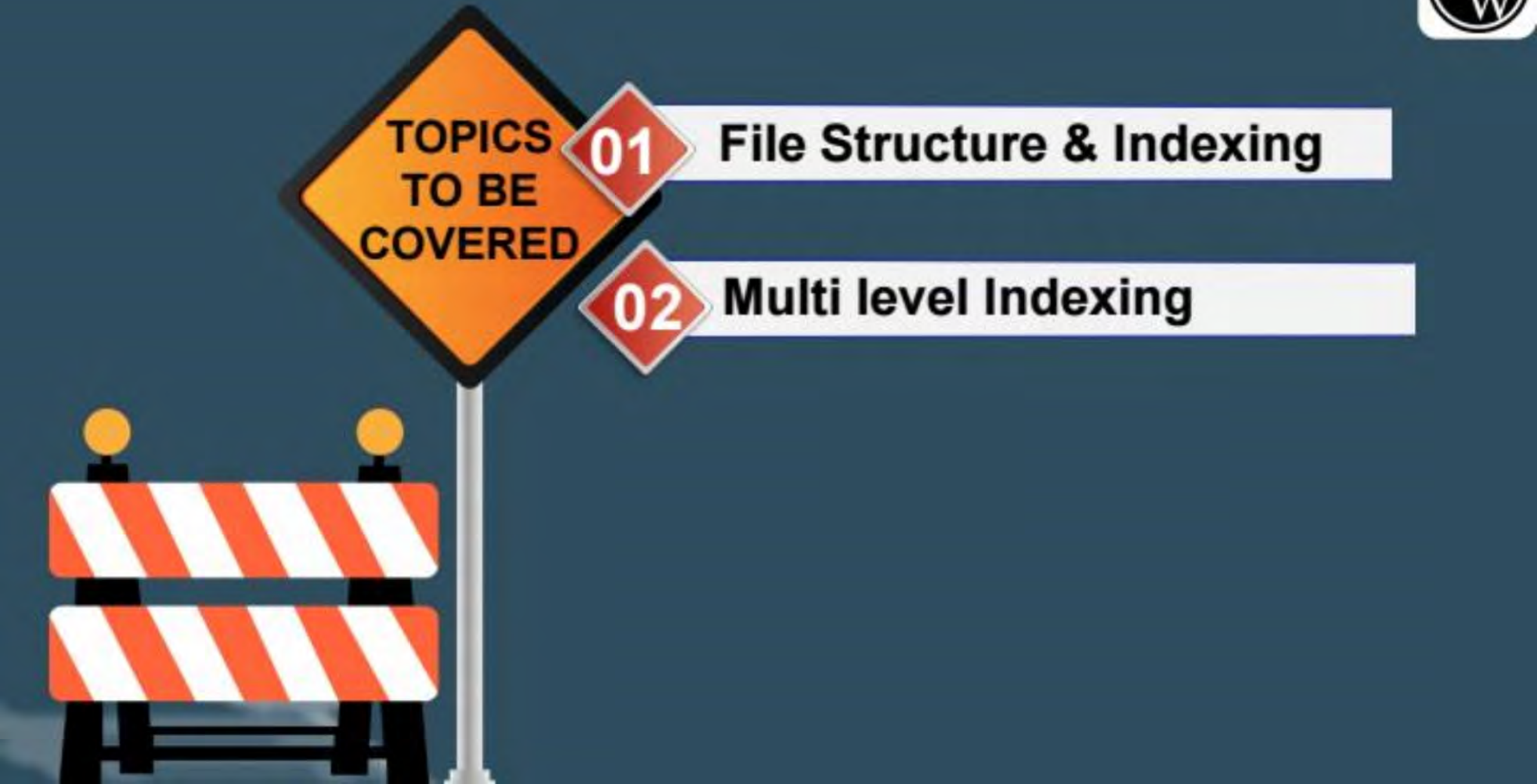
COMPUTER SCIENCE

Database Management System

File Org. & Indexing

Lecture_3

Vijay Agarwal sir


TOPICS TO BE COVERED
01
File Structure & Indexing
02
Multi level Indexing

'Blocks'

① SPANNED org.

② Unspanned org.

① ORDERED File $\longrightarrow \lceil \log_2 B \rceil$

② UNORDERED File $\longrightarrow$ Avg $= B/2$ Worst Case $= B$.

Indexing (Ordered file)

One Index Record Size = Size of key + Size of Pointer.

To Access Index Avg # Block Access $= \lceil \log_2 B_i \rceil$

To Access a Record Using Index Avg # Block Access $= \lceil \log_2 B_i \rceil + 1$

↳ DATA Block.

Dense Index

#Index Entries = # of DB Records.

SPARSE Index

#Index Entries = # of DB Blocks.

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]

Example:

- Suppose that:
  - record size R = 150 bytes, block size B = 512 bytes,
    r = 30000 records
- Then, we get:
  - blocking factor Bfr =
  - number of file blocks b =

Example:

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

    r = 30000 records
- Then, we get:
  - blocking factor Bfr = B div R = 512 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:

Index (Dense)

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

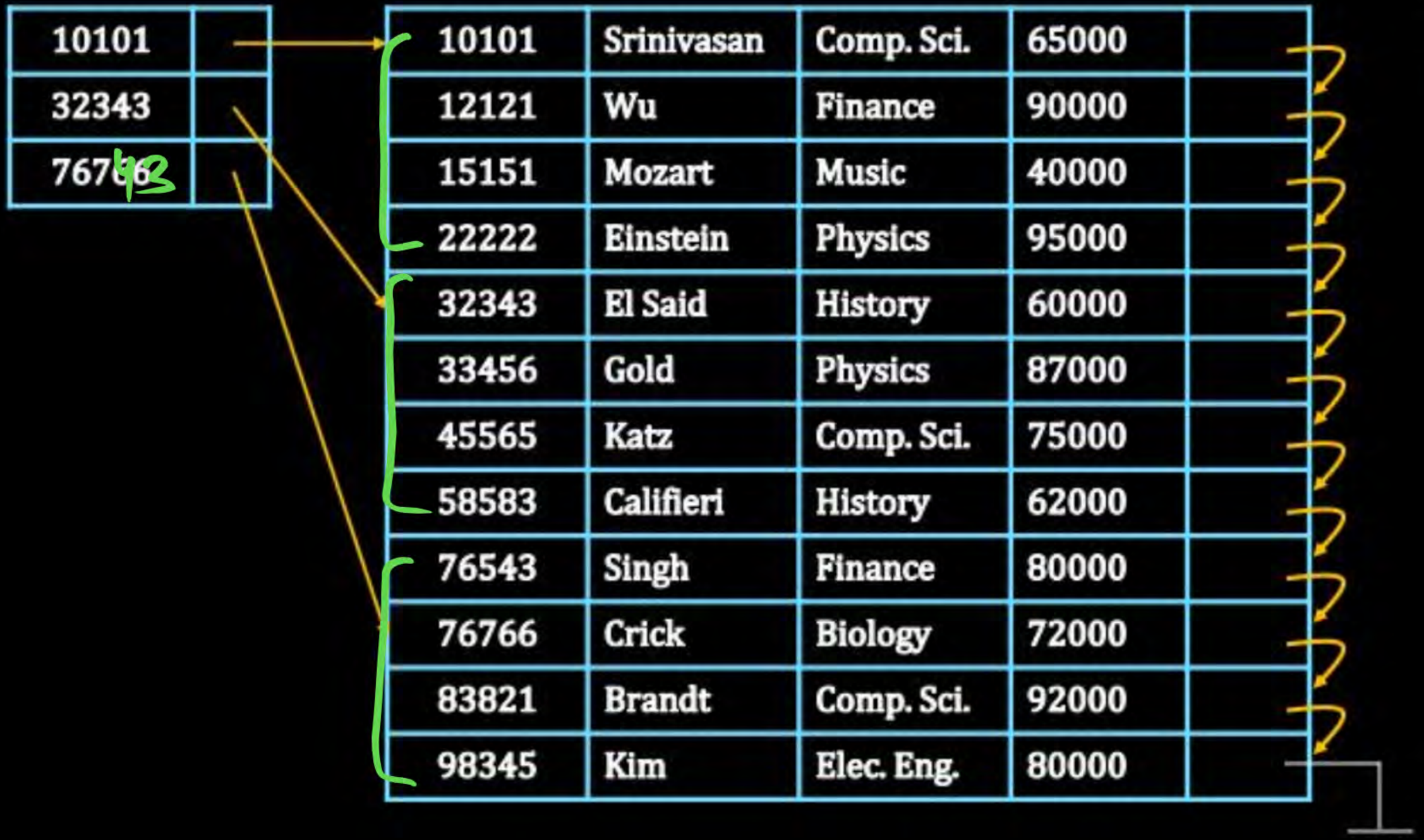
Dense Index Files

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

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

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

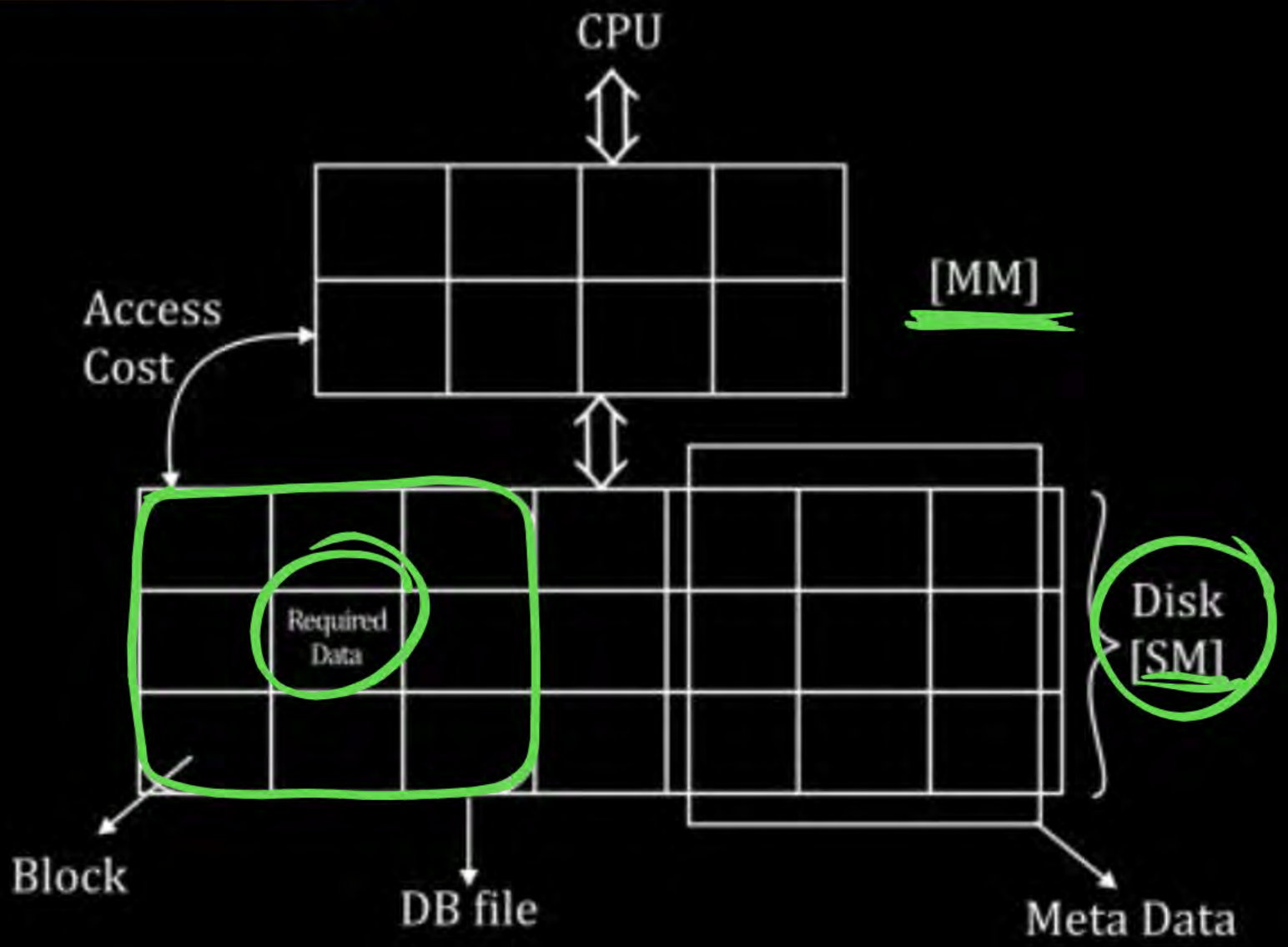
[Bi : Index Block]

