COMPUTER SCIENCE

Database Management System

File Org. & Indexing

Lecture_2

Vijay Agarwal sir

01 File Structure

02 Indexing & its Type

File Org.

① SPANNED ORG.

② UNSPANNED ORG.

① ORDERD File $[\log_2 B]$

② Unordered File/ Heap $[\frac{B}{2}, B]$

Indexing

One Index Record Size = Size of Search key + Bp.

$\log_2 B_i + 1$

① Dense

#Index entries = #DB Records.

② SPARSE

#Index entries = #DB Block

① Primary Index (P.K + ordered file)

② Clustering Index (Non key + ordered file)

③ Secondary Index ($\frac{\text{Non key}}{\text{Cand. key}}$ + Unordered file)

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]

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

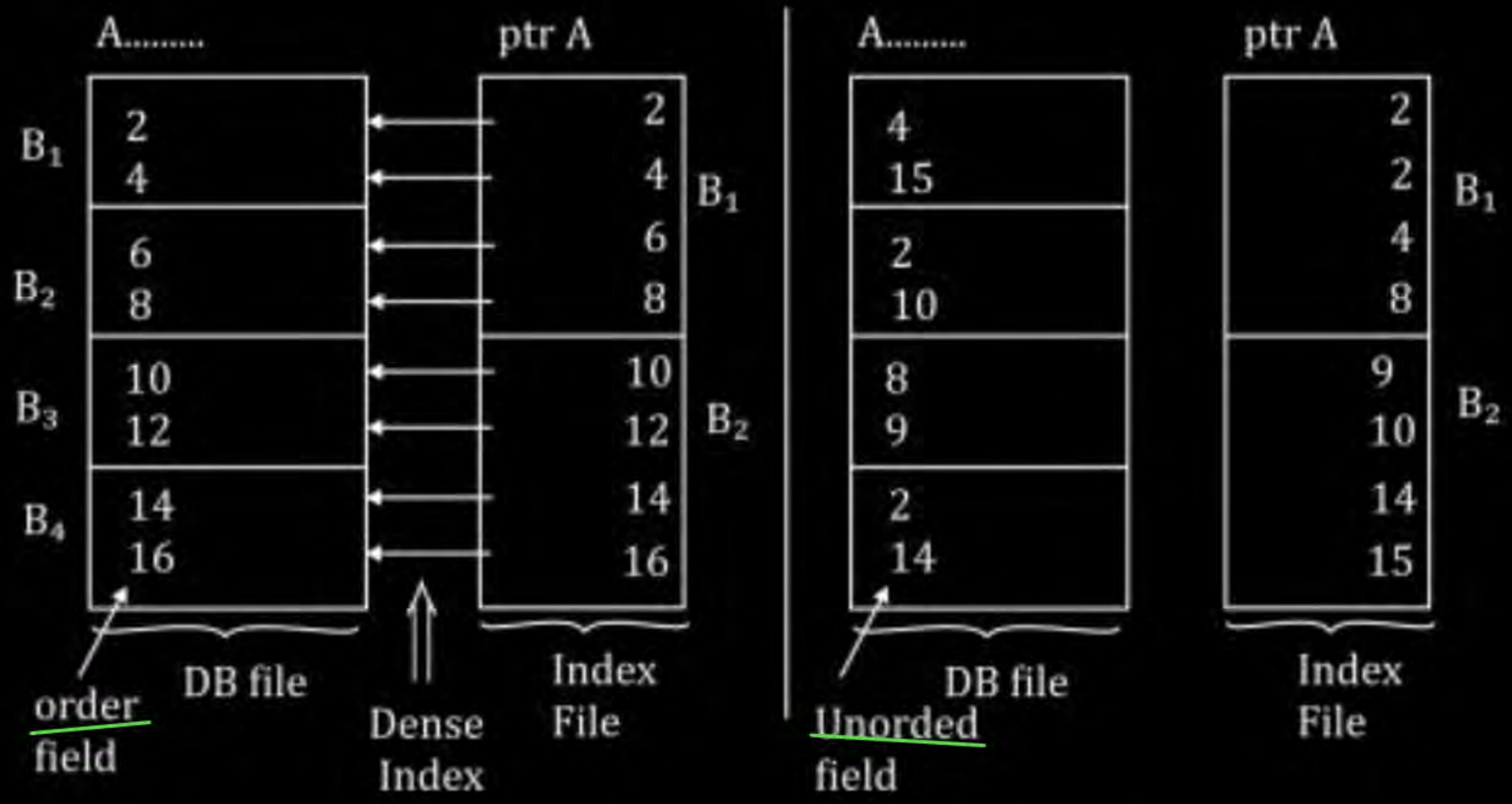
- Dense Index - Index record appears for every search-key values in the file.
- Example - index on *ID* attribute of *instructor* relation

10101	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	

Dense Index [More entries in Index File]

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

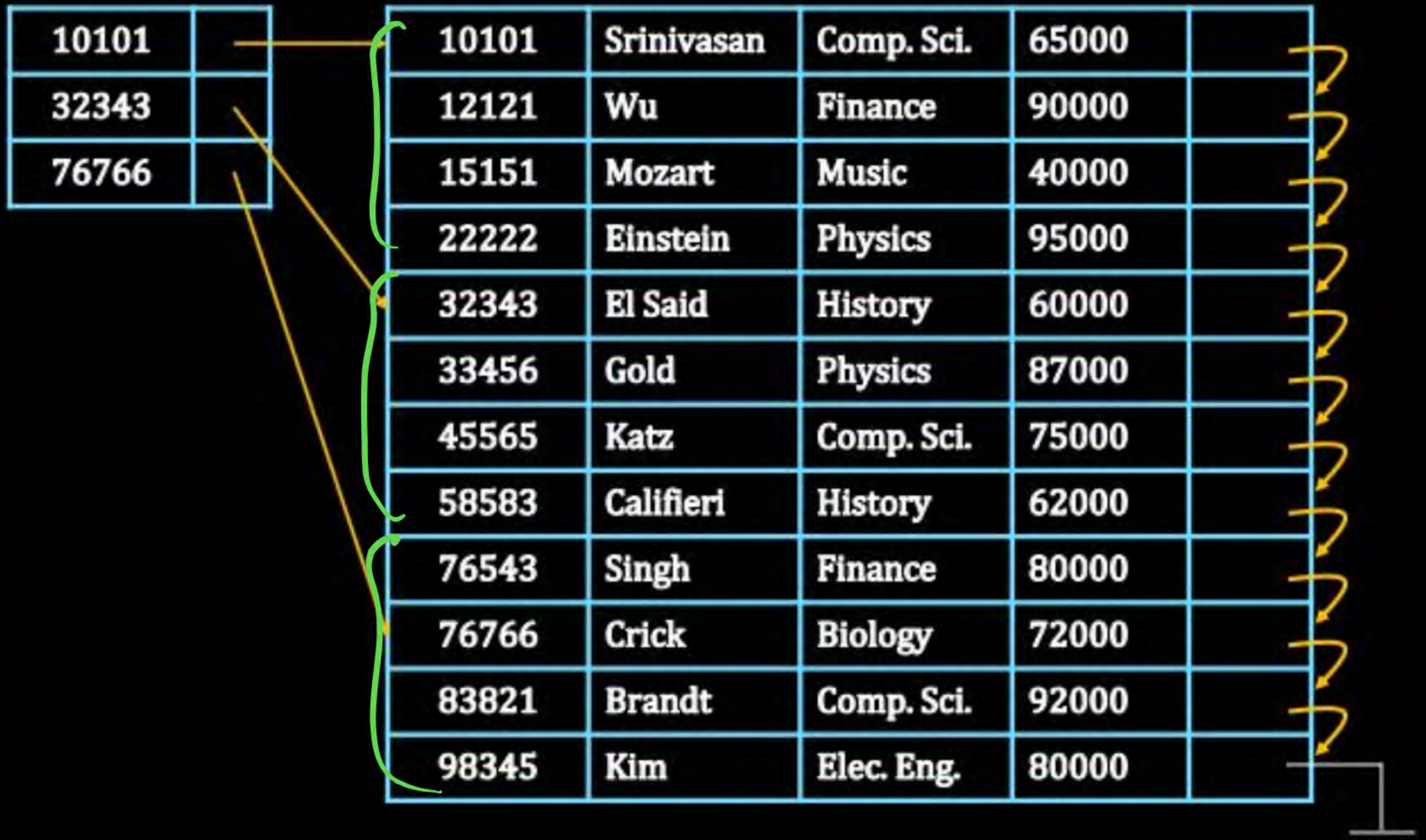
[Dense Index]



