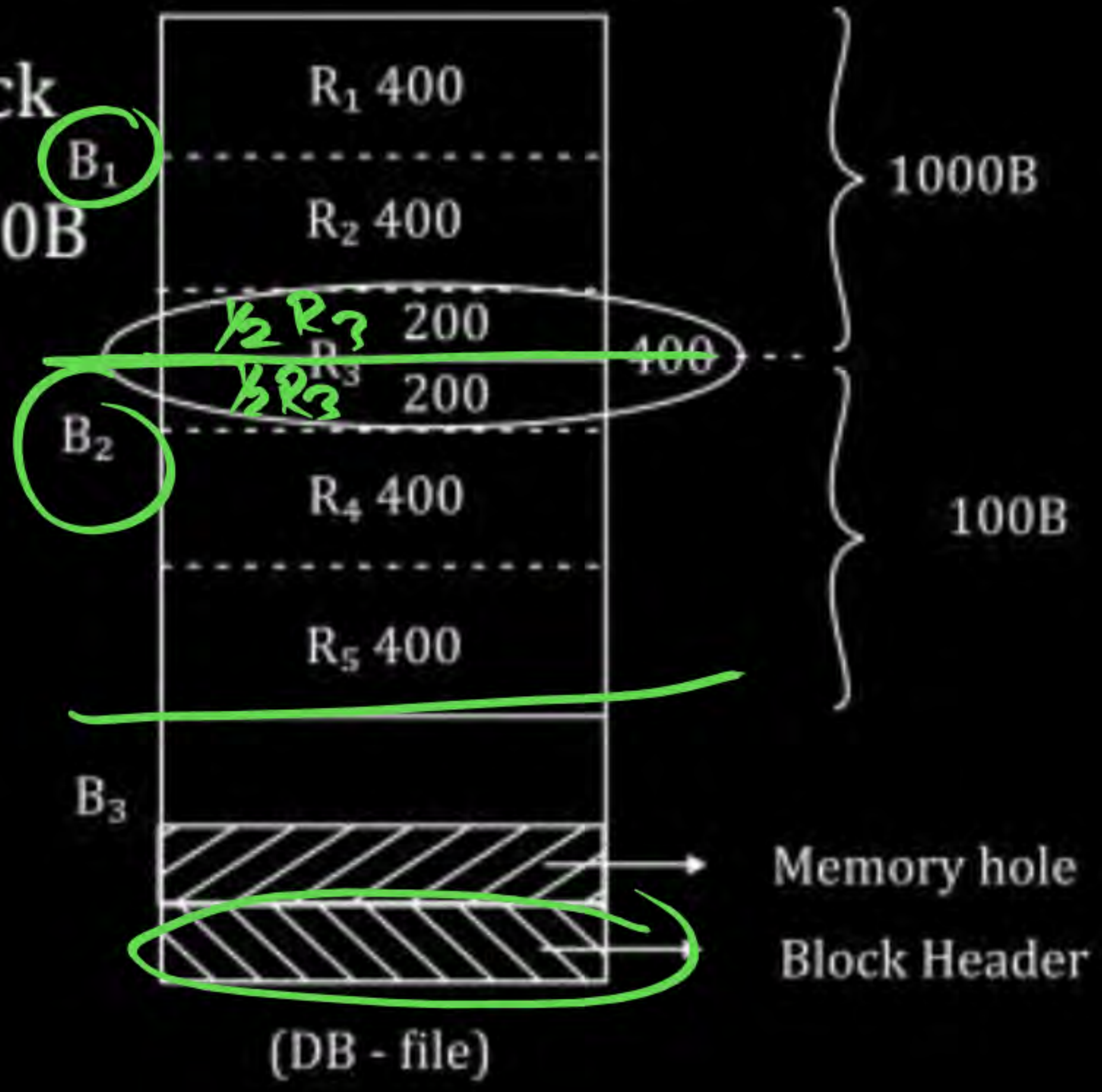
Record allow to span in more than one block

Example: Block size 1000B; Record size 400B

Block Factor $= \frac{1000-0}{400} = 2.5\ R/B$

⇒ Too complex to manage records

⇒ More access cost to access records.



Advantage:

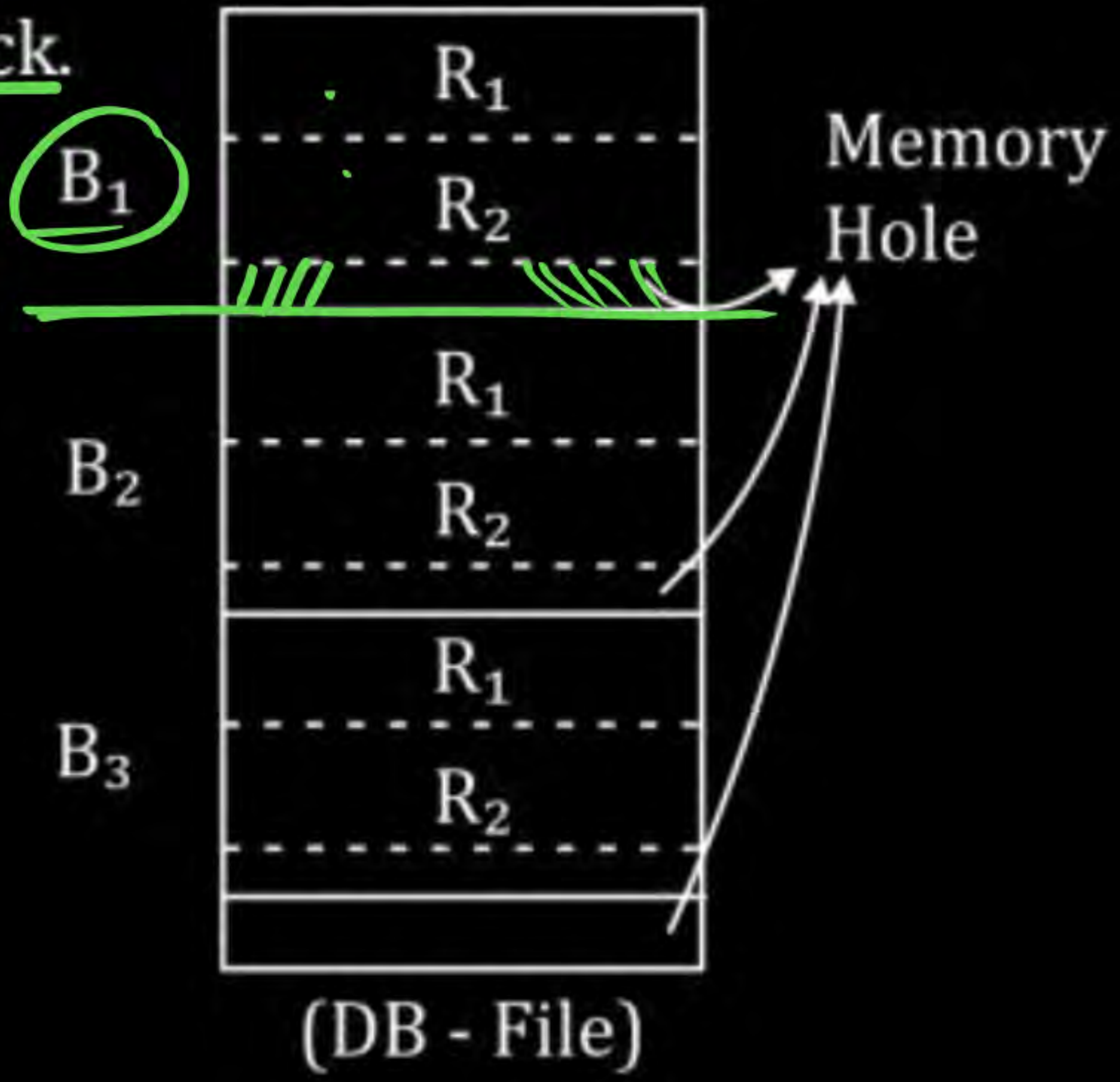
⇒ Possible to allocate file with no internal fragmentation

Record Organization:

(a) Unspanned Organization

Complete Record must be stored in one Block.

$$\text{Block Factor} = \left\lfloor \frac{1000-0}{400} \right\rfloor = 2\ R/B$$



- ⇒ Easy to Manage records
- ⇒ Access cost is also less.
- ⇒ May not possible to avoid internal fragmentation.