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

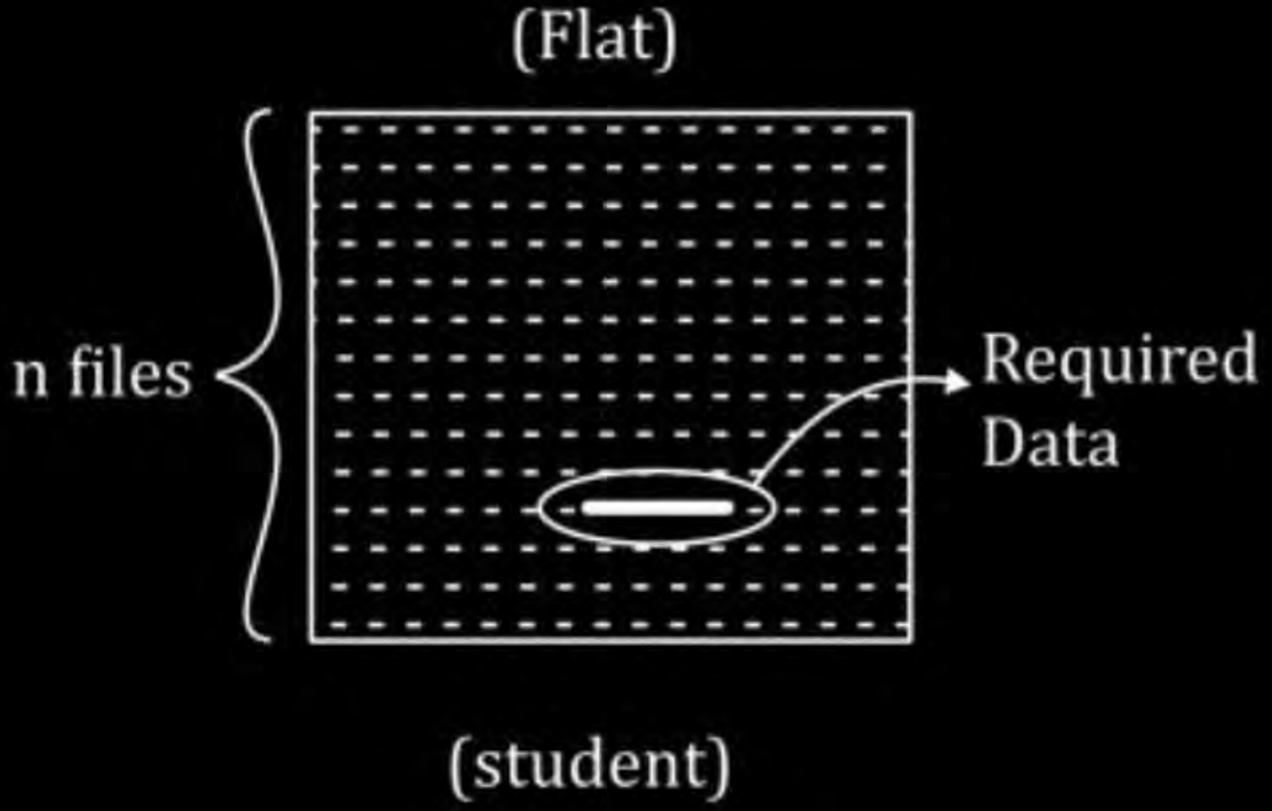
Meta Data [Data Dictionary] → DATA About 'DATA'.

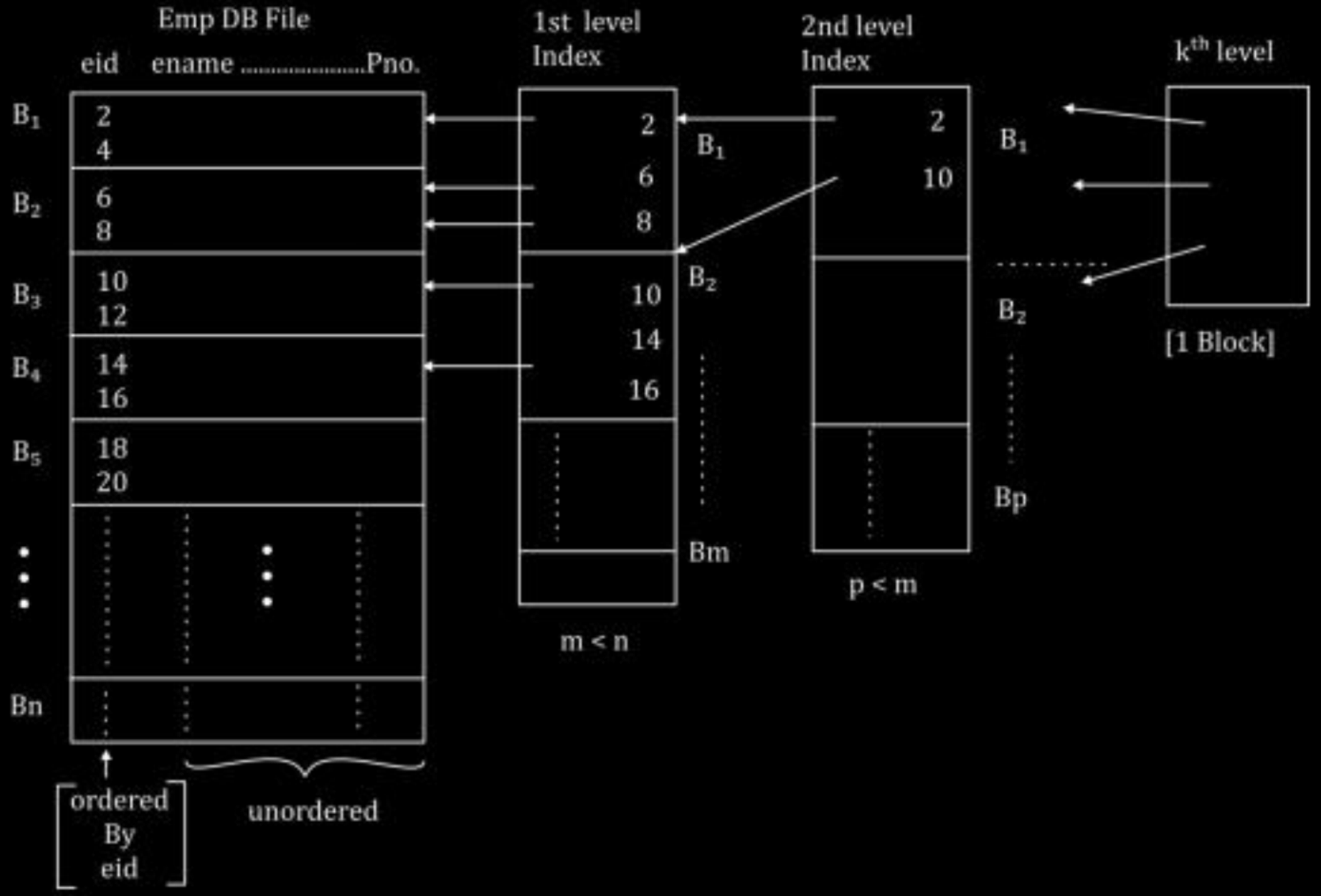
- → record format
- → Field format
- → Number of blocks allocated for file
- → Number of record in file
- → Ordered field of file, etc

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

Index File:

B-H

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

$$BF_i = \left\lfloor \frac{\text{Block Size}}{\text{Index Record Size}} \right\rfloor$$

K: key

P: Pointer.

Index File

k_1	p_1
k_2	p_2
⋮	⋮

$\therefore$ BF $= \left\lfloor \frac{B-H}{k+p} \right\rfloor$

BF of DB File

BF of Index File

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

{# of DB File block} > {# of Index blocks}

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

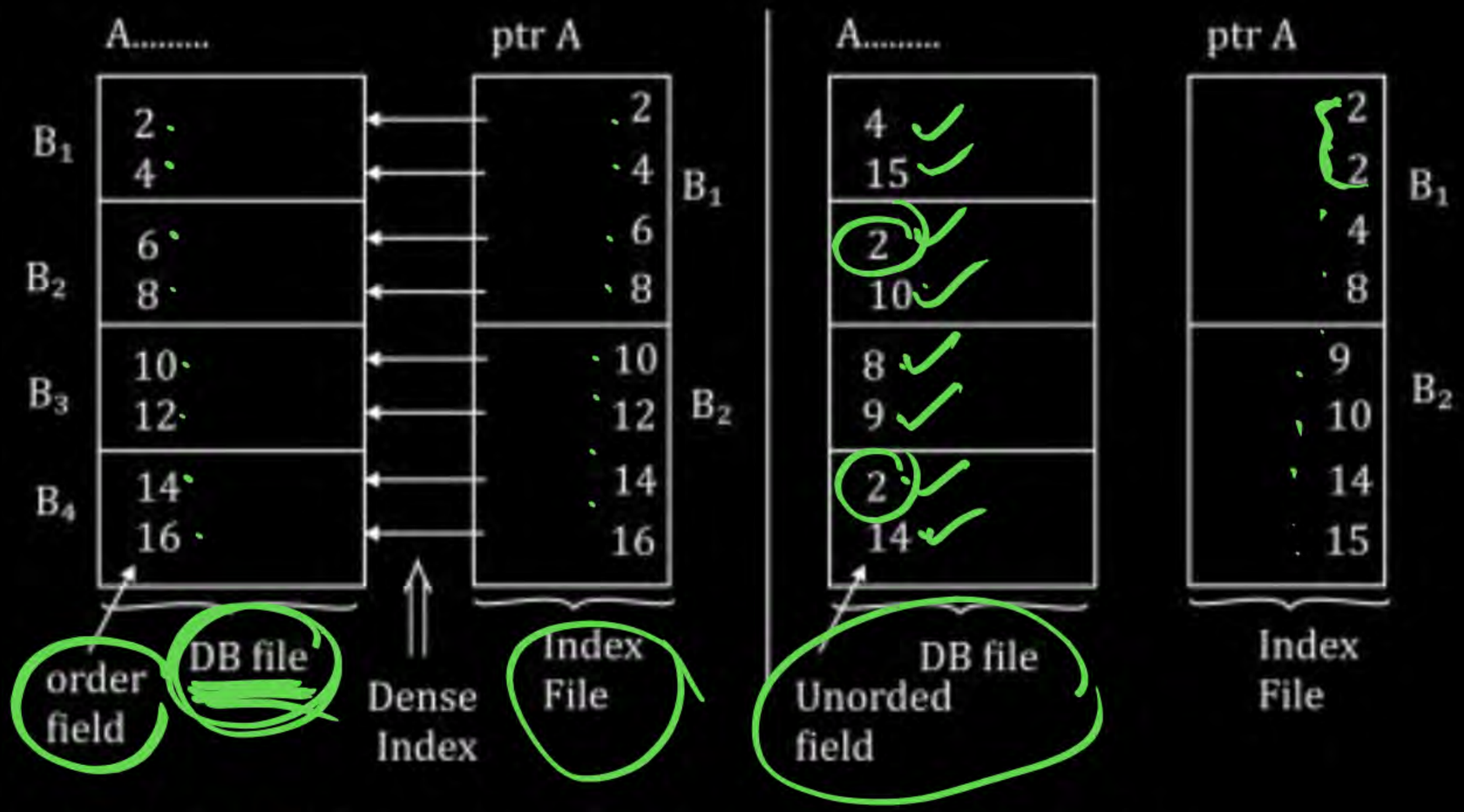
2) Sparse Index Files

Number of Index entries = Number of Blocks

Dense Index [More entries in Index File]

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

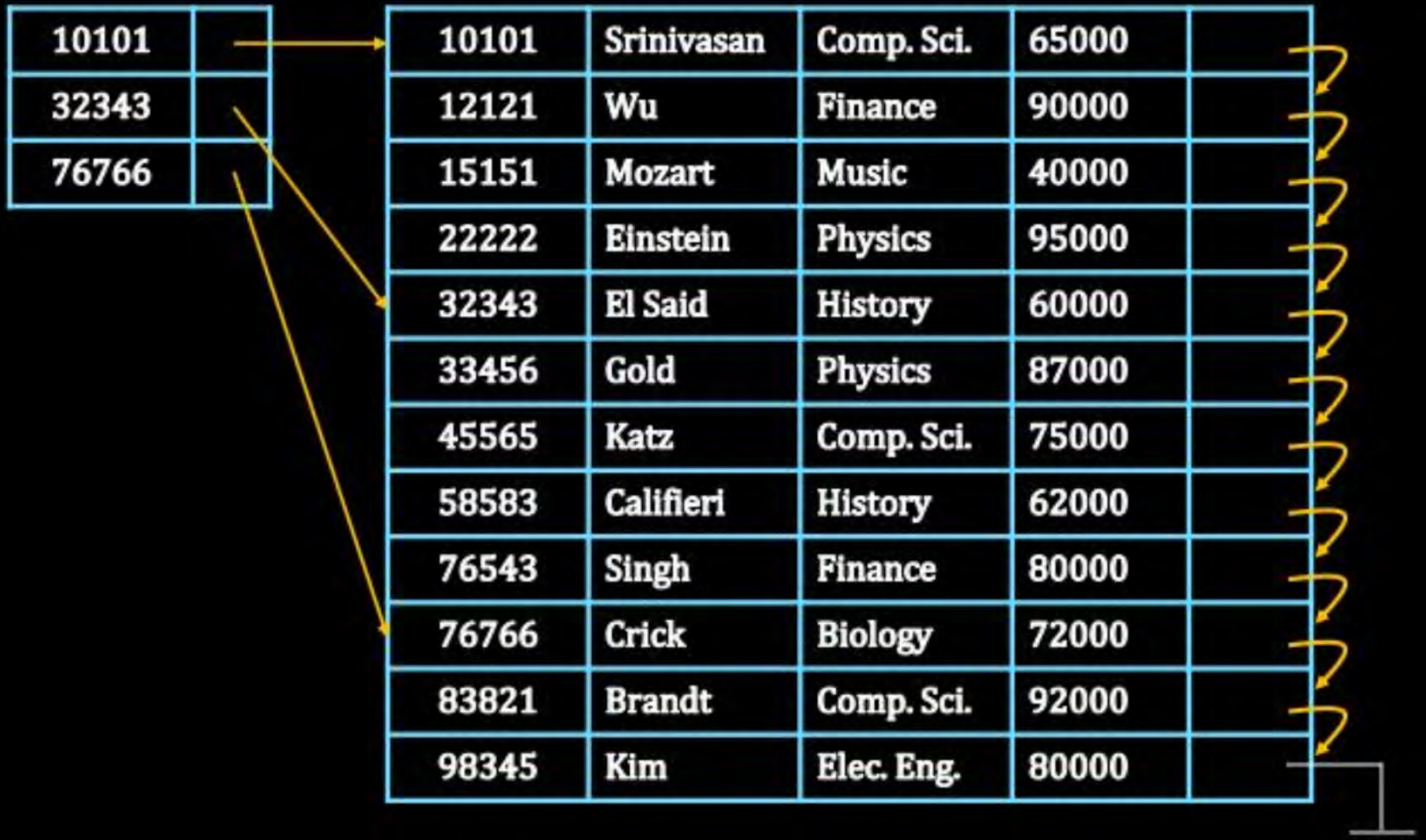
[Dense Index]



⇒ (# 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

Index File:

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

Index File

k_1	p_1
k_2	p_2
⋮	⋮

$\therefore$ BF $= \left\lfloor \frac{B-H}{k+p} \right\rfloor$

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

{# of DB File block} > {# of Index blocks}

Example:

- Suppose that:
  - record size R = 150 bytes, block size B = 512 bytes,
    r = 30000 records
- Then, we get:
  - blocking factor Bfr = B div R = 512 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

Index Record Size = 22 + 10 = 32B

⇒ 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] =

Example:

- Suppose that:

Unspanned ⇒ Floor

Assume
30001 : Yogesh
30002 : Zoya .