⇒ (# 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]

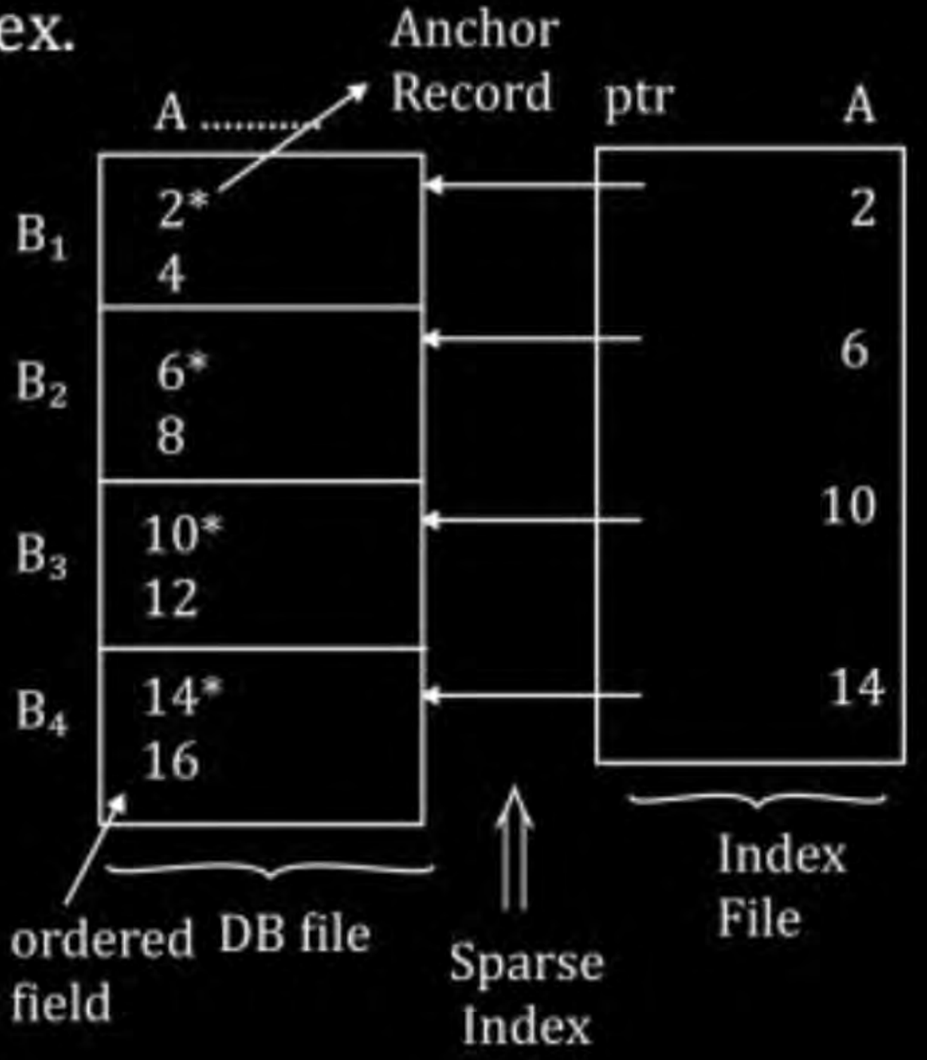
⇒ For set of DB records there exist entry in Index file such a Index is called Sparse Index.

⇒ 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]