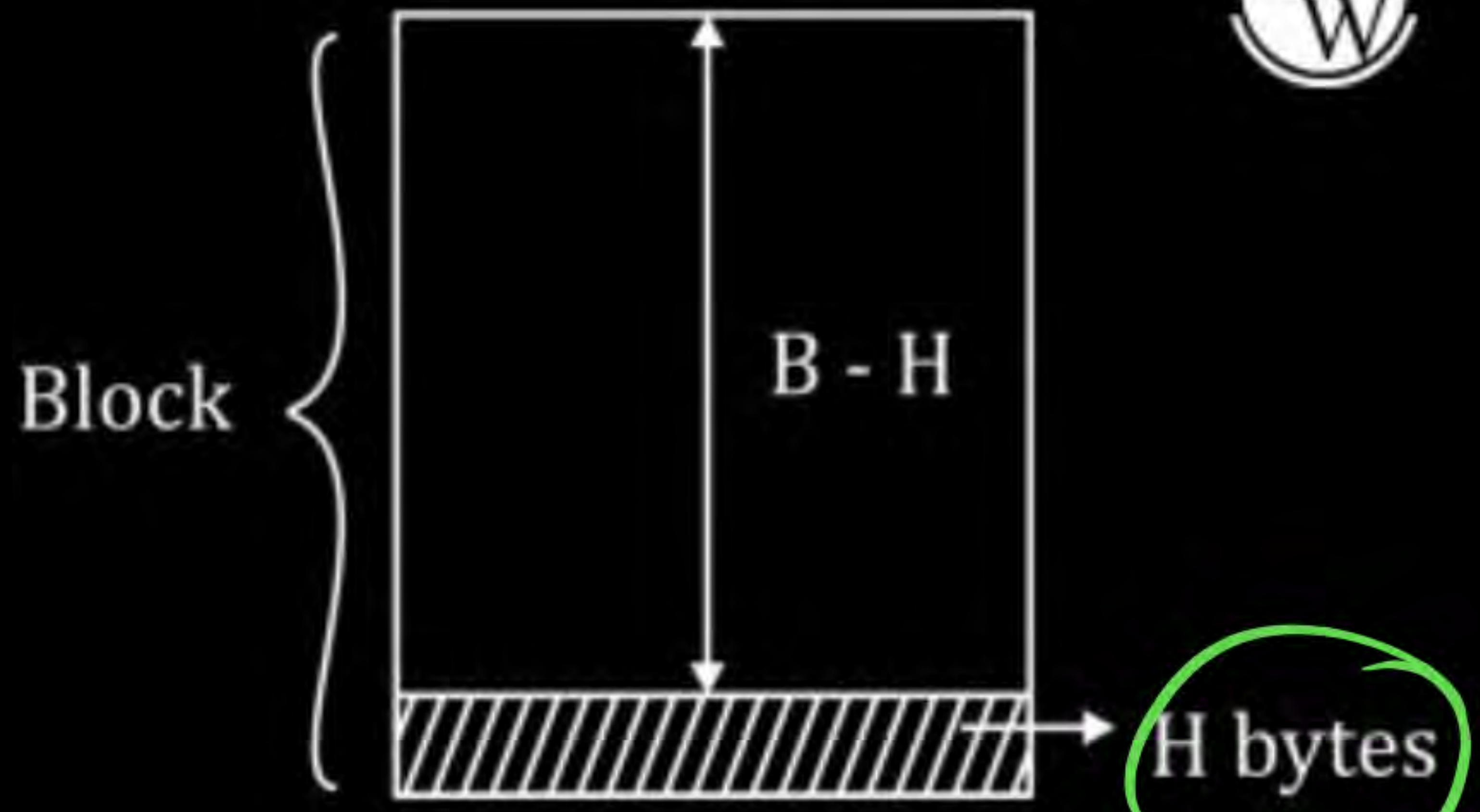
Block Factor

Maximum Possible Records per Block.

Block Size: B bytes

Block Header size: H bytes

Record size: R bytes



'0' Byte

- Block factor for unspanned: $\left\lfloor \frac{B-H}{R} \right\rfloor$ record / block

  because for fixed length record unspanned is preferred.

- Block Factor for spanned: $\frac{B-H}{R}$ record / block

  $\frac{B-H}{R}$

NOTE:

1) To organize DB file with fixed length record unspanned organization is preferred.

2) To organize DB file with variable length record spanned organization preferred.

Indexing (Basic Concepts)

- Indexing mechanisms used to speed up access to desired data.
  - E.g., author catalog in library
- **Search Key** - attribute to set of attributes used to look up records in a file.
- An **index file** consists of records (called **index entries**) of the form

search-key	pointer

- Index files are typically much smaller than the original file.
- Two basic kinds of indices:
  - **Ordered indices:** search keys are stored in sorted order
  - **Hash indices:** search keys are distributed uniformly across "buckets" using a "hash function".