  - record size R = 150 bytes, block size B = 512 bytes,
    r = 30000 records
- Then, we get:
  - blocking factor Bfr = $\lfloor \frac{512}{150} \rfloor$ = 3 Record Per Block
  - number of file blocks b = $\lceil \frac{30002}{3} \rceil$ = 10001

Note
If we take 10000
then 10000 X 3 = 30,000

Assume Records = 3001

Block Size = 1024, Record Size = 100B.

Block factor of DB file $= \left\lfloor \frac{1024B}{100B} \right\rfloor =$ 10 Record Per Block

Total # Record = 3001

3001 : Zaheer Khan

Total # DB Block $= \left\lceil \frac{3001}{10} \right\rceil =$ 301 Data Block

If we taken 300 then total $300 \times 10 = 3000$ Records

Zaheer Khan Record Not found.

Types of Index

Single-level Ordered Indexes

- Primary indexes
- Clustering indexes
- Secondary indexes

Multilevel Indexes

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

Note Index is always Ordered file

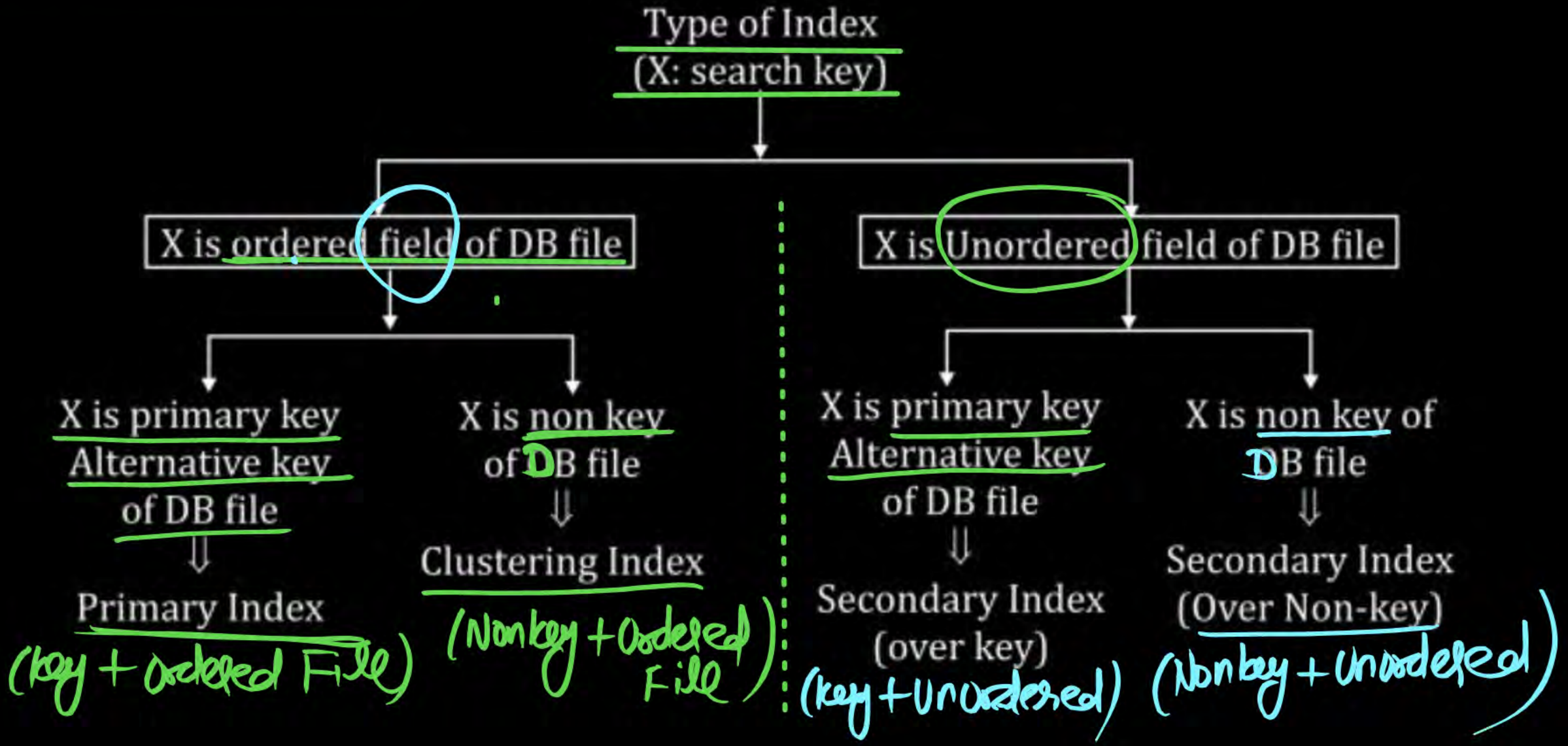
DB File

① Primary Index (key + Ordered file)
   (C.K = P.K + A.K)

② clusted Index (Non key + Ordered file)

③ Secondary Index (Non key + Unordered
   (OR)
   key + Unordered)

Types of Index



① PI (key + ordered)

② CI (Non key + ordered)

③ SI → (Non key + unordered)

→ (key + Unordered)

(Note) At Most One Primary Index possible Per Relation (Table)

(Note) More than One Secondary Index Possible

(Note) In a Relation either CI (or) PI Any One Possible, Not Both.

Relation R.

① PI ✓ CI ✗

② CI ✓ PI ✗

} Any one, Not Both on Relation R.

---

③ PI ✓ SI ✓

④ CI ✓ SI ✓

∴

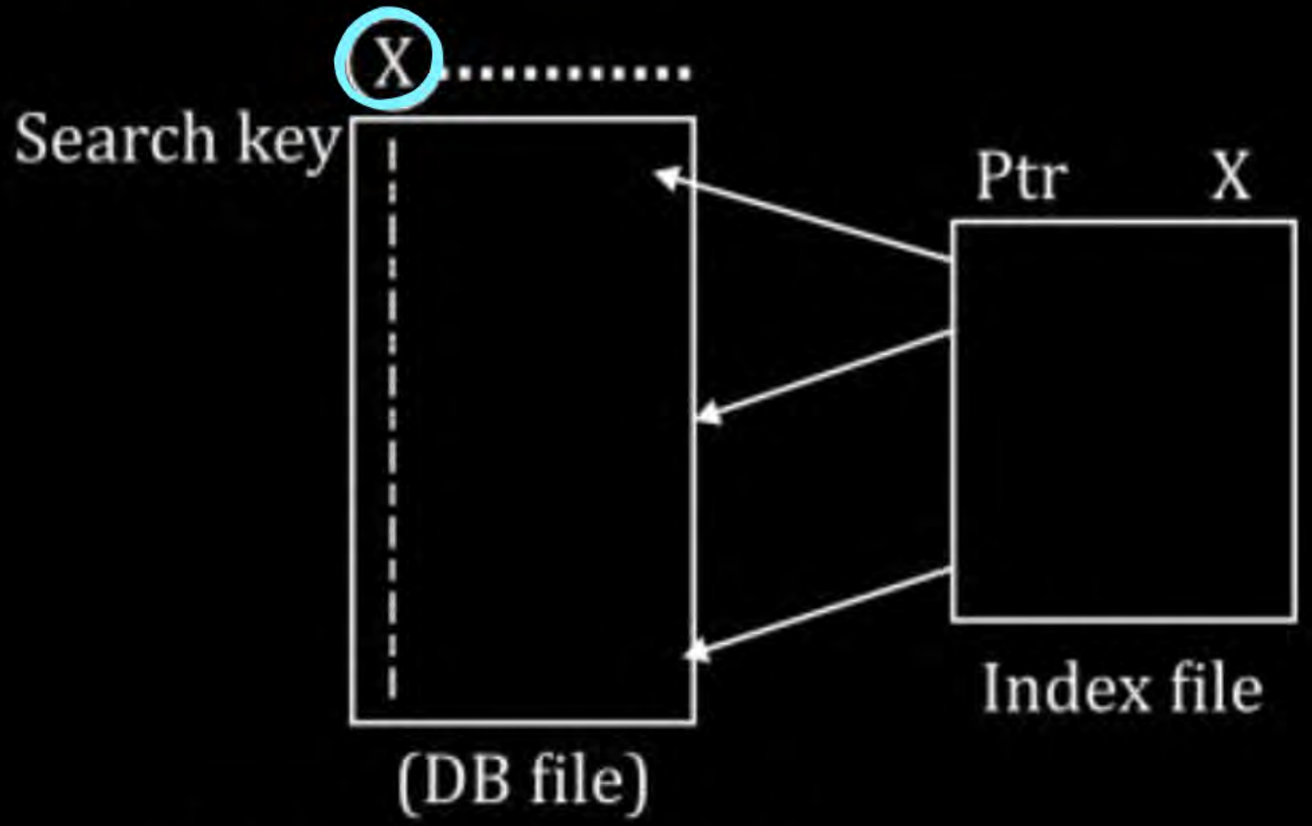
Q) Table 4 Attribute w, x, y, z then How many Attribute we After Indexing?

Soln

1 Primary Index (Any)

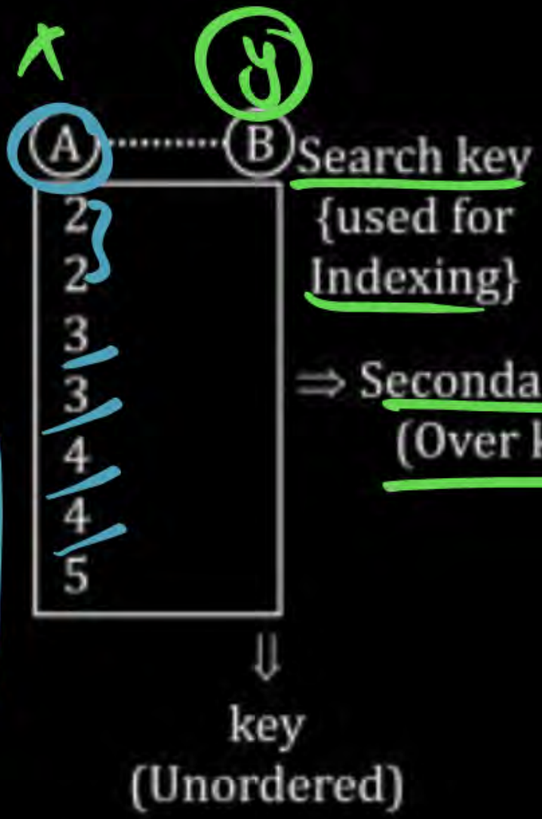
3 Secondary Index (Any)

Types of Index



Types of Index

Q.1 Data records ordered based on non-key and Index order key field of DB Table :