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



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

Types of Index

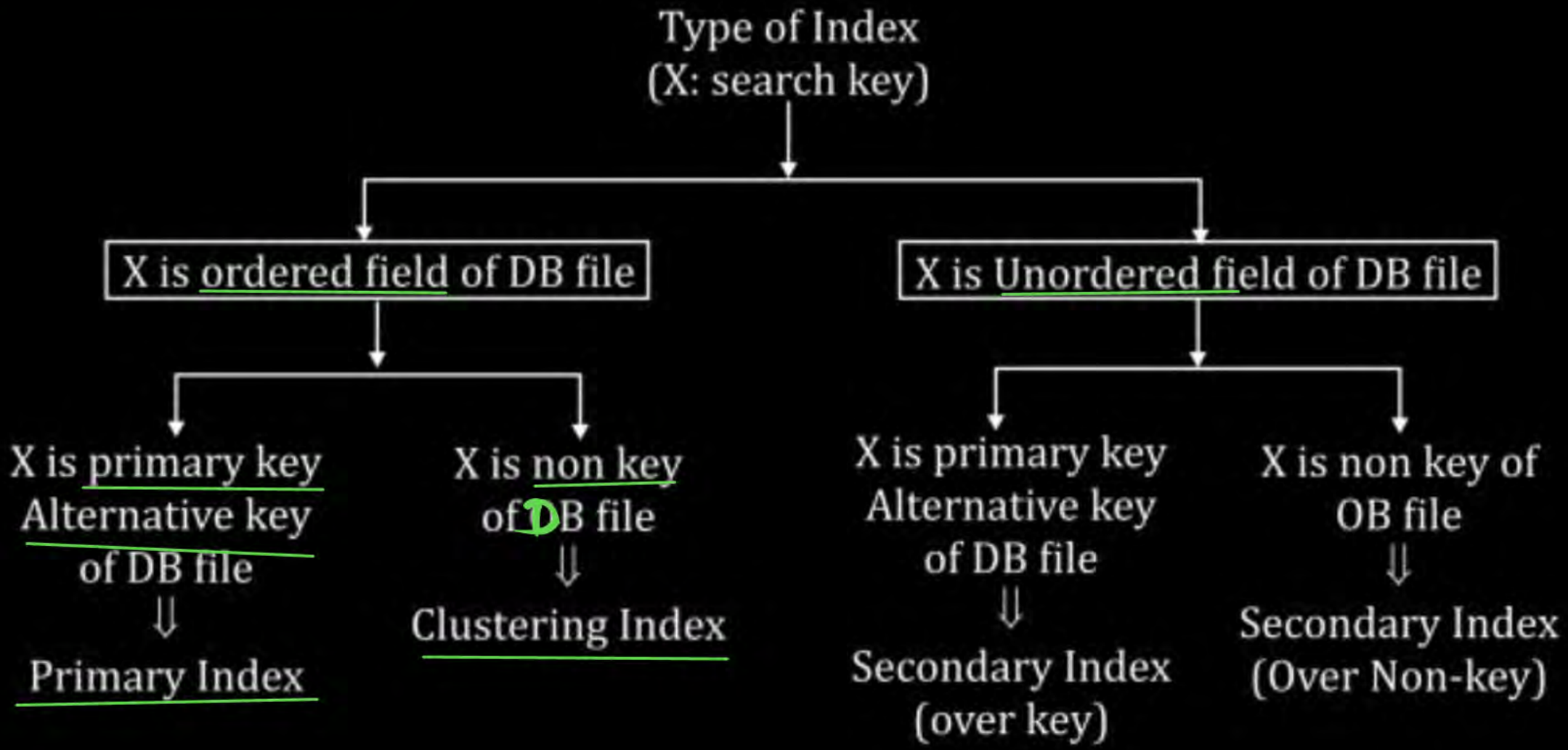
Single-level Ordered Indexes

1. Primary indexes
2. Clustering indexes
3. Secondary indexes

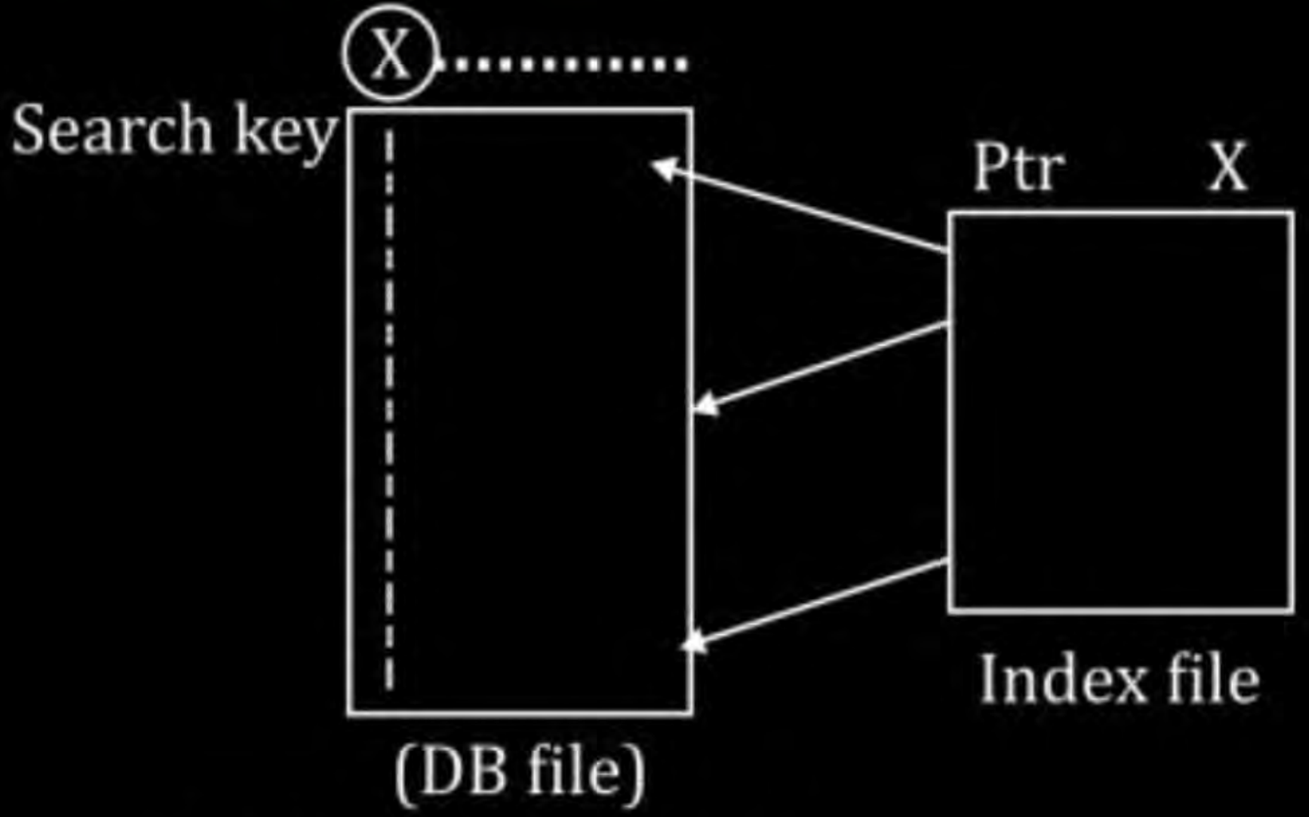
Multilevel Indexes

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

Types of Index



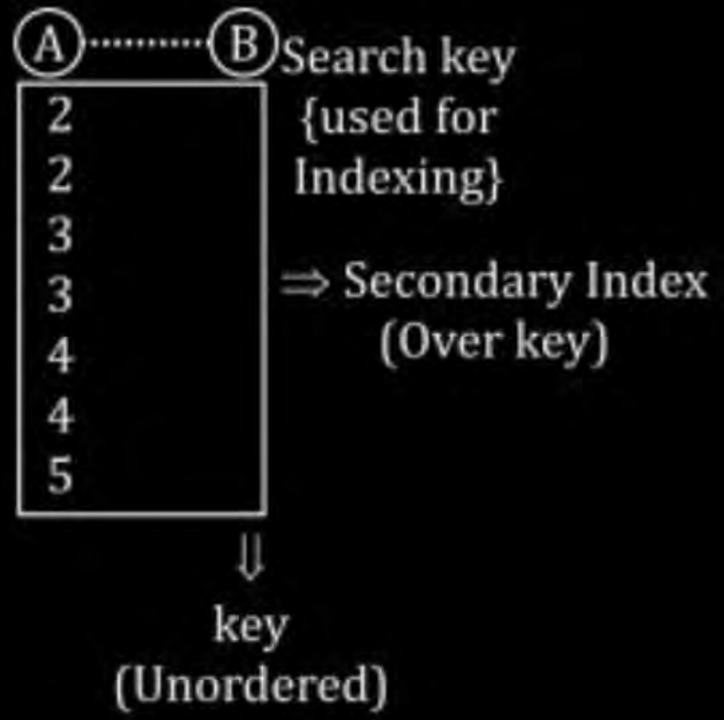
Types of Index



{Ordered Unordered
key/Non-key}

Types of Index

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

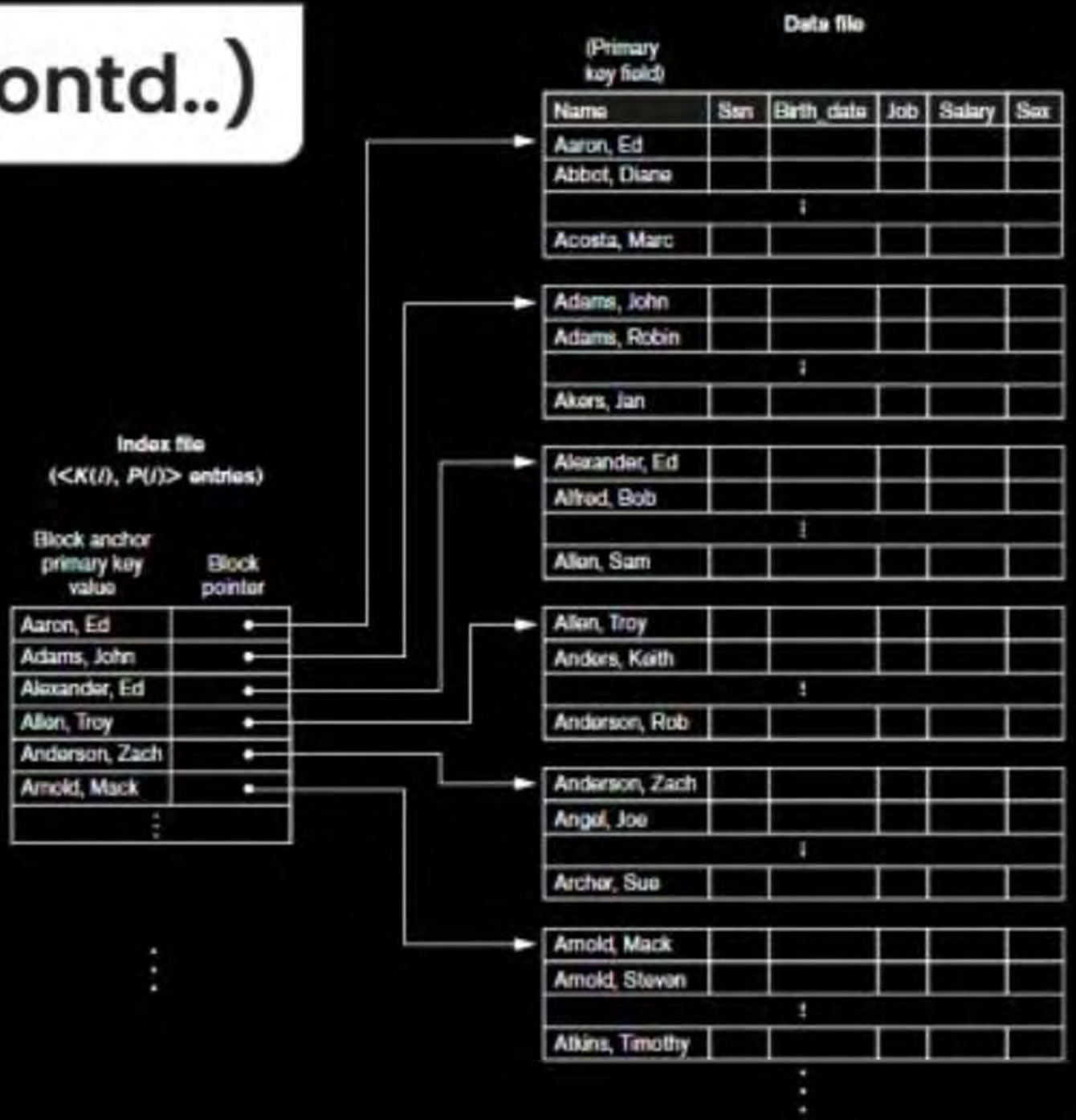


Primary Indexes

- 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

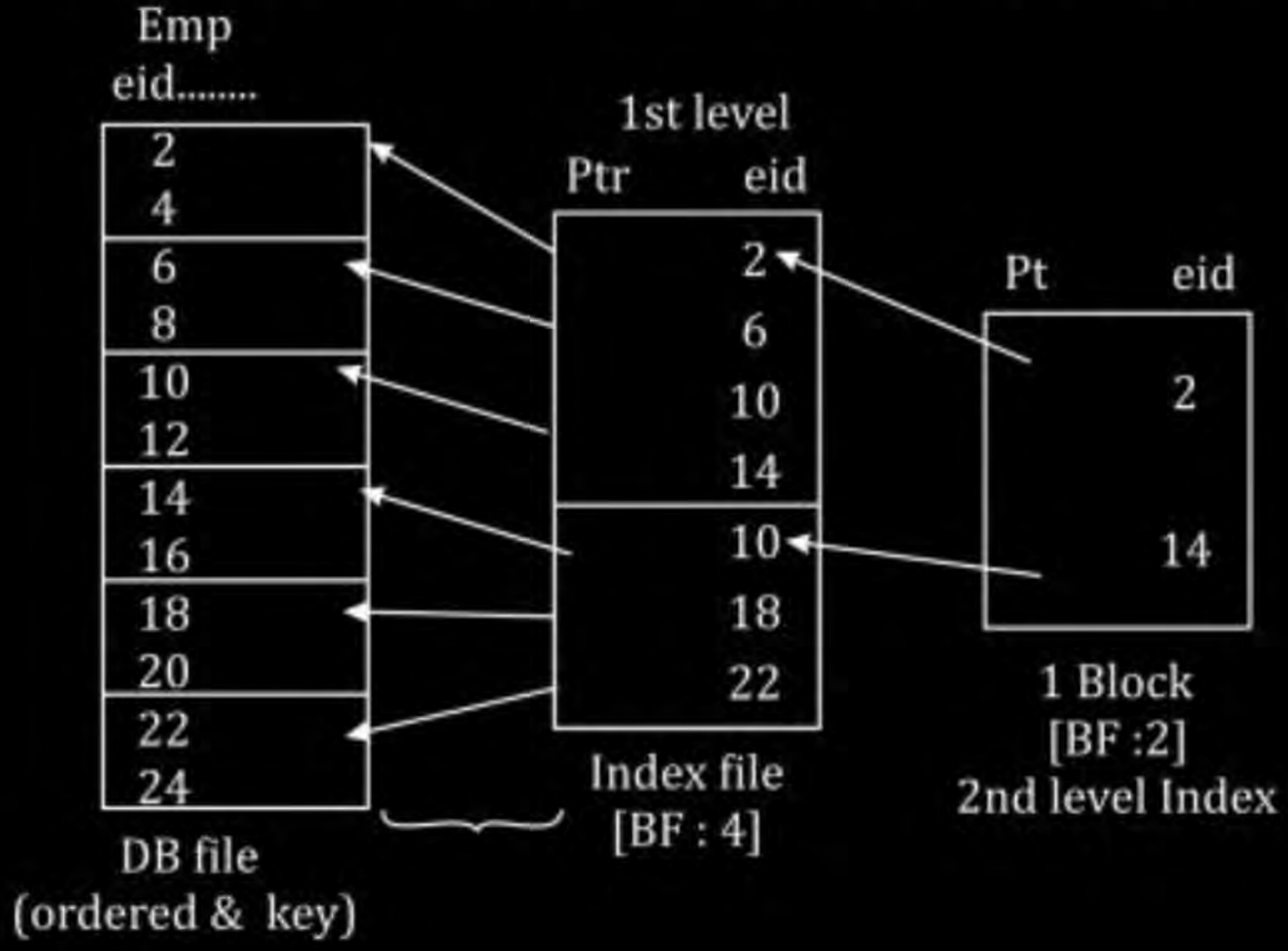
Primary Index (Contd..)

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

- Primary Index can be Dense or sparse

  [sparse PI can be preferred]

- For any database relation at most one PI is possible

  [because of Index over ordered field]

#Record = 30,000 Block Size = 1024 Byte, Record Size = 100 B Unspanned Ordered File

Key = 9 Byte & Bp = 6 Byte Primary Index

Without Index :

Blocking factor of DB File [Unspanned] $= \left\lfloor \frac{\text{Block size}}{\text{Record size}} \right\rfloor = \left\lfloor \frac{1024\,B}{100\,B} \right\rfloor =$ 10 Record per Block