Index file block size is same as DB file Block Size

Block Size of Index File = Block Size of DB file

One Index Record Size = Size of Search Key + Size of Block Pointer

NOTE: To Access a Record Average number of block access
$= \log_2 B_i + 1$

Index Block access

Data Block access

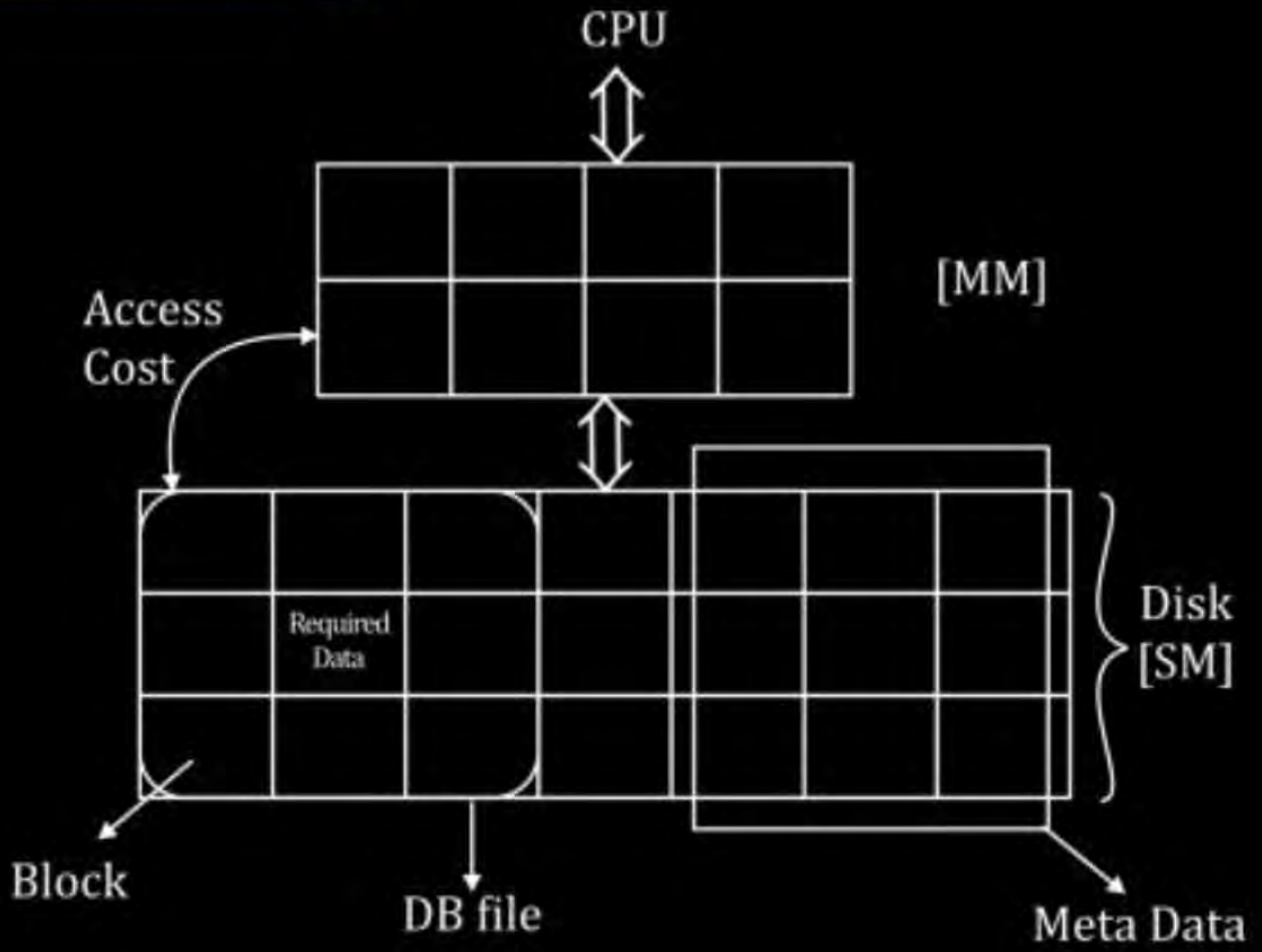
[Bi : Index Block]

Indexing

Used to reduce access cost or I/O cost.

Access Cost: Number of SM(secondary Memory) Blocks (Disk blocks) to transfer from SM to MM in order to access required data.

Indexing



CPU
[MM]
Access
Cost
Required
Data
Disk
[SM]
Block
DB file
Meta Data

Meta Data [Data Dictionary]