Non key + Ordered

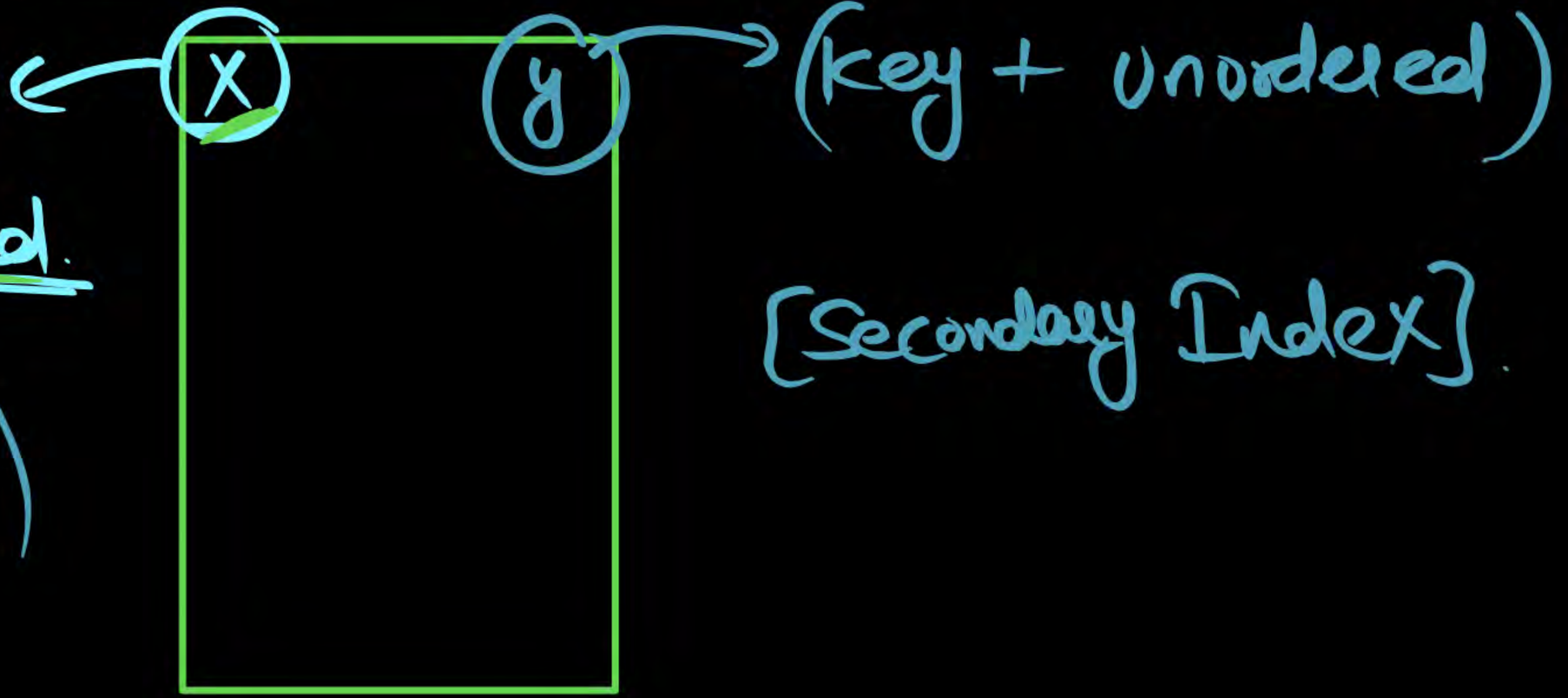
(clustering Index)

(key + unordered)

SI over Non key Attribute.

Types of Index

Q.1 Data records ordered based on non-key (x) and Index built over key field (y) of DB Table :

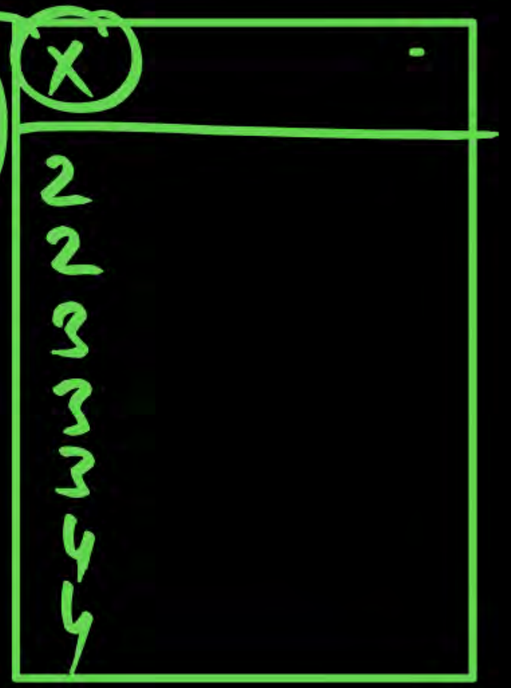


Types of Index

Q2 Data records ordered based on non-key and Index build over same non-key. Clustering Index Ans

(Non key + Ordered)

(Clustering Index)



Primary Index (key + Ordered DBFile)

↳ Dense (or) SPARSE.

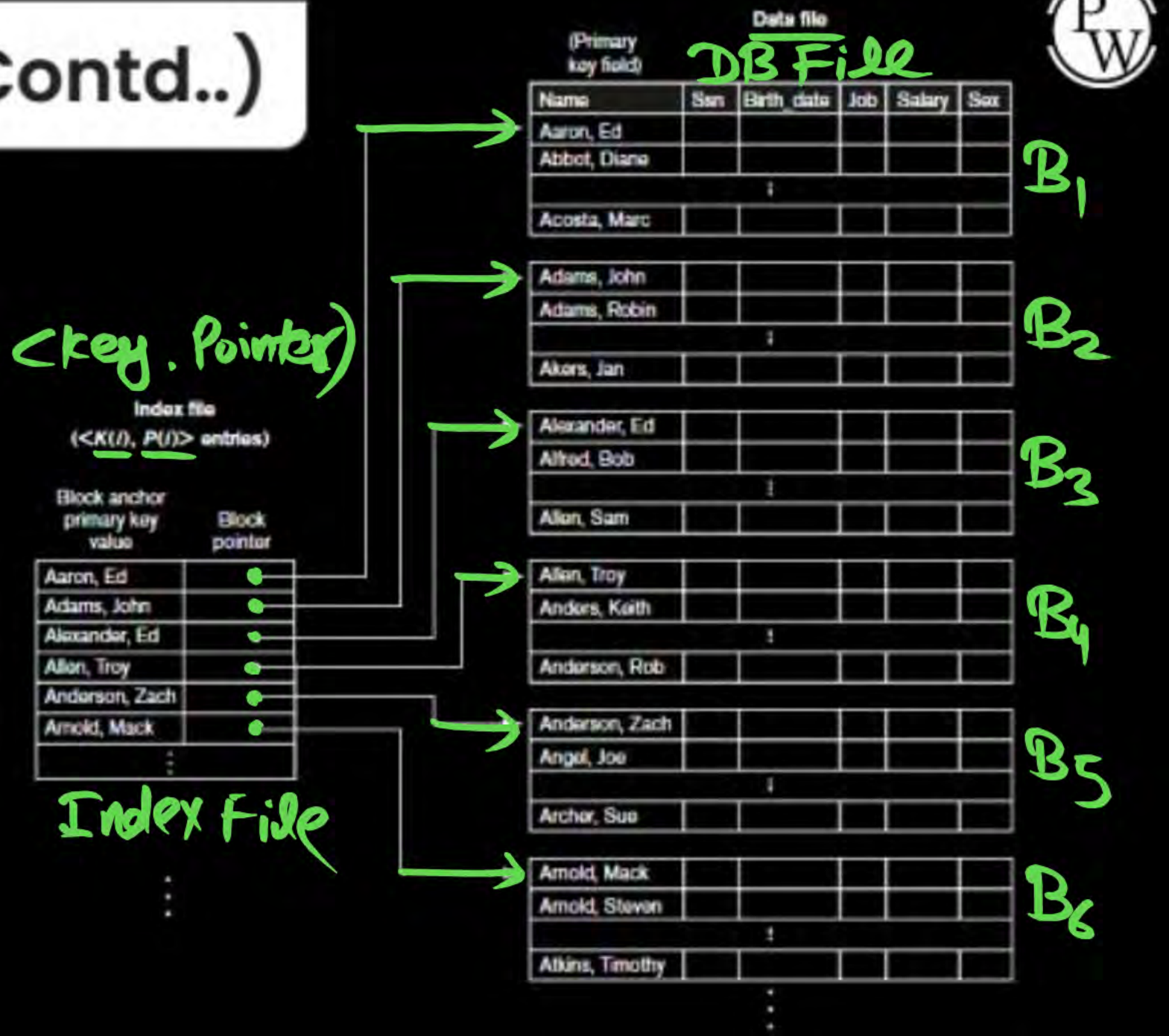
↳ Most Apply 'SPARSE' Index.

Primary Indexes

- Ordered file with two fields
  - Primary key, K(i)
  - Pointer to a disk block, P(i)
- One index entry in the index file for each block in the data file
- Indexes may be dense or sparse
  - Dense index has an index entry for every search key, value in the data file
  - Sparse index has entries for only some search values

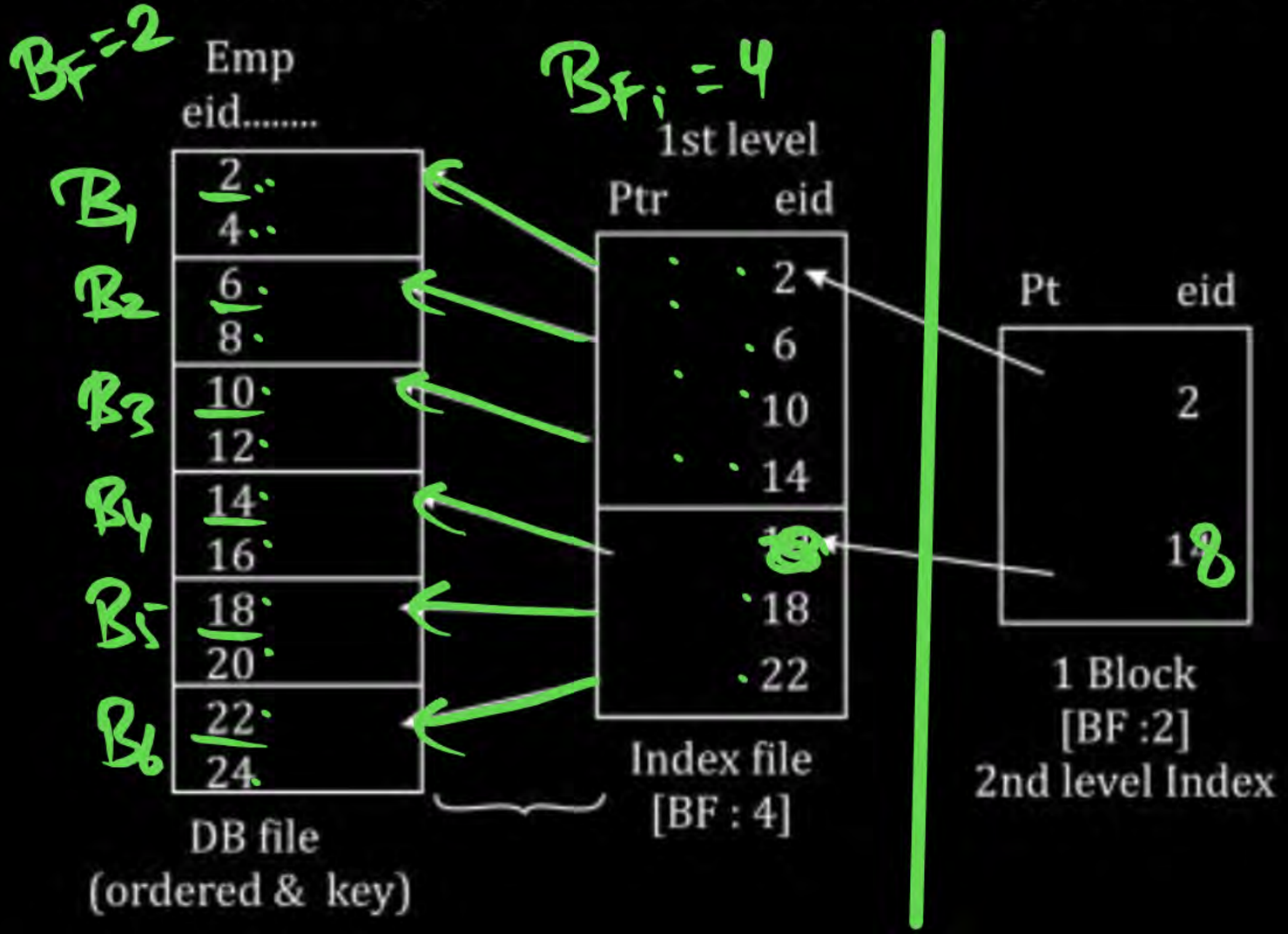
Primary Index (Contd..)

Primary index on the ordering key field of the file shown in Figure



(I) Primary Indexing:

Search key: Ordered field and key of the file.