Yogesh 2049

30002

Total Number of Record = 30,000.

Number of Data Block [B] $= \left\lceil \frac{30,000}{10} \right\rceil =$ 3000 Data Block

ORERED File

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

$2^{10} = 1024$

$2^{11} = 2048$

$2^{12} = 4096$

#Record = 30,000 Block Size = 1024 Byte, Record Size = 100 B Unspanned Ordered File

Key = 9 Byte & Bp = 6 Byte Primary Index

With Index: One Index Record Size = 9 + 6 = **15 Byte**

Blocking factor of Index File $[BF_i] = \left\lfloor \frac{\text{Block Size}}{\text{Record size}} \right\rfloor = \left\lfloor \frac{1024B}{15B} \right\rfloor = 68$ Index entries per Block

P.I: **SPARSE** Total # Index Entries = 3000 [# DB Blocks]

Total Number of Index Block $[B_i] = \left\lceil \frac{3000}{68} \right\rceil = \lceil 44.11 \rceil \Rightarrow 45$ Index Block

Note: If we take 44 $\Rightarrow$ 44 X 68 = 2992 Block $\Rightarrow$ 8 DB Block Missing $\Leftarrow$ 8 X 10 $\Leftarrow$ 80 Record Missing.

To Access Index Block Avg # Block Access $= \lceil \log_2 B_i \rceil = \lceil \log_2 45 \rceil = 6$ Block Access

To Access (Fetch) the Record using Indexing Avg # Block Access $= \lceil \log_2 B_i \rceil + 1 \Rightarrow 6 + 1 = 7$ Block Access

#Record = 30,000 Block Size = 1024 Byte, Record Size = 100 B Unspanned Ordered File