→ record format

→ Field format

→ Number of blocks allocated for file

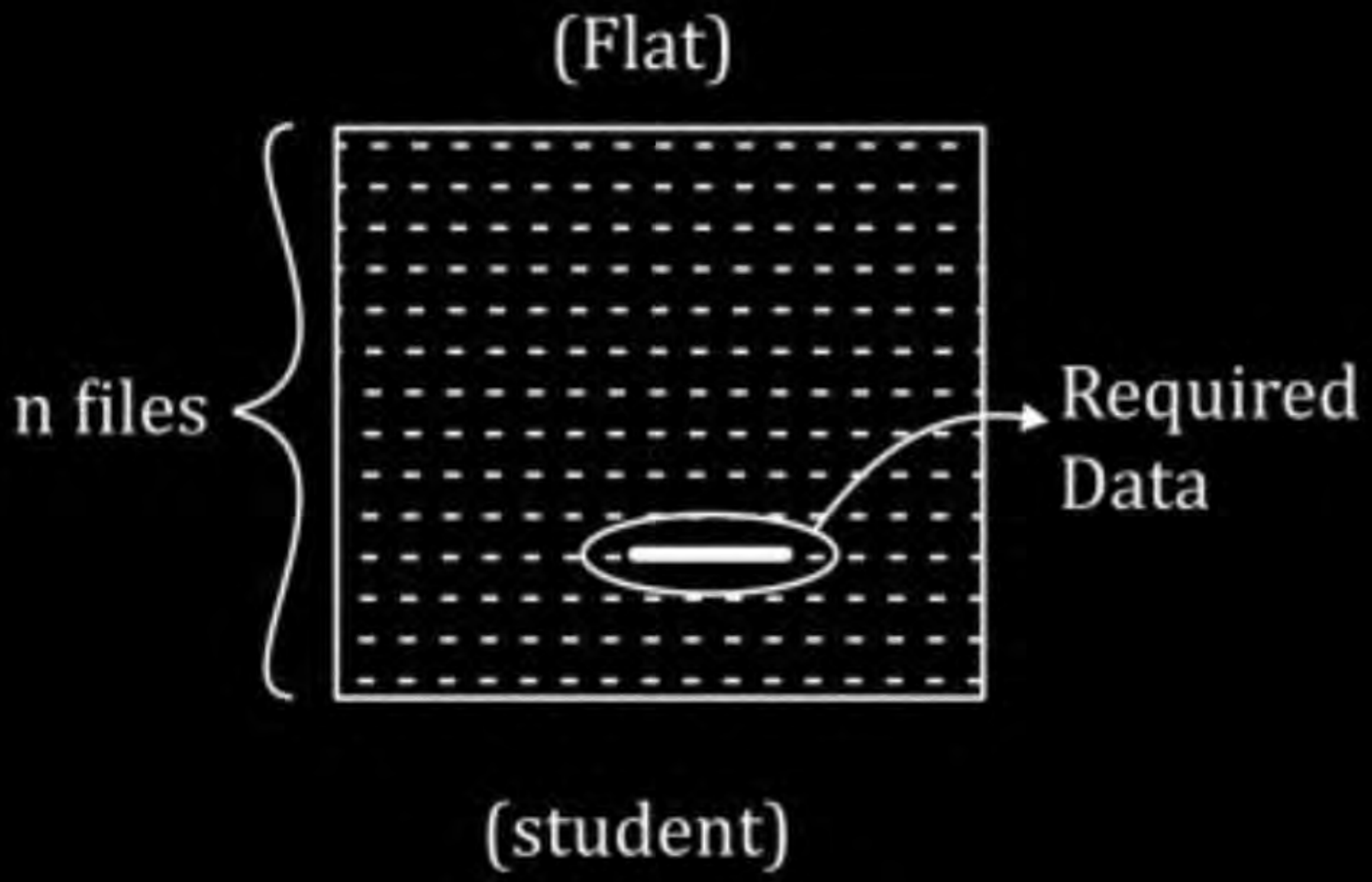
→ Number of record in file

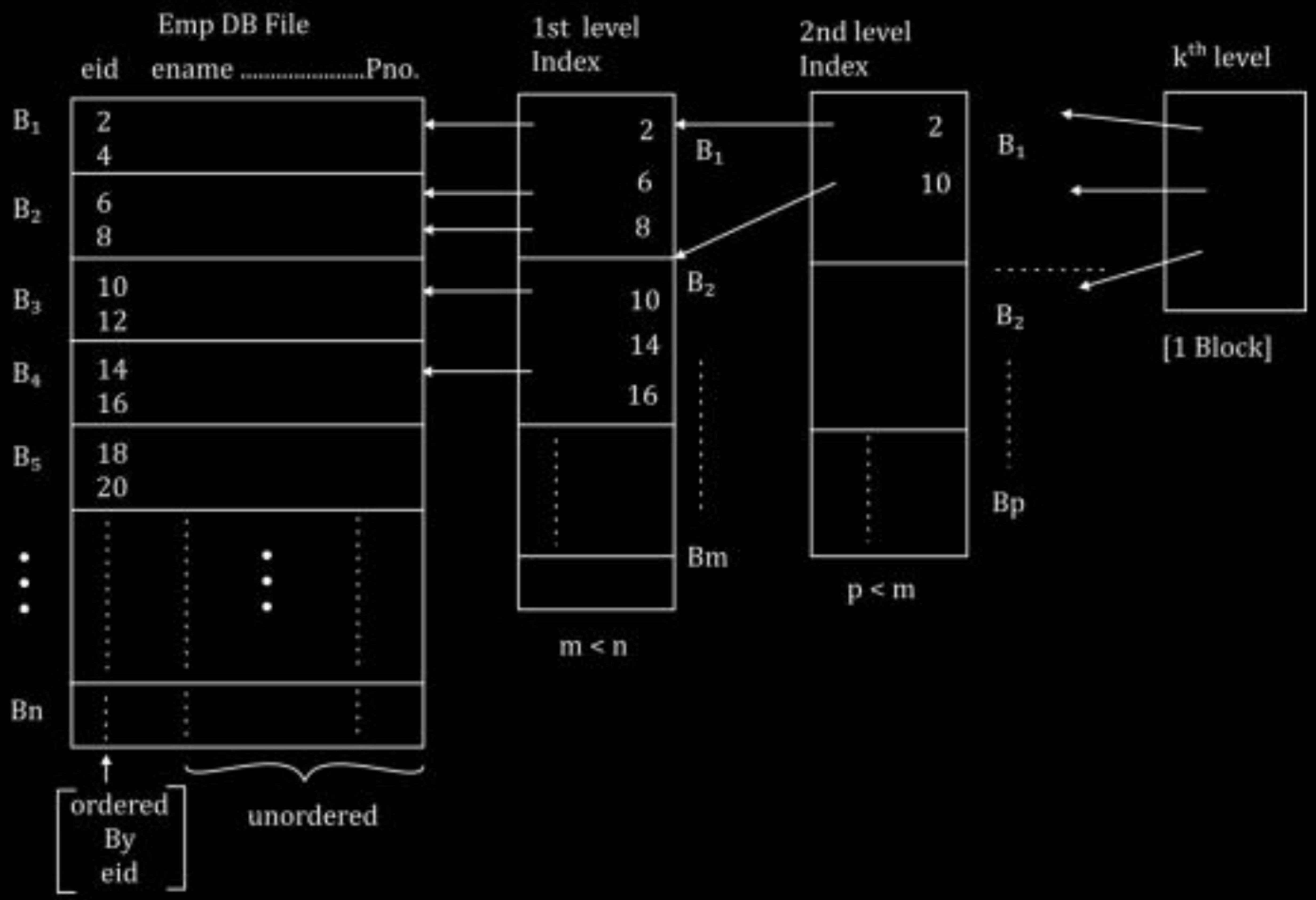
→ Ordered field of file, etc

Indexing

Huge access cost to access data from flat file system.