⇒ Access cost to access record with PI with multilevel Index:

(K + 1) blocks

Note Primary Index can be Dense or sparse

[sparse PI can be preferred]

Note For any database relation at most one PI is possible

[because of Index over ordered field]

CI ✓ PI ✗

PI ✓ CI ✗

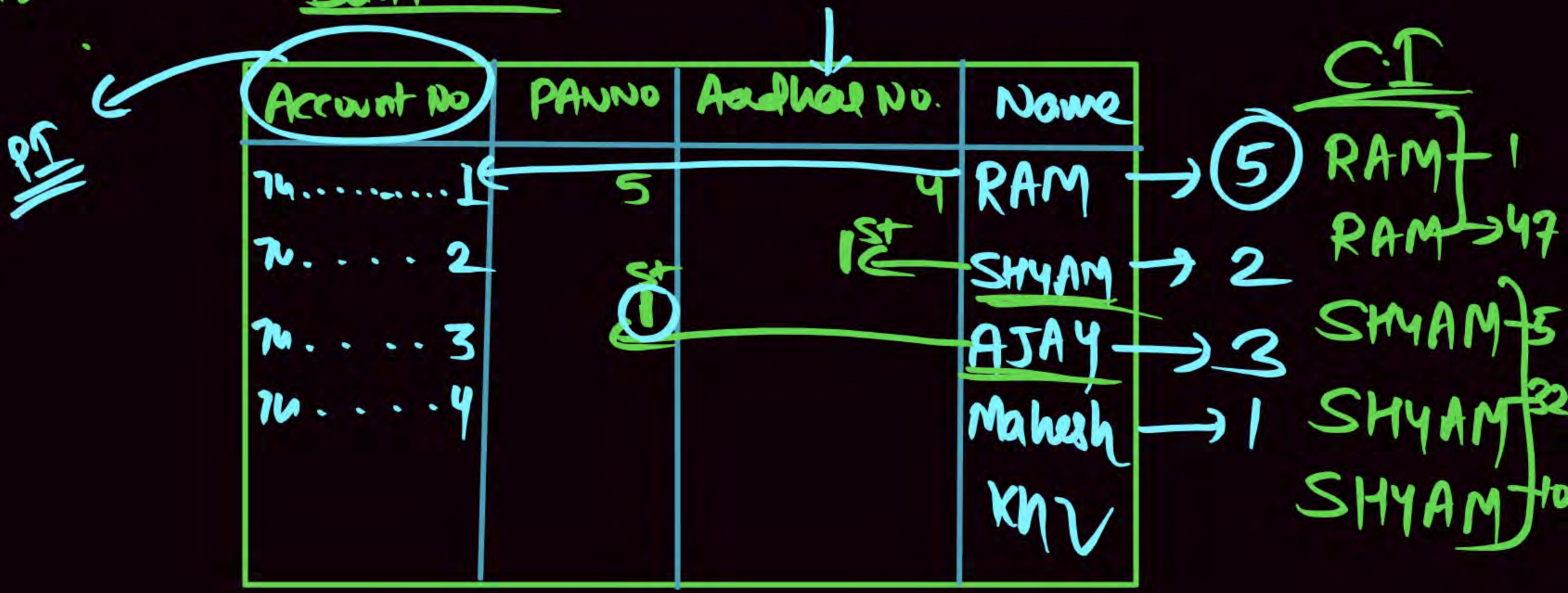
PI ✓ SI ✓

CI ✓ SI ✓

CI ✓ PI ✗

PI ✓ CI ✗

Bank DB



Q.1 Suppose that we have Ordered file of 30,000 records, stored on a disk with Block Size 1024 Byte, file records are of fixed length & unspanned of size 100 Byte (Record size) and suppose that we Have created a primary index on the key field of the file of size 9 Byte and Block pointer of size 6 Byte then find the average number of Block Access to search for a record using with and Without Index ?

Records = 30,000, Block Size = 1024 B, Record Size = 100B

Key = 9B, Pointer = 6Byte, Ordered file, Fixed length [unspanned]
Primary Index.

(1) (WITHOUT INDEX)

Block factor of DB File [unspanned] $= \left\lfloor \frac{1024B}{100B} \right\rfloor = 10$ Record Per Block.

Total Number of DB Record = 30,000

Total Number of Data Block [B] $= \left\lceil \frac{30000}{10} \right\rceil =$ 3000 DATA Blocks

ORDERED File

To Access a Record Avg # Block Access $= \lceil \log_2 B \rceil \Rightarrow \lceil \log_2 3000 \rceil =$ 12 Block Access Avg

Records = 30,000, Block Size = 1024 B, Record Size = 100B

Key = 9B, Pointer = 6 Byte, Ordered file, Fixed length [unspanned] Primary Index.

(2) (WITH Index)

One Index Record Size = 9 + 6 = 15 Byte

Block factor of Index file $[B_{F_i}] = \left\lfloor \frac{1024B}{15B} \right\rfloor = 68$ Index Entries per Block.

Primary Index: SPARSE: Total # Index Entries = 3000 (# DATA Blocks)

Total # Index Blocks $[B_i] = \lceil \frac{3000}{68} \rceil = 45$ Index Block.

(1) To Access Index Block Avg # Block Access $= \lceil \log_2 B_i \rceil \Rightarrow \lceil \log_2 45 \rceil = 6$ Block Access (Avg)

(2) To Access Record Using Index Avg # Block Access $= \lceil \log_2 B_i \rceil + 1 = \lceil \log_2 45 \rceil + 1 \Rightarrow 6 + 1 = 7$ Avg

Clustering Indexes

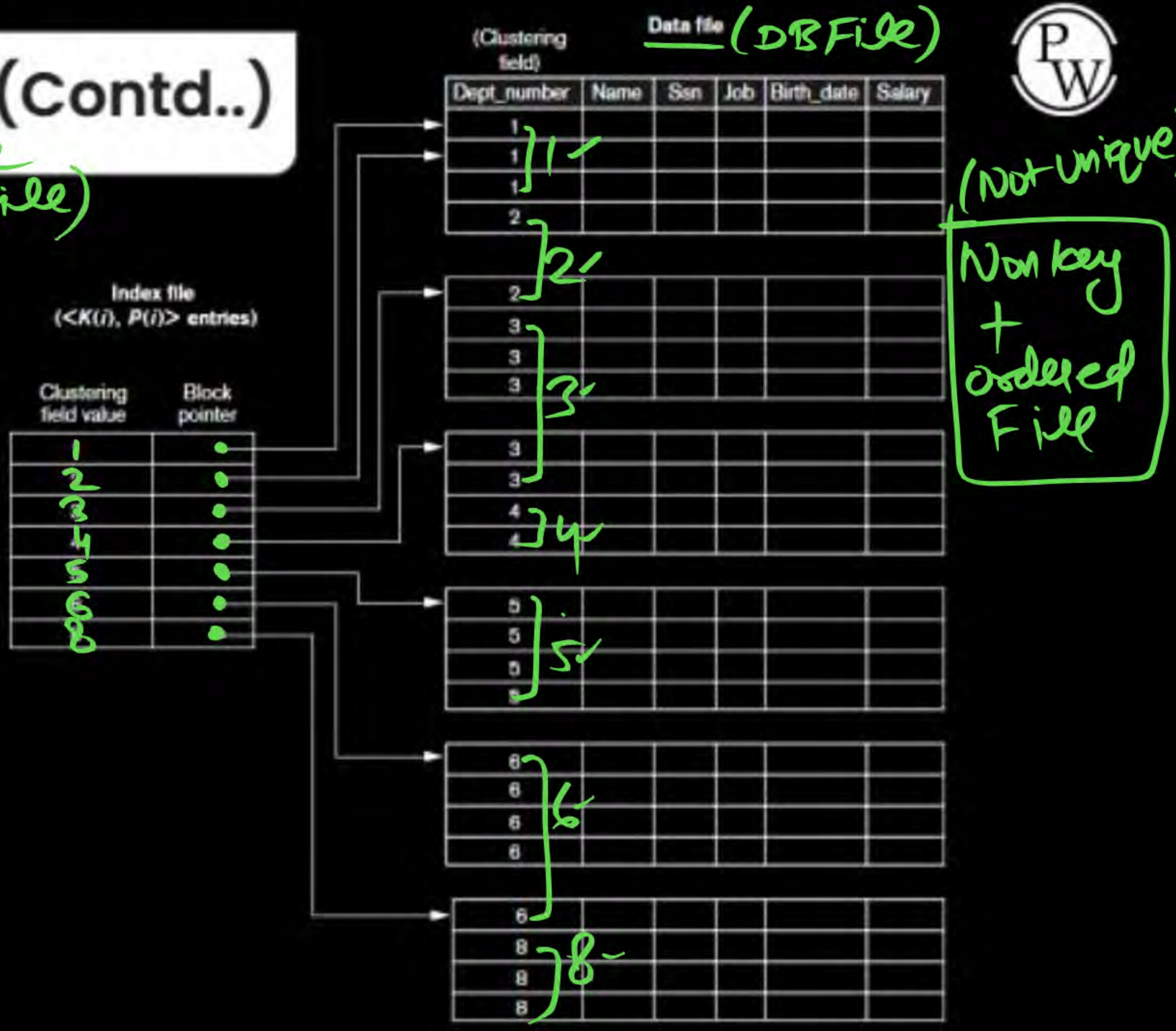
- Clustering field
  - File records are physically ordered on a nonkey field without a distinct value for each record
- Ordered file with two fields
  - Same type as clustering field
  - Disk block pointer

Clustering Indexes (Contd..)

↳ (Non key + ordered file)

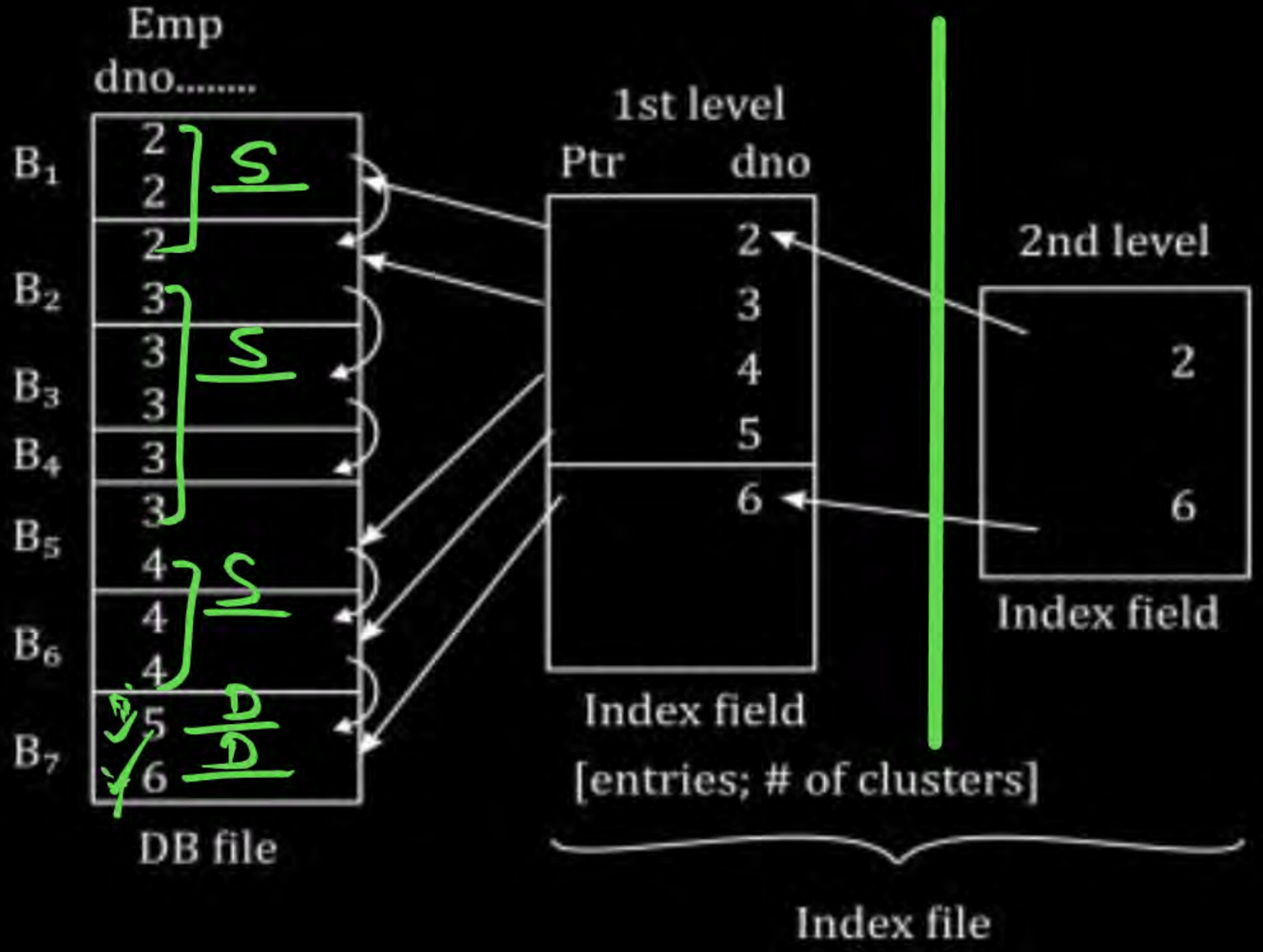
DB File

A clustering index on the Dept_number ordering nonkey field of an EMPLOYEE file



Clustering Index:

Search key: Ordered field & Non- key.



- Clustering Index mostly sparse Index [Dense CI also possible If each cluster with one record]

Note At most one CI is possible for any Database relation [ordering required]

Note For any DB relation can build either PI or CI but not both.