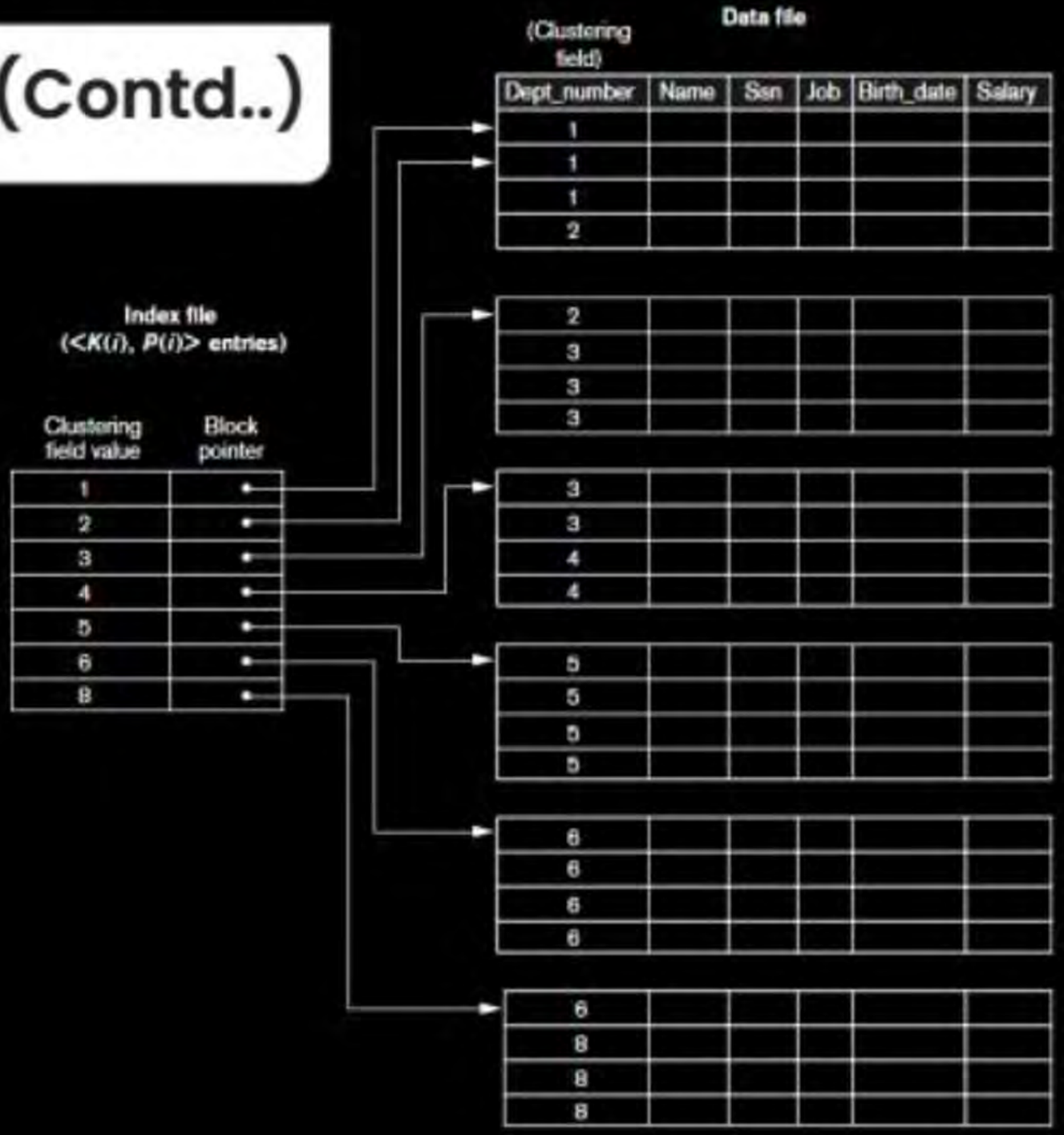
Key = 9 Byte & B_p = 6 Byte Primary Index

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

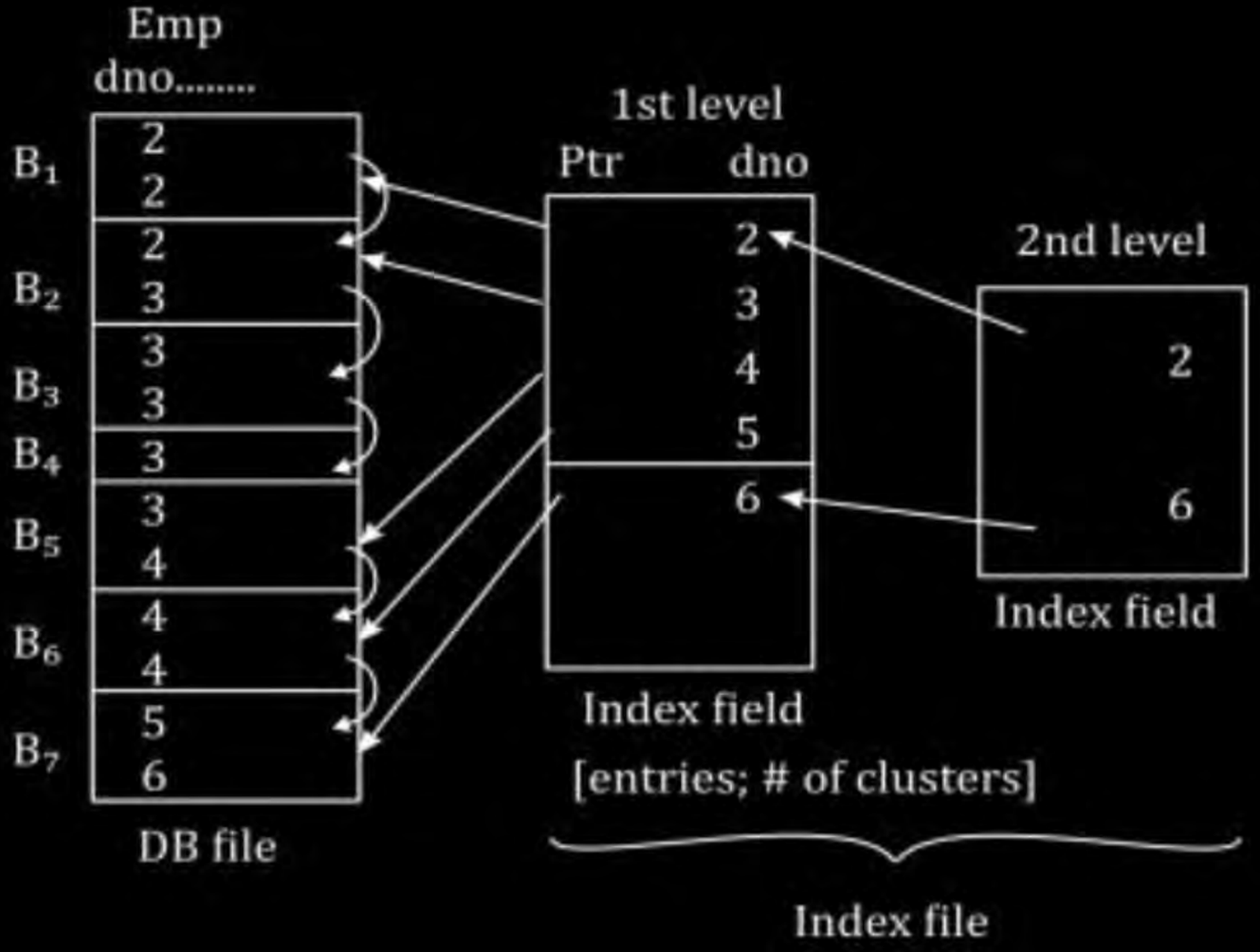
Clustering Indexes (Contd..)

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



Clustering Index:

Search key: Ordered field & Non- key.