[complete file should transfer to mm to locate required data]





Emp DB File
eid ename ....Pno.
B1
2
4
B2
6
8
B3
10
12
B4
14
16
B5
18
20
Bn
ordered By eid
unordered
1st level Index
2
6
8
10
14
16
B1
B2
Bm
m < n
2nd level Index
2
10
B1
B2
Bp
p < m
k^th level
[1 Block]

Index File:

$$\boxed{\text{BF (Block Factor) of Index} = \left\lfloor \frac{B-H}{K+P} \right\rfloor \text{ entries / block}}$$

$$\text{Index File}\left\{\boxed{\begin{array}{cc} k_1 & p_1 \\ k_2 & p_2 \\ \vdots & \vdots \end{array}}\right. \quad \therefore \ \mathrm{BF} = \left\lfloor \frac{\mathrm{B-H}}{\mathrm{k-p}} \right\rfloor$$

$$\boxed{\left\lfloor \frac{B-H}{R} \right\rfloor \text{ record / block} < \left\lfloor \frac{B-H}{K+P} \right\rfloor \text{ entries / block}}$$

$$\left\{\begin{array}{l} \text{\# of DB} \\ \text{File block} \end{array}\right\} > \left\{\begin{array}{l} \text{\# of Index} \\ \text{blocks} \end{array}\right\}$$

Multilevel Index:

1) 1st level index is index to DB file and 2nd level onward Index to index file until 1 block index at last level.
2) Idle access cost to access record using multi level index is $(n + 1)$ blocks, n is number of level in index.

Categories of Index:

1) Dense Index [More entries in Index File]
2) Sparse Index [Less entries in Index File]

Category of Index

1) Dense Index Files

Number of Index entries = Number of DB Records

2) Sparse Index Files

Number of Index entries = Number of Blocks

Dense Index Files

- Dense Index - Index record appears for every search-key values in the file.
- Example - index on *ID* attribute of *instructor* relation

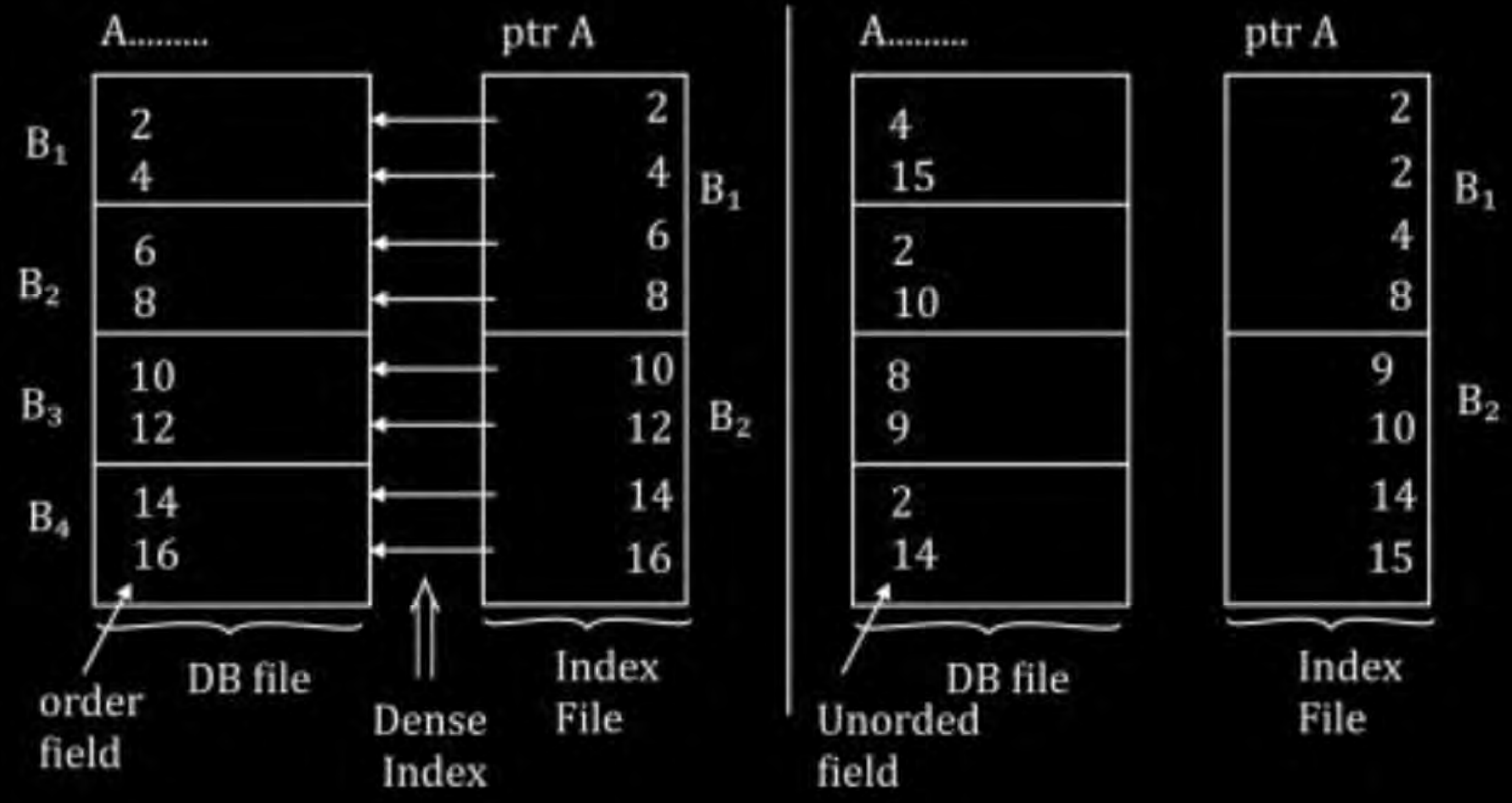
10101
12121
15151
22222
32343
33456
45565
58583
76543
76766
83821
98345