CI ✓ PI ✗

PI ✓ CI ✗

PI ✓ SI ✓

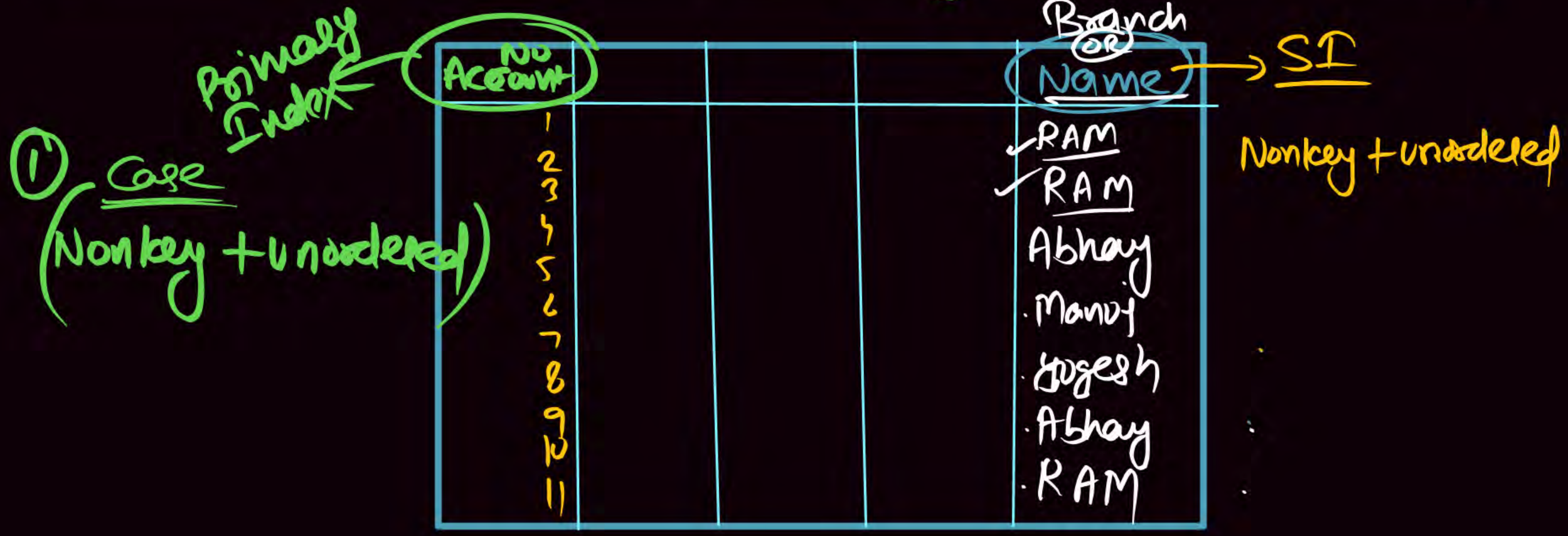
CI ✓ SI ✓

Secondary Index [Non key / Cand. key + Unordered File]

PI ✓ SI ✓

CI ✓ SI ✓

Secondary Index [① Non key + Unordered
② Cand. key File]



① Case (Nonkey + unordered)

Secondary Index [Non key / Cand. key + Unordered File]

(2) Case
(key + unordered)

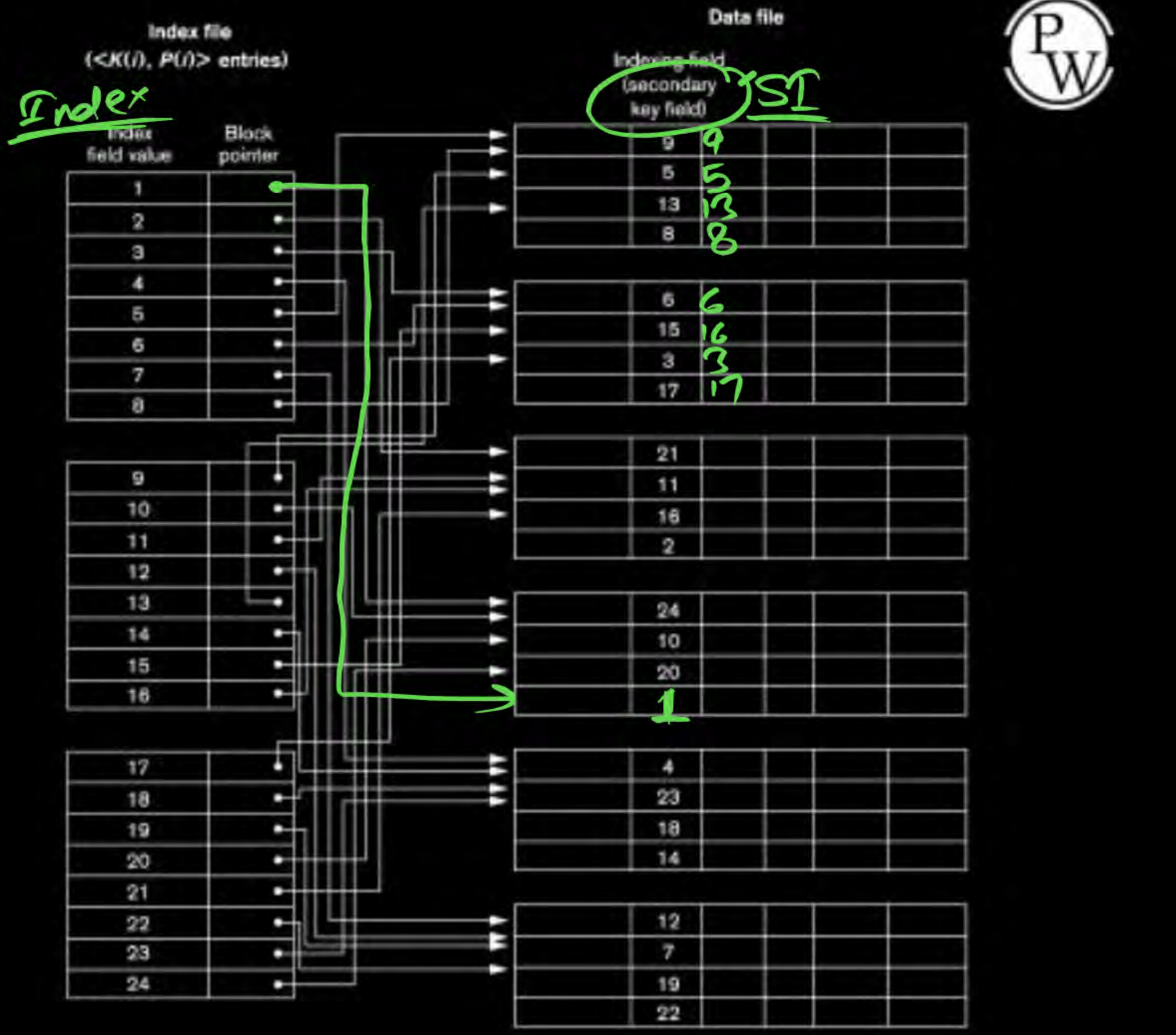
(key + unordered)

Secondary Index [Non key / Cand. key + Unordered File]

Secondary Indexes

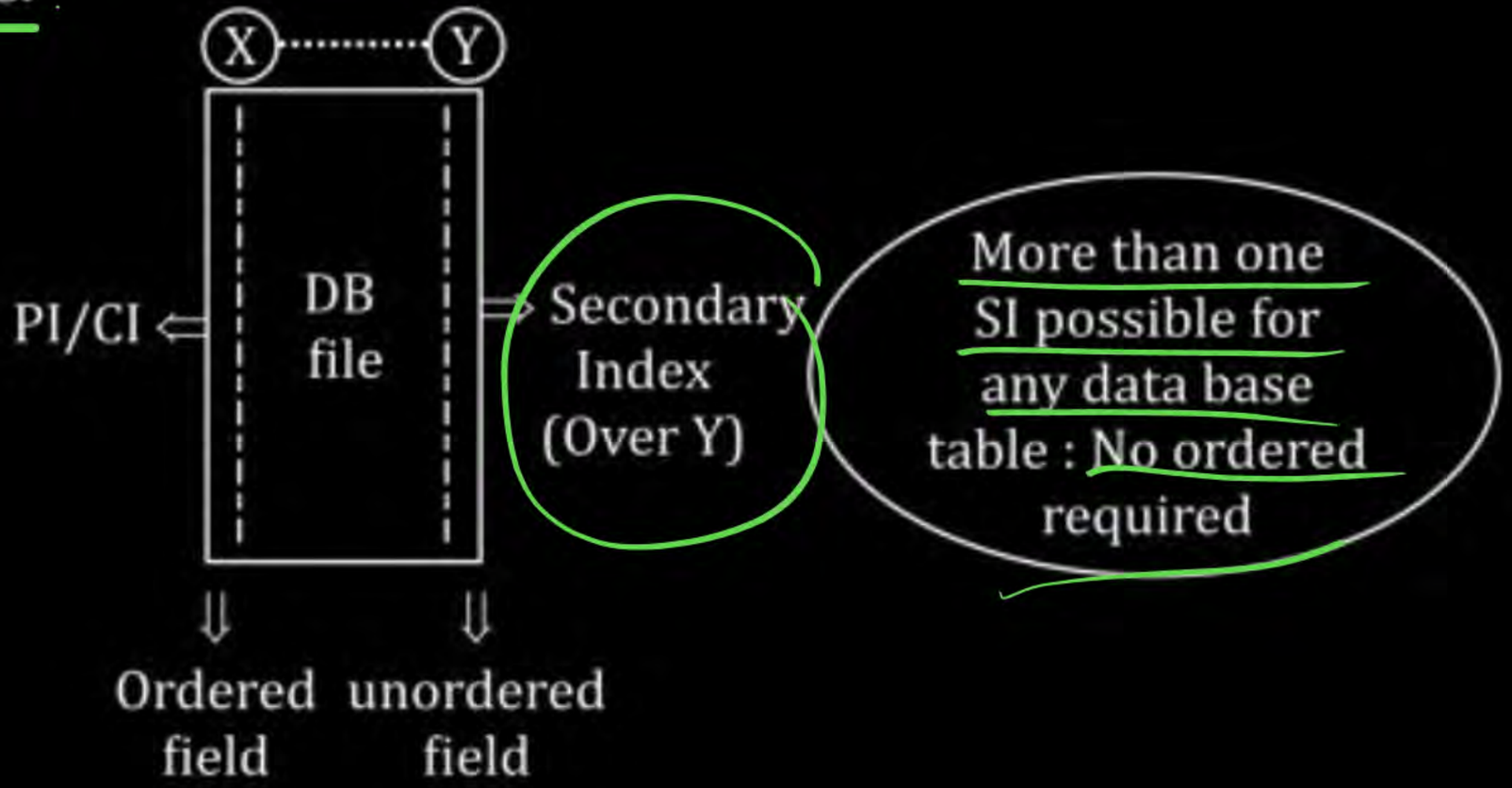
- Secondary Index
  - A secondary index provides a secondary means of accessing a file for which some primary access already exists.
  - The secondary index may be on a field which is a candidate key and has a unique value in every record, or a non-key with duplicate values.
  - The index is an ordered file with two fields.
    - The first field is of the same data type as some non-ordering field of the data file that is an indexing field.
    - The second field is either a block pointer or a record pointer.
    - There can be many secondary indexes (and hence, indexing fields) for the same file.
- Includes one entry for each record in the data file; hence, it is a dense index

Secondary Indexes (Contd..)

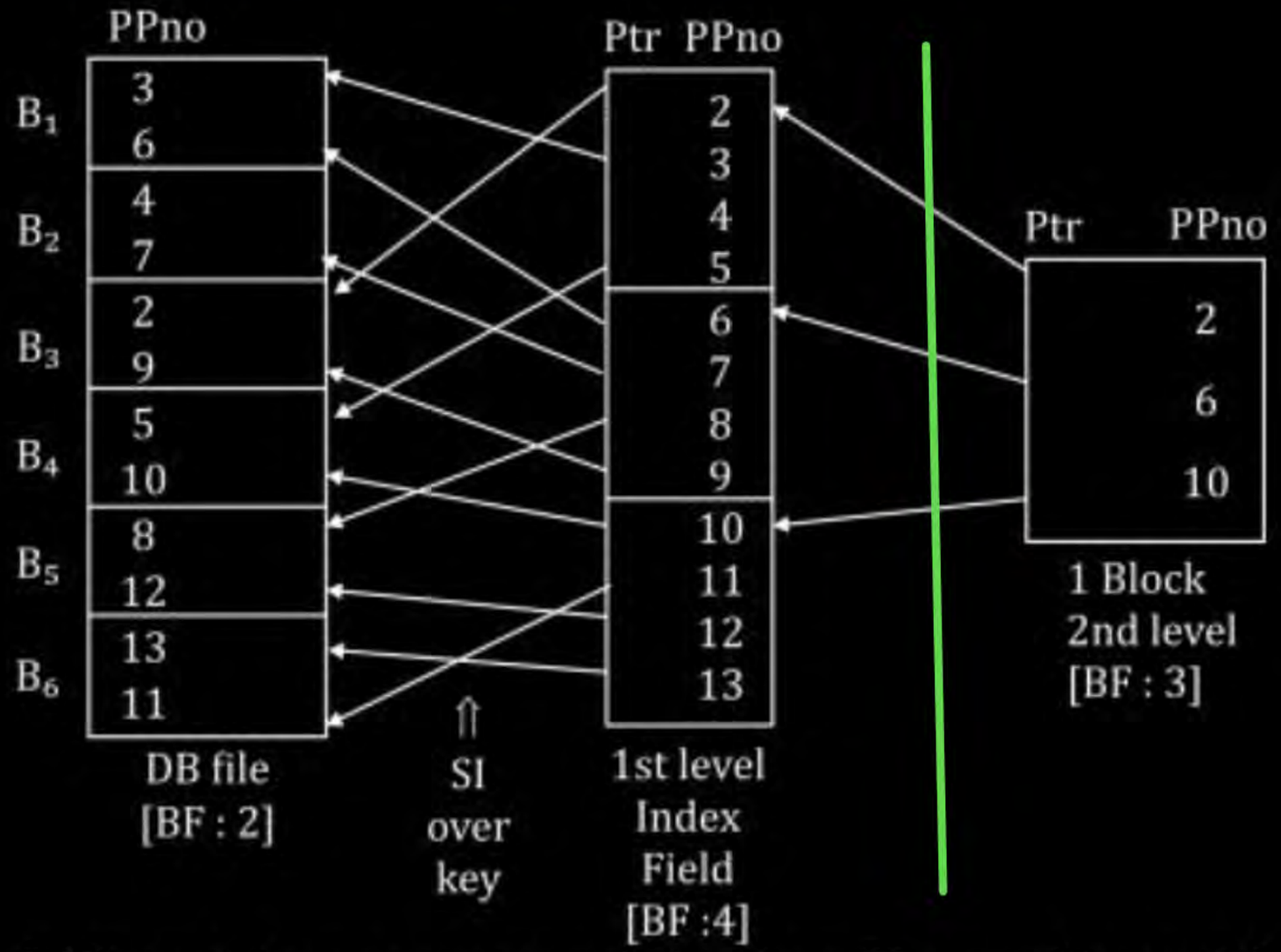


Secondary Index:

- Search key: Unordered field & Key/Non- key.
- Secondary possible way to access data using index even PI/CI indexes already exists.