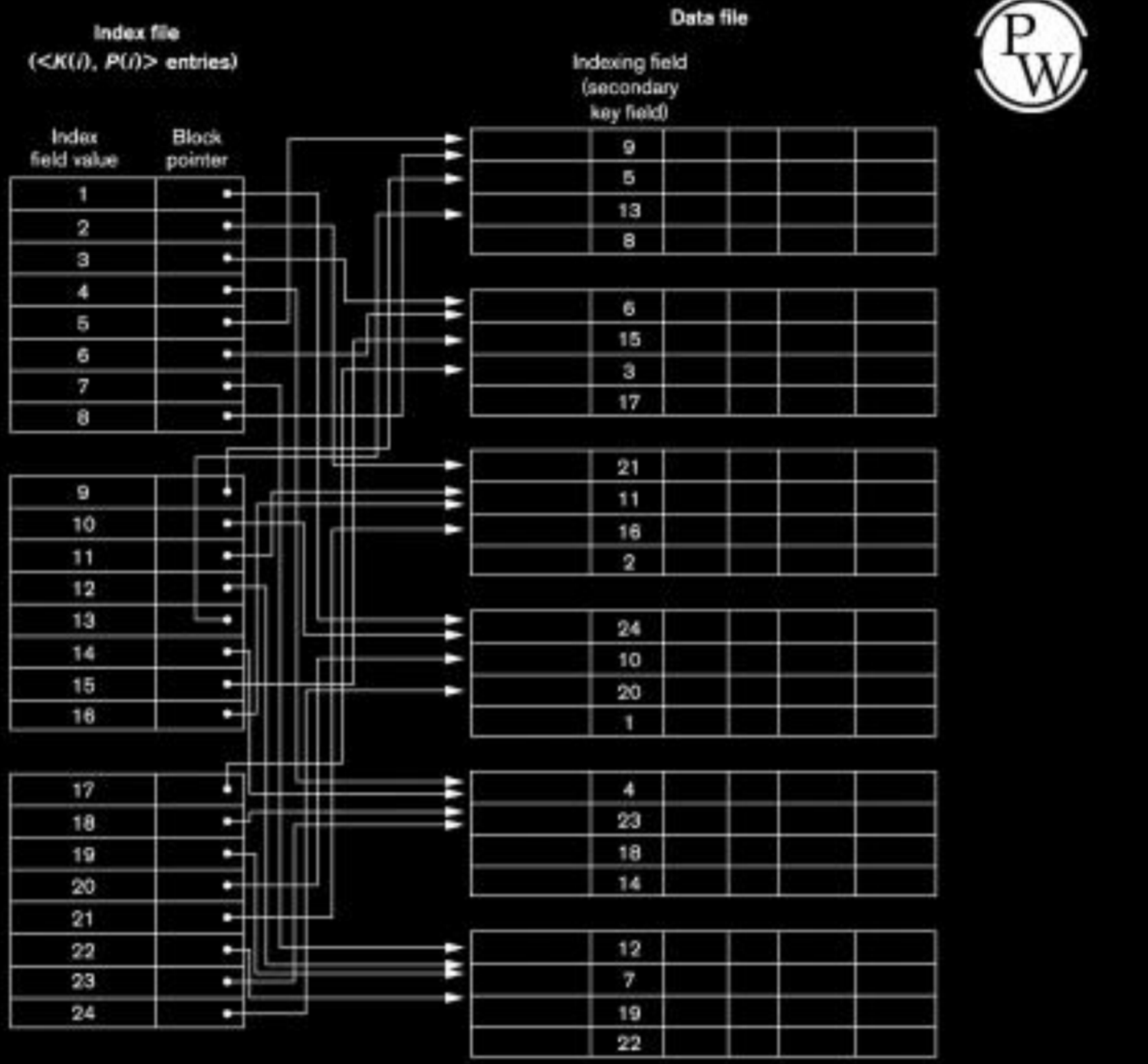


- Clustering Index mostly sparse Index [Dense CI also possible If each cluster with one record]
- At most one CI is possible for any Database relation [ordering required]
- For any DB relation can build either PI or CI but not both.

Secondary Indexes

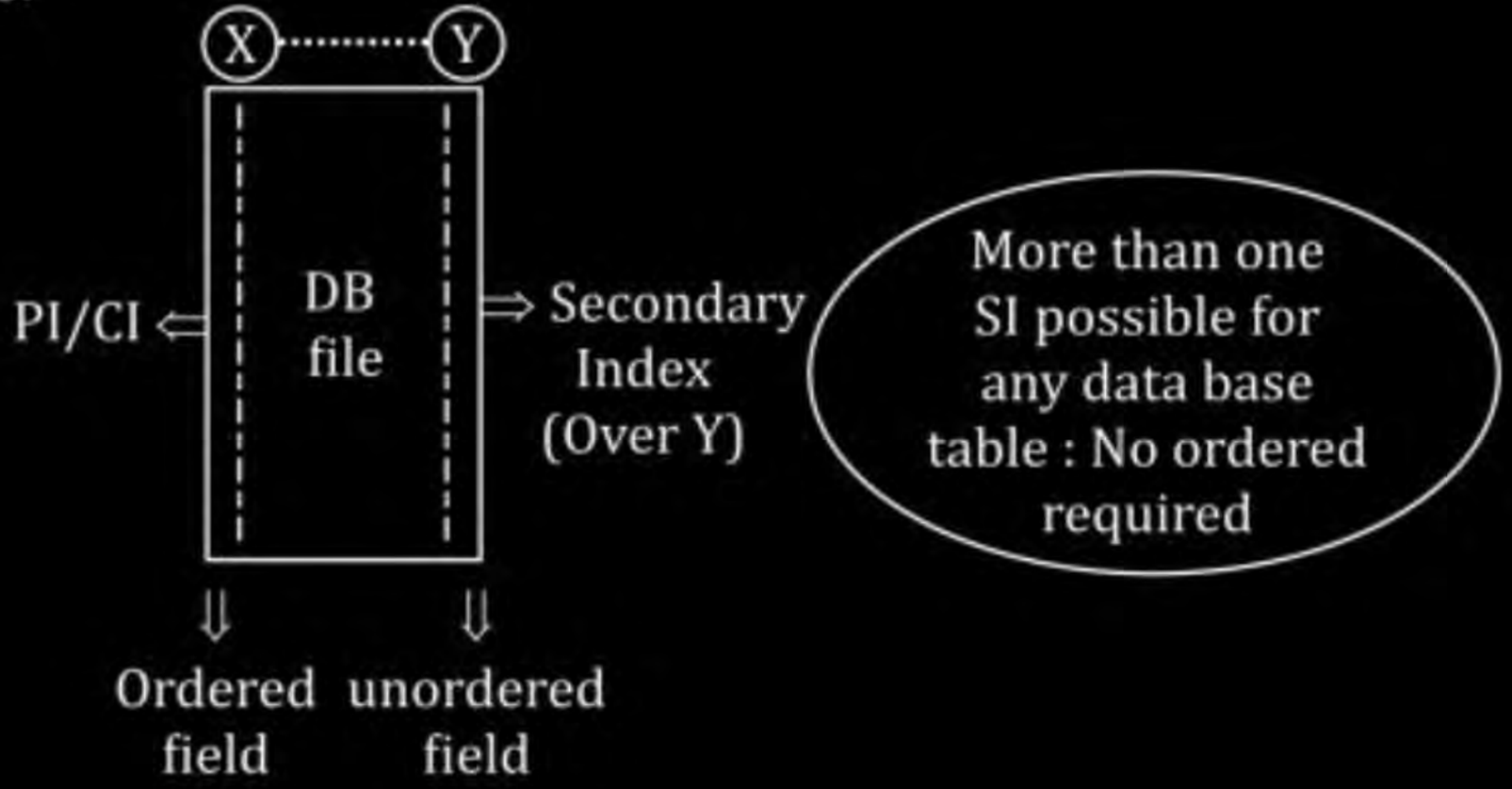
- Secondary Index
  - A secondary index provides a secondary means of accessing a file for which some primary access already exists.
  - The secondary index may be on a field which is a candidate key and has a unique value in every record, or a non-key with duplicate values.
  - The index is an ordered 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..)