10101	Srinivasan	Comp. Sci.	65000	
12121	Wu	Finance	90000	
15151	Mozart	Music	40000	
22222	Einstein	Physics	95000	
32343	El Said	History	60000	
33456	Gold	Physics	87000	
45565	Katz	Comp. Sci.	75000	
58583	Califieri	History	62000	
76543	Singh	Finance	80000	
76766	Crick	Biology	72000	
83821	Brandt	Comp. Sci.	92000	
98345	Kim	Elec. Eng.	80000	

Dense Index [More entries in Index File]

⇒ For each DB record of DB file there exist entry in index file

[Dense Index]



⇒ (# of Index Entries in Index File) ≡ (# of DB records in DB file)

Sparse Index Files

- Sparse Index: contains index records for only some search-key values.
  - Applicable when records are sequentially ordered on search-key
- To locate a record with search-key value K we:
  - Find index record with largest search-key value $< K$
  - Search file sequentially starting at the record to which the index record points

10101	
32343	
76766	

10101	Srinivasan	Comp. Sci.	65000	
12121	Wu	Finance	90000	
15151	Mozart	Music	40000	
22222	Einstein	Physics	95000	
32343	El Said	History	60000	
33456	Gold	Physics	87000	
45565	Katz	Comp. Sci.	75000	
58583	Califieri	History	62000	
76543	Singh	Finance	80000	
76766	Crick	Biology	72000	
83821	Brandt	Comp. Sci.	92000	
98345	Kim	Elec. Eng.	80000	

Sparse Index [Less entries in Index File]

⇒ For set of DB records there exist entry in Index file such a Index is called Sparse Index.

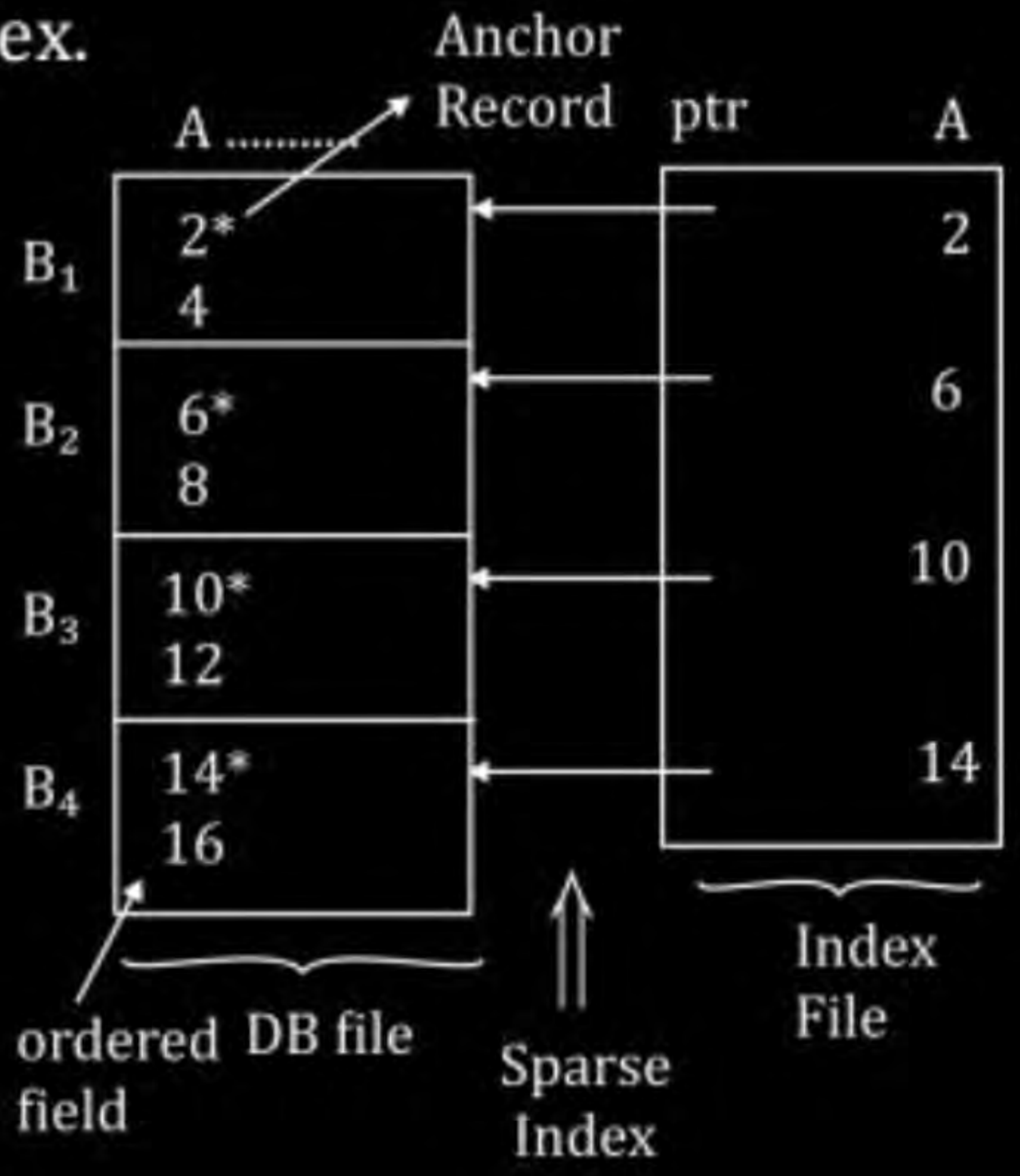
⇒ Sparse Index can build only over ordered field of file.