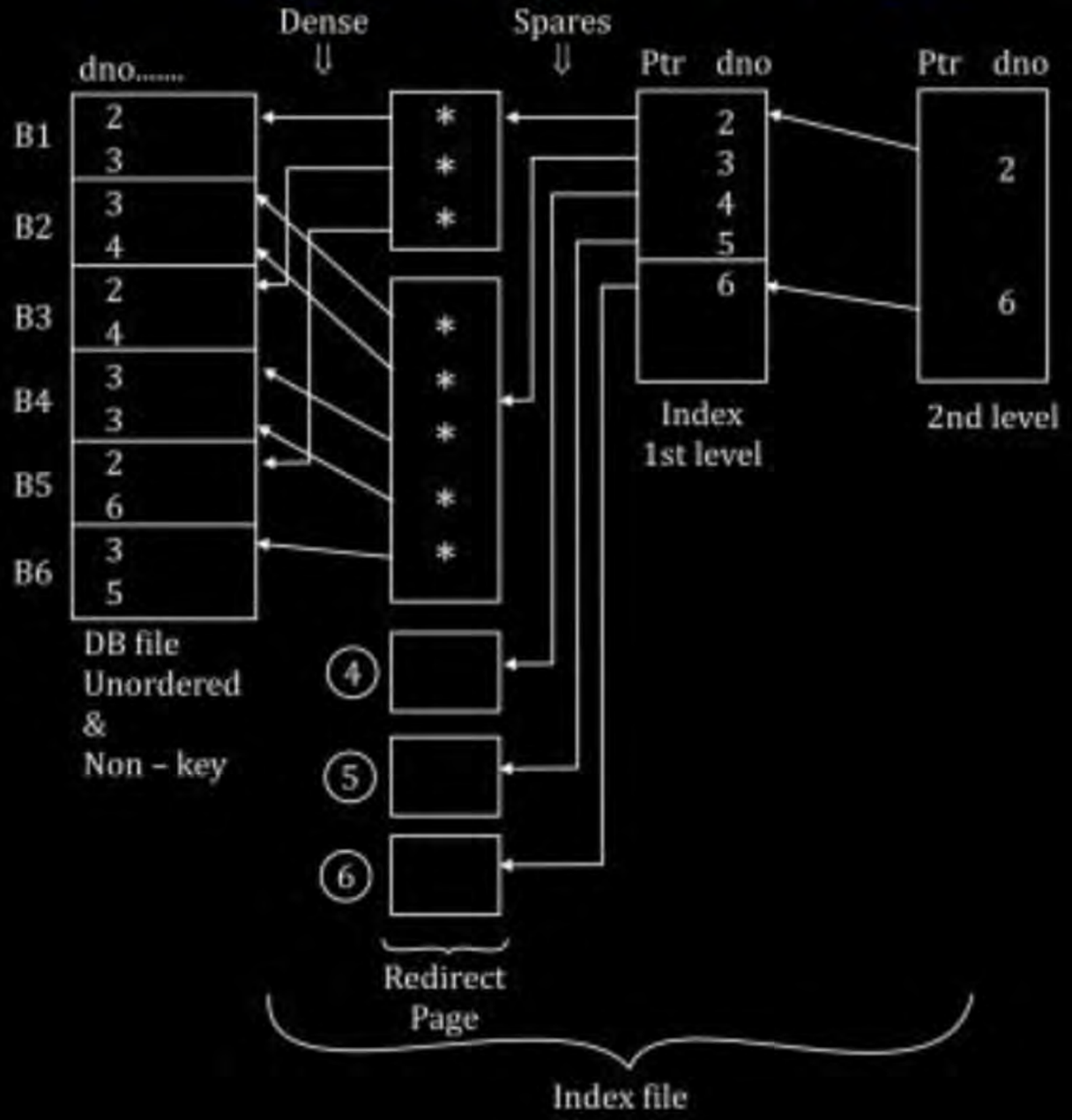


- Secondary Index (over key):



- I/O cost to access record using SI over key with MLI is (K + 1) blocks.

- ## Secondary Index (over Non-key):



- I/O cost to access record of some non-key using SI over non-key with MLI: {k+ # of blocks of DB equal to # of pointers in given redirect page}

Q.2 Consider a secondary Index on the key field of the file of question number 1, then find the Average number of Block Access to Access a record using with & without Index?

Number of records = 30000 Block size = 1024 B
record size = 1000 B
Key = 9B Bp = 6 Byte
Unspanned & Unordered.

Unordered & Secondary (Dense)

Records = 30,000, Block Size = 1024 B, Record Size = 100B

Key = 9B, Pointer = 6Byte, Unordered file, Secondary (Dense Index)

(1) (WITHOUT INDEX)

Block factor of DB File [unspanned] $= \left\lfloor \frac{1024B}{100B} \right\rfloor = 10$ Record Per Block.

Total Number of DB Record = 30,000

Total Number of Data Block [B] $= \left\lceil \frac{30000}{10} \right\rceil =$ 3000 DATA Blocks

ORDERED File

To Access a Record Avg # Block Access $= \frac{B}{2} \Rightarrow \frac{3000}{2} =$ 1500 Block Access Avg

Records = 30,000, Block Size = 1024 B, Record Size = 100B

Key = 9B, Pointer = 6Byte, Unordered, Secondary (Dense)

② (WITH Index)

One Index Record Size = 9 + 6 = 15 Byte

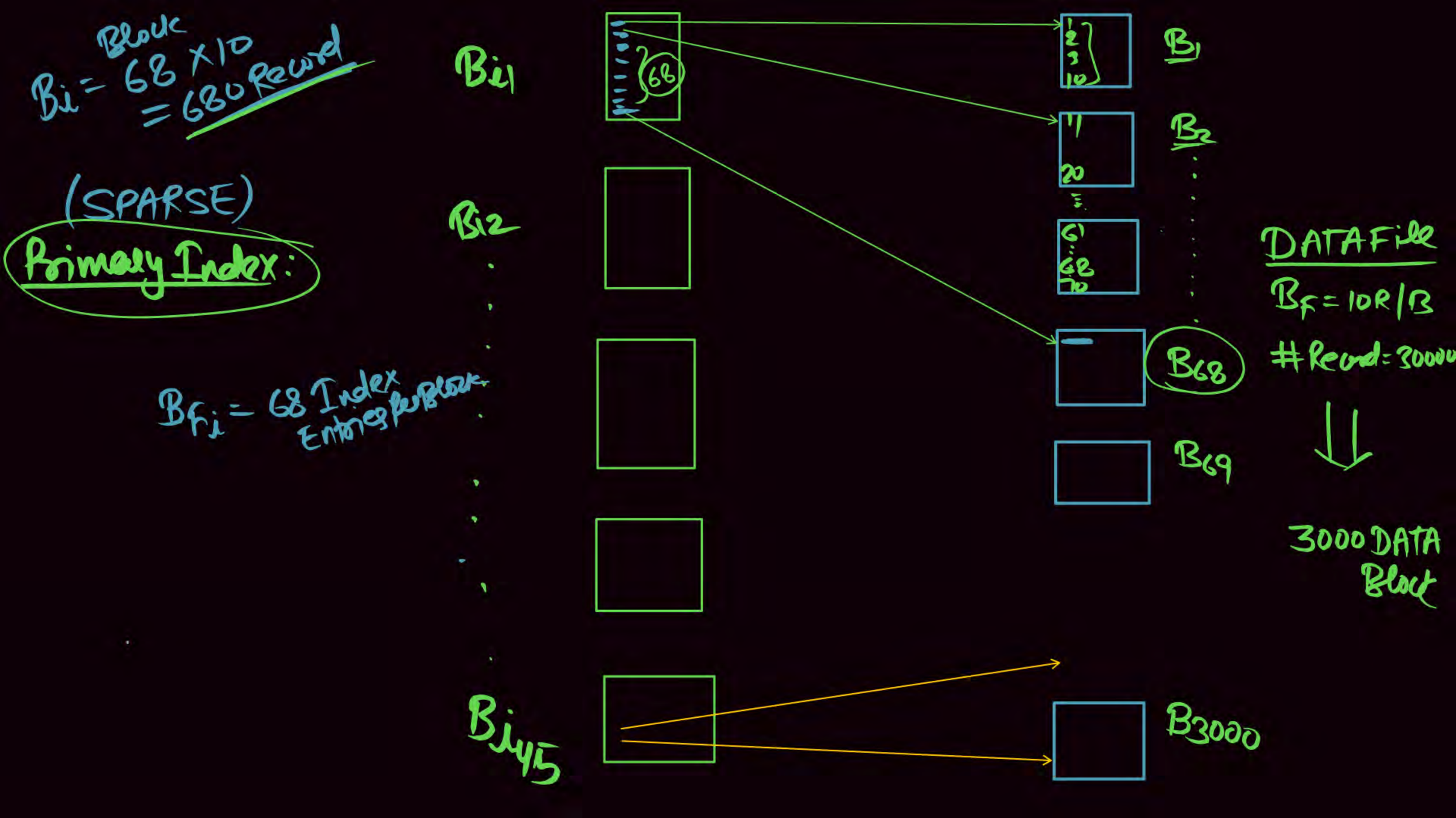
Block factor of Index file $[BF_i] = \left\lfloor \frac{1024B}{15B} \right\rfloor = 68$ Index Entries per Block.

Secondary Dense: Total # Index Entries = 30000 (# DATA Record)

Total # Index Blocks $[B_i] = \lceil \frac{30000}{68} \rceil = 442$ Index Block.

① To Access Index Block Avg # Block Access $= \lceil \log_2 B_i \rceil \Rightarrow \lceil \log_2 442 \rceil = 9$ Block Access (Avg)

② To Access Record Using Index Avg # Block Access $= \lceil \log_2 B_i \rceil + 1 = \lceil \log_2 442 \rceil + 1 = 9 + 1 = 10$ Avg


Block
Bi = 68 X 10
= 680 Record
(SPARSE)
Primary Index:
Bfi = 68 Index Entries per Block
Bi1
Bi2
Bi45
68
1
2
3
10
11
20
61
68
70
B1
B2
B68
B69
B3000
DATAFile
Bf = 10R/B
# Record = 30000
3000 DATA Block


Bi1 : 68 Record
Bfi = 68
(Dense)
Secondary Index
Bi1
Bi2
Bi3
Bi442
1
2
3
10
11
20
61
68
B1
B2
B68
B69
B3000
DATAFile
BF = 10R/B
#Record = 30000
3000 DATA Block

Q WHY Multi-level Indexing ?