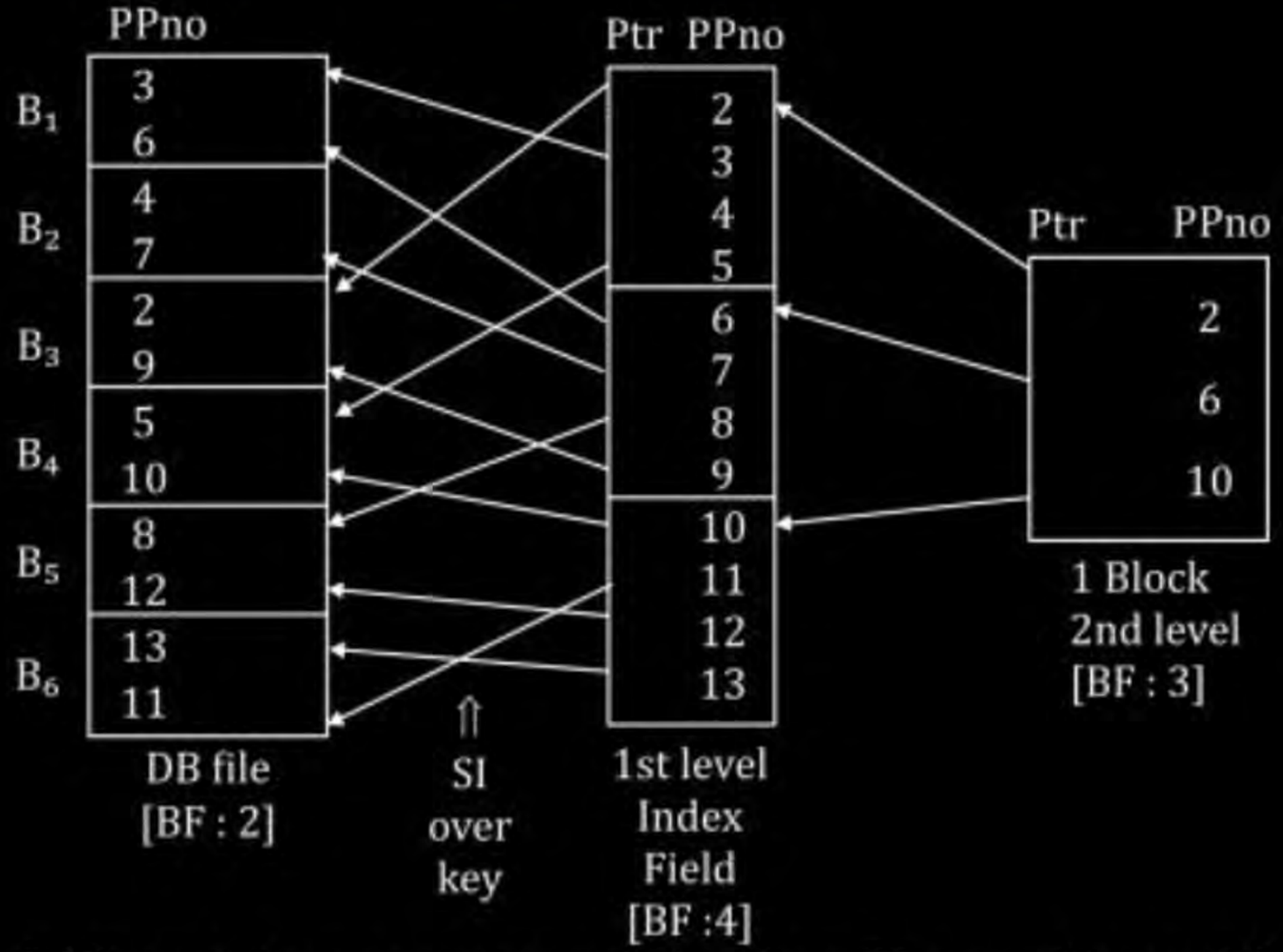


Secondary Index:

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

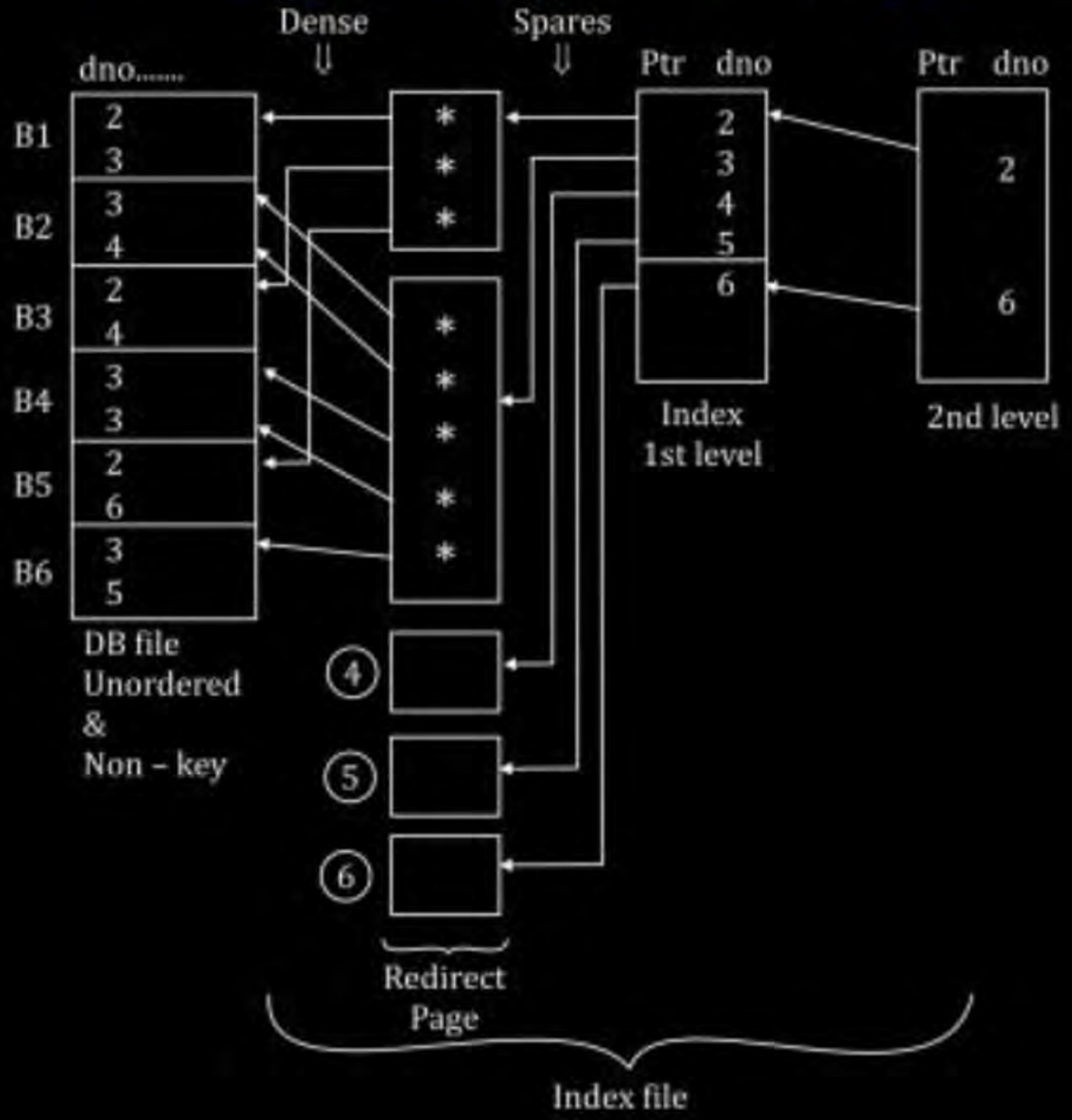


- Secondary Index (over key):



- 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?

Unordered file & SI (Dense)

Number of records = 30000 Block size = 1024 B

record size = 100 B

Key = 9B Bp = 6 Byte

Unspanned & Unordered.

One Index Record Size = 9 + 6 = 15 Byte

#Record = 30,000 Block Size = 1024 Byte, Record Size = 100 B Unspanned Unordered File

Key = 9 Byte & Bp = 6 Byte Secondary Index

Without Index :

Blocking factor of DB File [Unspanned] $= \left\lfloor \frac{\text{Block Size}}{\text{Record Size}} \right\rfloor = \left\lfloor \frac{1024\,B}{100\,B} \right\rfloor =$ 10 Record per Block

Yogesh 2049 ↑ 3000?

Total Number of Record = 30,000.

Number of Data Block [B] $= \left\lceil \frac{30,000}{10} \right\rceil =$ 3000 Data Block

UNORERED File

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

Worst Case = B = 3000 Ans

#Record = 30,000 Block Size = 1024 Byte, Record Size = 100 B Unspanned Ordered File

Key = 9 Byte & Bp = 6 Byte Primary Index

With Index: One Index Record Size = 9 + 6 = **15 Byte**

Blocking factor of Index File $[BF_i] = \left\lfloor \frac{\text{Block Size}}{\text{Record size}} \right\rfloor = \left\lfloor \frac{1024B}{15B} \right\rfloor = 68$ Index entries per Block

SI: **Dense** Total # Index Entries = 30000 [# DB Records]

Total Number of Index Block $[B_i] = \left\lceil \frac{30000}{68} \right\rceil = \lceil 441.17 \rceil \Rightarrow$ 442 Index Block

(Note)

To Access Index Block Avg # Block Access $= \lceil \log_2 B_i \rceil = \lceil \log_2 442 \rceil = 9$ Block Access

To Access (Fetch) the Record using Indexing Avg # Block Access $= \lceil \log_2 B_i \rceil + 1 \Rightarrow 9 + 1 = 10$ Block Access.

in previous eg:

PI : SPARSE ⇒ # Index Block [45]

SI : Dense ⇒ # Index Block [4448]

very very large.

442 Index Block is

very - very large

So Apply Multi Level Indexing.