Anchor Record: First record of Block

[Sparse Index]

⇒ [# of entries in Index] < [# of DB records]

$$\left\{\begin{matrix}\text{\# of Index} \\ \text{entries}\end{matrix}\right\} \equiv \left\{\begin{matrix}\text{\# of DB} \\ \text{blocks}\end{matrix}\right\}$$



Index File:

$$\boxed{\text{BF (Block Factor) of Index} = \left\lfloor \frac{B-H}{K+P} \right\rfloor \text{ entries / block}}$$

$$\text{Index File} \left\{ \boxed{\begin{matrix} k_1 & p_1 \\ k_2 & p_2 \\ \vdots & \vdots \end{matrix}} \right. \quad \therefore \text{BF} = \left\lfloor \frac{\text{B}-\text{H}}{\text{k}-\text{p}} \right\rfloor$$

$$\boxed{\left\lfloor \frac{B-H}{R} \right\rfloor \text{ record / block} < \left\lfloor \frac{B-H}{K+P} \right\rfloor \text{ entries / block}}$$

$$\left\{ \begin{matrix} \text{\# of DB} \\ \text{File block} \end{matrix} \right\} > \left\{ \begin{matrix} \text{\# of Index} \\ \text{blocks} \end{matrix} \right\}$$

Example:

- Suppose that:
  - record size $R = 150$ bytes, block size $B = 512$ bytes, $r = 30000$ records
- Then, we get:
  - blocking factor $Bfr = B \text{ div } R = 512 \text{ div } 150 = 3$ records/block
  - number of file blocks $b = (r/\text{Bfr}) = (30000/3) = 10000$ blocks

Example:

Given the following data file

EMPLOYEE (NAME, SSN, ADDRESS, JOB, SAL, ...)

Suppose that:

- record size R=150 bytes, block size B=512 bytes r=30000 records
- For an index on the SSN field, assume the field size V_{SSN} =9 bytes, assume the record pointer size P_R=7 bytes. Then:

Example:

Given the following data file

EMPLOYEE (NAME, SSN, ADDRESS, JOB, SAL, ...)

Suppose that:

- record size $R=150$ bytes, block size $B=512$ bytes $r=30000$ records
- For an index on the SSN field, assume the field size $V_{SSN}=9$ bytes, assume the record pointer size $P_R=7$ bytes. Then:
  - index entry size $R_1=(V_{SSN}+P_R)=(9+7)=16$ bytes
  - index blocking factor $Bfr_1=B \text{ div } R_1=512 \text{ div} 16=32$ entries / block
  - number of index blocks $b=(r/Bfr_1)=(30000/32)=938$ blocks
  - binary search needs $\log_2 bI=\log_2 938=10$ block accesses

Q.

# of records of file: 16384

Block size: 4096 Bytes

Record size: 256 Bytes

Search key size: 22 Bytes

Pointer : 10 Bytes

⇒ I/O cost to access record without index based on ____

(a) Ordered field =

(b) Unordered Filed =

⇒ Index Dense [Using]

(a) # of index blocks at 1st level =

(b) I/O cost to access record: [Using 1st level] =

⇒ By using sparse Index:

(a) # of Index blocks at 1st level =

(b) I/O cost to access record: [Using 1st level] =

Types of Index

Single-level Ordered Indexes

- Primary indexes
- Clustering indexes
- Secondary indexes

Multilevel Indexes

Dynamic multilevel indexes Using B-Tress and B^+ Trees.

Types of Index

Type of Index
(X: search key)

- X is ordered field of DB file
  - X is primary key Alternative key of DB file ⇓ Primary Index
  - X is non key of OB file ⇓ Clustering Index
- X is Unordered field of DB file
  - X is primary key Alternative key of DB file ⇓ Secondary Index (over key)
  - X is non key of OB file ⇓ Secondary Index (Over Non-key)

THANK YOU!