⇓

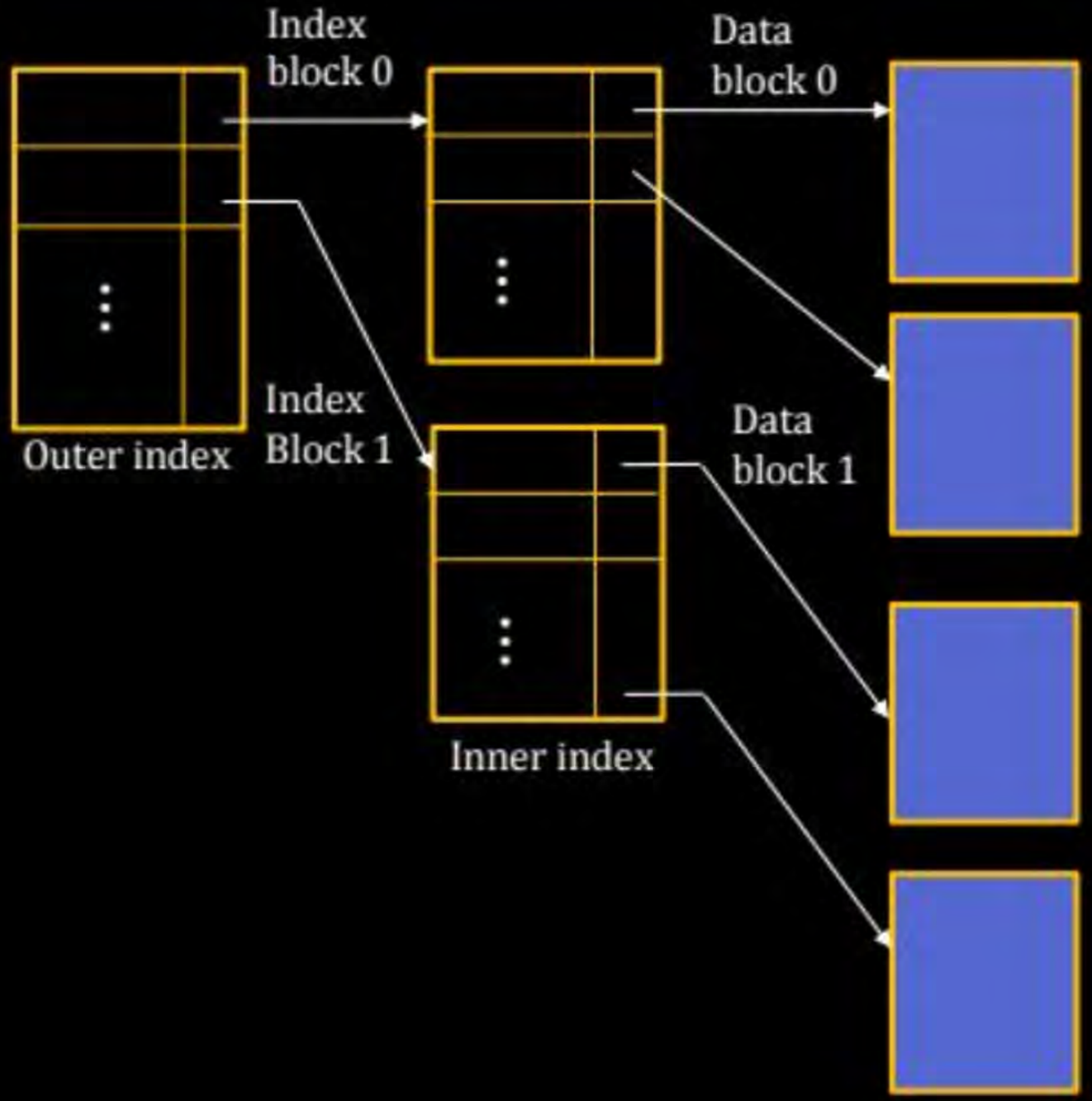
Similar to
Mutilevel Paging.

Bi$_{4\mu2}$ Index Block

Multilevel Index

- If index does not fit in memory, access becomes expensive.
- Solution: treat index kept on disk as a sequential file and construct a sparse index on it.
  - outer index - a sparse index of the basic index
  - inner index - the basic index file
- If even outer index is too large to fit in main memory, yet another level of index can be created, and so on.
- Indices at all levels must be updated on insertion or deletion from the file.

Multilevel Index (Contd..)

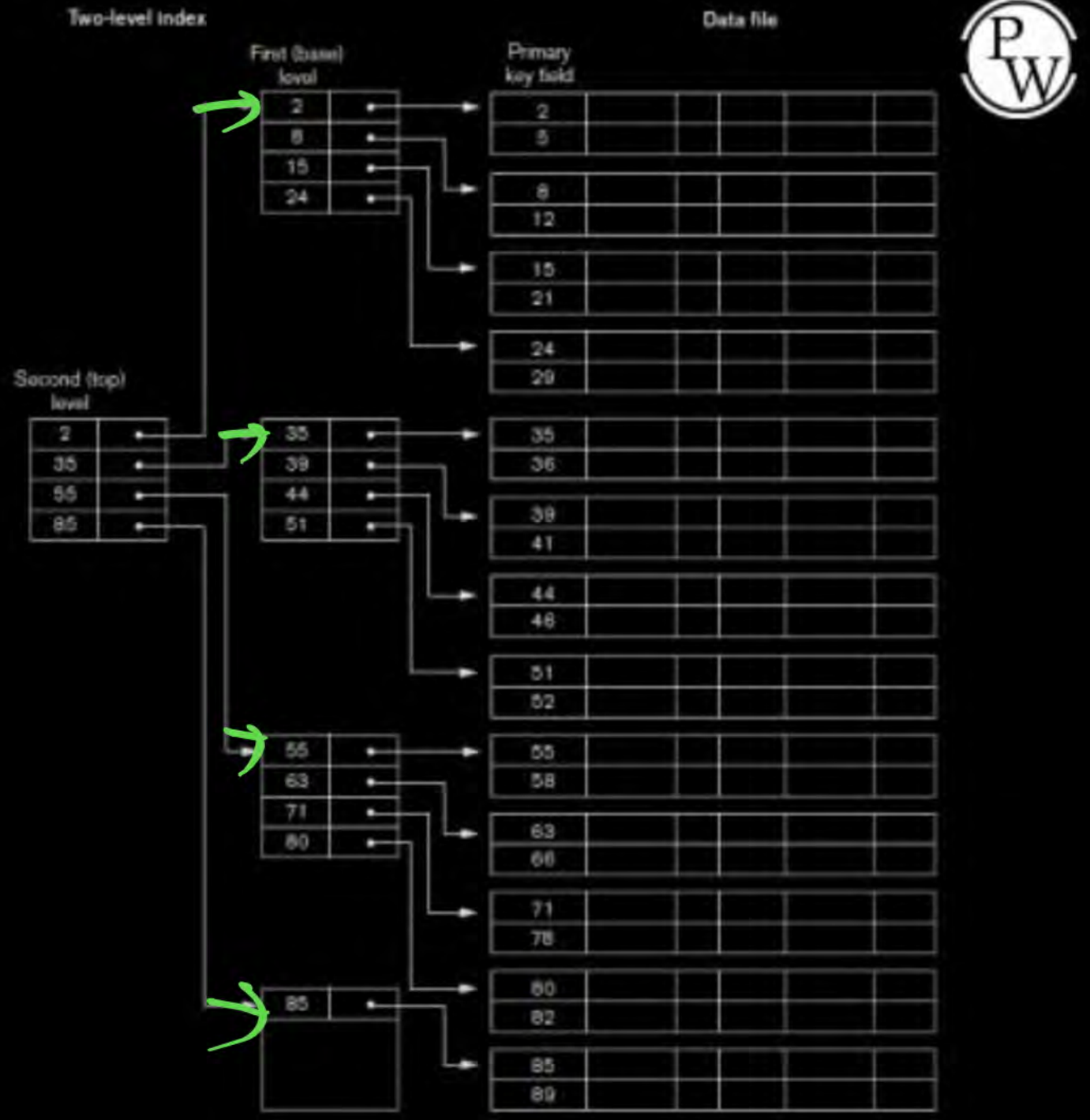


Multilevel Indexes

- Designed to greatly reduce remaining search space as search is conducted
- Index file
  - Considered first (or base level) of a multilevel index
- Second level
  - Primary index to the first level
- Third level
  - Primary index to the second level


A - G
H - P
Q - U
V - Z
A B C . . . G

A two-level primary index resembling ISAM (indexed sequential access method) organization



NOTE: We can Repeat the above process until index entries fit into One Block.

NOTE: If there are n level in multilevel index then the number of Block Access to search for a record = n + 1

(at each level One Index Block + 1 Data Block)

Q.3 Find the average number of block access required to search for a record if multilevel Index is created on the Data file of Question 2.

I^{st} Level # Index Block = 442.

Block factor of Index file = 68 Index Entries Per Block.

In multilevel

Now Apply SPARSE Index of Basic (I^{st} Level)

Q.3 Find the average number of block access required to search for a record if multilevel Index is created on the Data file of Question 2.

Block factor of Index file = 68 Index entries per Block

I[st] Level: Total number of Index Block = 442 Index Block

II[nd] Level: number of Index Records (entries) = 442 (SPARSE number of I[st] level block) & Block factor = 68 Index entries for block

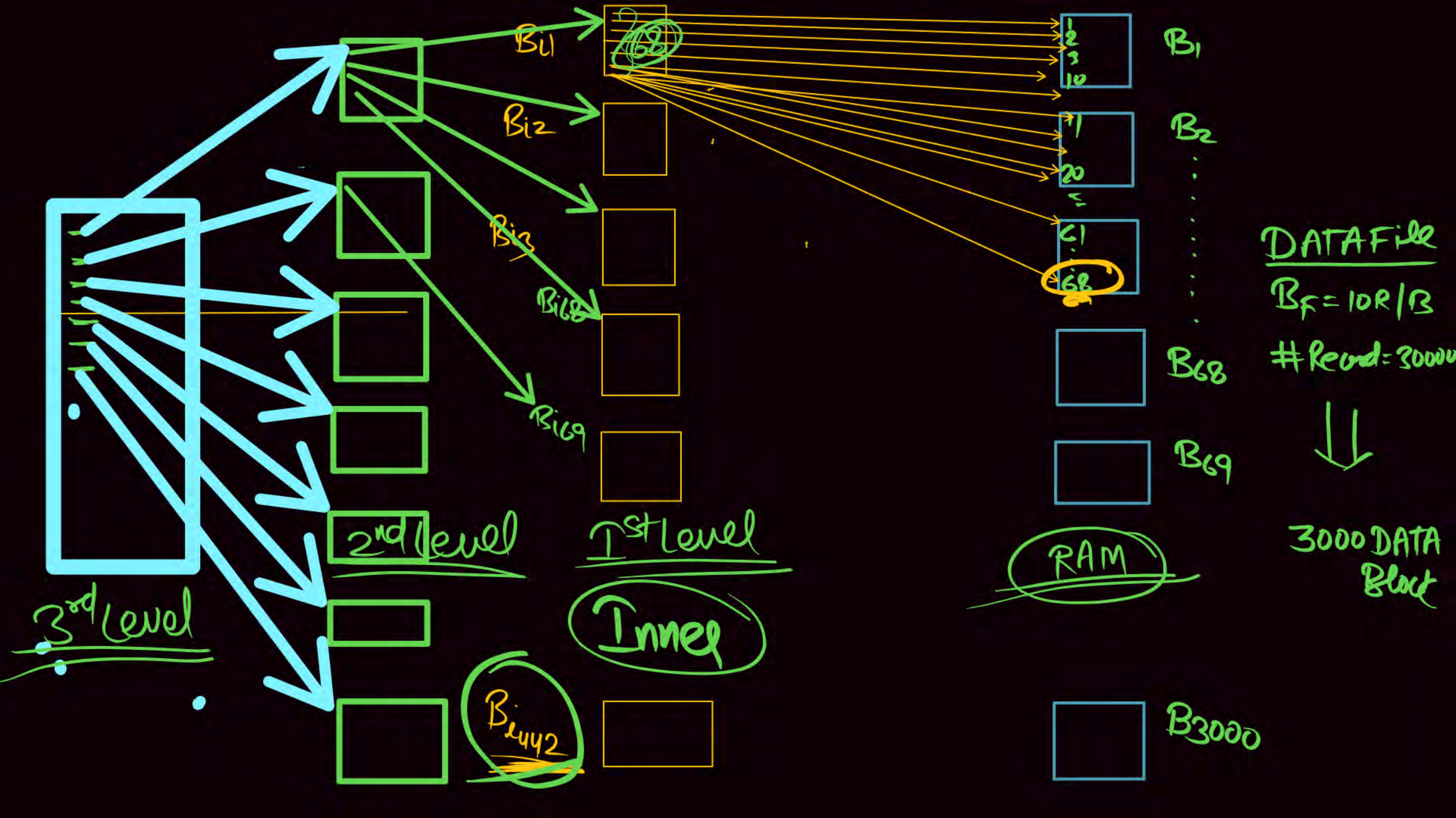
Total number Index Block = $\lceil \frac{442}{68} \rceil$ = 7 Index Block

III[rd] Level: Number of Index Record = 7 (number of 2[nd] level block)

Total number of index Block = $\lceil \frac{7}{68} \rceil$ = 1

Total 3 level required.

Average Number of block Access = 1 + 1 + 1 + 1 = 4


Bi1
Bi2
Bi3
Bi68
Bi69
Bi442
1st level
2nd level
3rd level
Index
B1
B2
B68
B69
B3000
RAM
DATAFile
BF = 10R/B
#Record = 30000
3000 DATA Block

Q.1 A clustering index is defined on the fields which are of type

[GATE-2008 : 1 Mark]

A. Non-key and ordering

B. Non-key and non-ordering (SI)

C. key and ordering (PI)

D. key and non-ordering (SI with key Attribute)

Q.2 Consider a file of 16384 records. Each record is 32 bytes long and its key field is of size 6 bytes. The file is orderd on a non-key field, and the file organization is unspanned. The file is stored in a file system with block size 1024 bytes, and the size of block pointer is 10bytes. If the secondary index is built on the key field of the file, and a multilevel index scheme is used to store the secondary index, the number of first-level and second-level block in the multilevel index are respectively [GATE-2008 : 2 Marks]

(A) 8 and 0

(B) 128 and 6

(C) 256 and 4

(D) 512 and 5

THANK YOU!