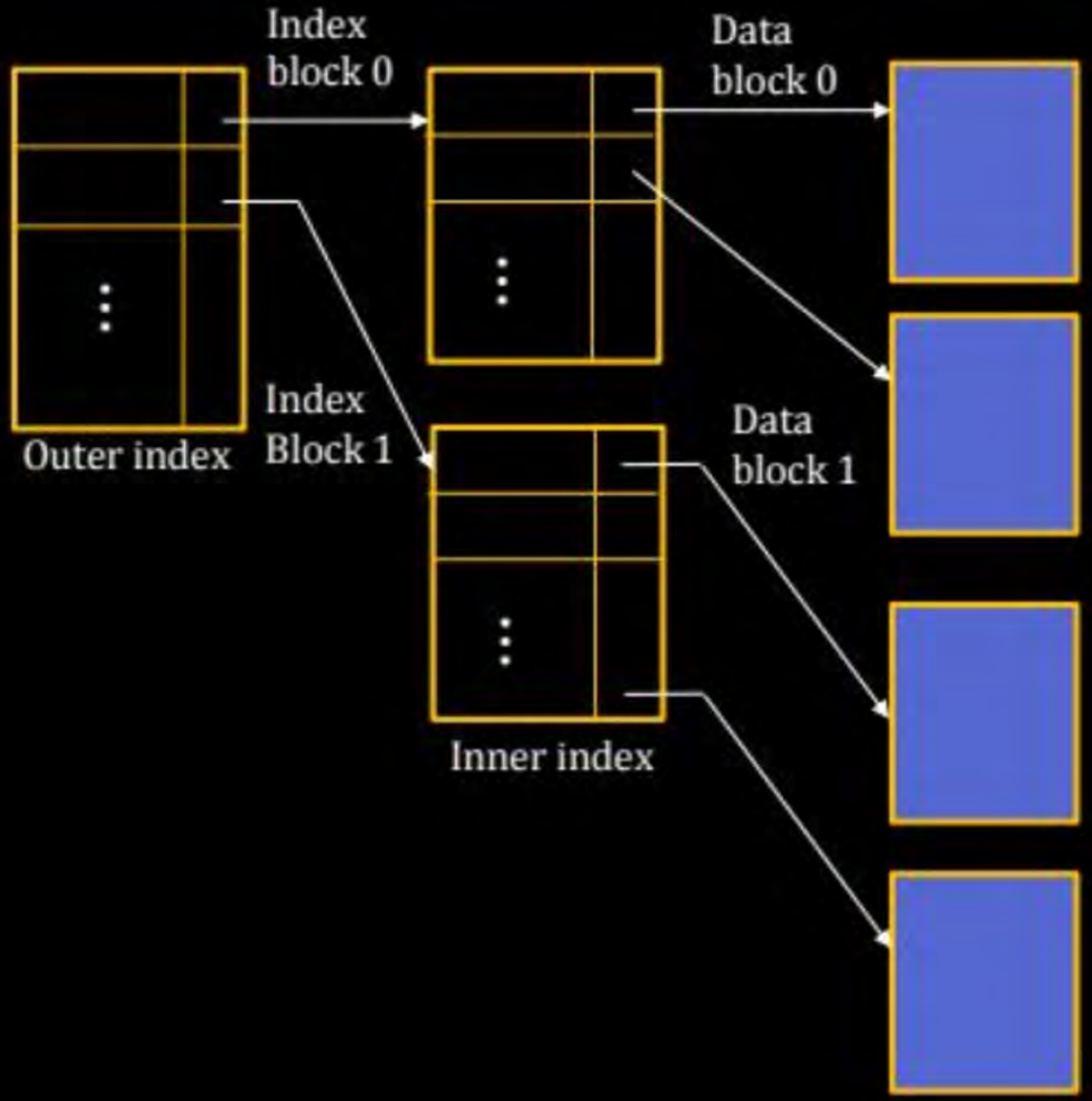
⇓

SPARSE

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.

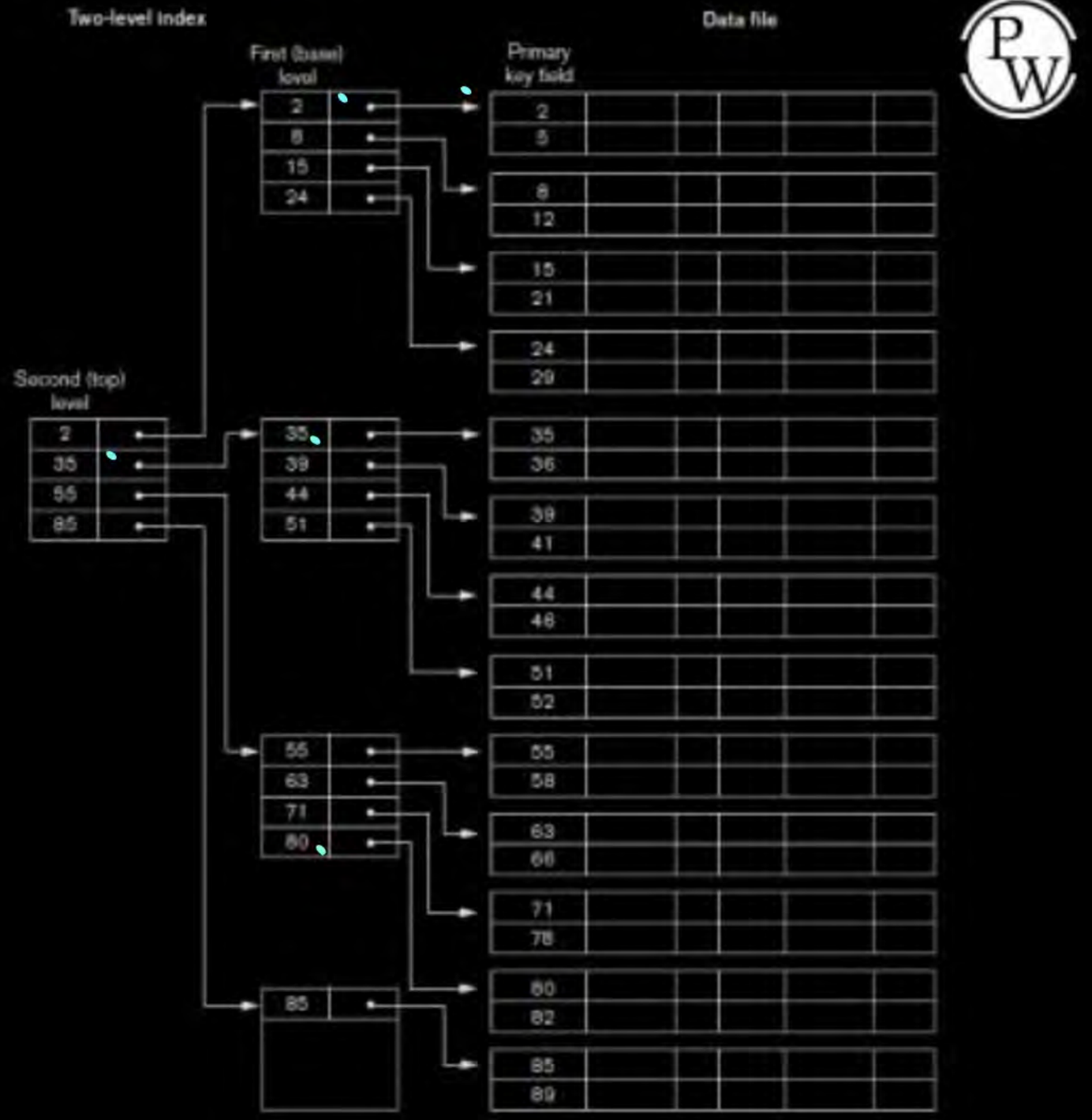
Multilevel Index (Contd..)



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 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.

1st Level { # Index Block = 442.

Blocking factor of Index file = 68 Index entries per Block.

IInd Level :

Total (#) Number of Index Entries = 442 (# Ist Level Block)

Block factor of Index file = 68 Index entries per Block

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

IIIrd Level

Total (#) Number of Index entries = 7 (# 2nd Level Block)

Total # Index Block = $\lceil \frac{7}{68} \rceil$ = 1 Index Block (So Stop)

Total 3 Level Required.

To Access a Record Avg # Block Access = n+1 = 3+1 = (4) Ans

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

Ist Level: Total number of Index Block = 442 Index Block

IInd Level: number of Index Records (entries) = 442 (SPARSE number of Ist level block) & Block factor = 68 Index entries for block

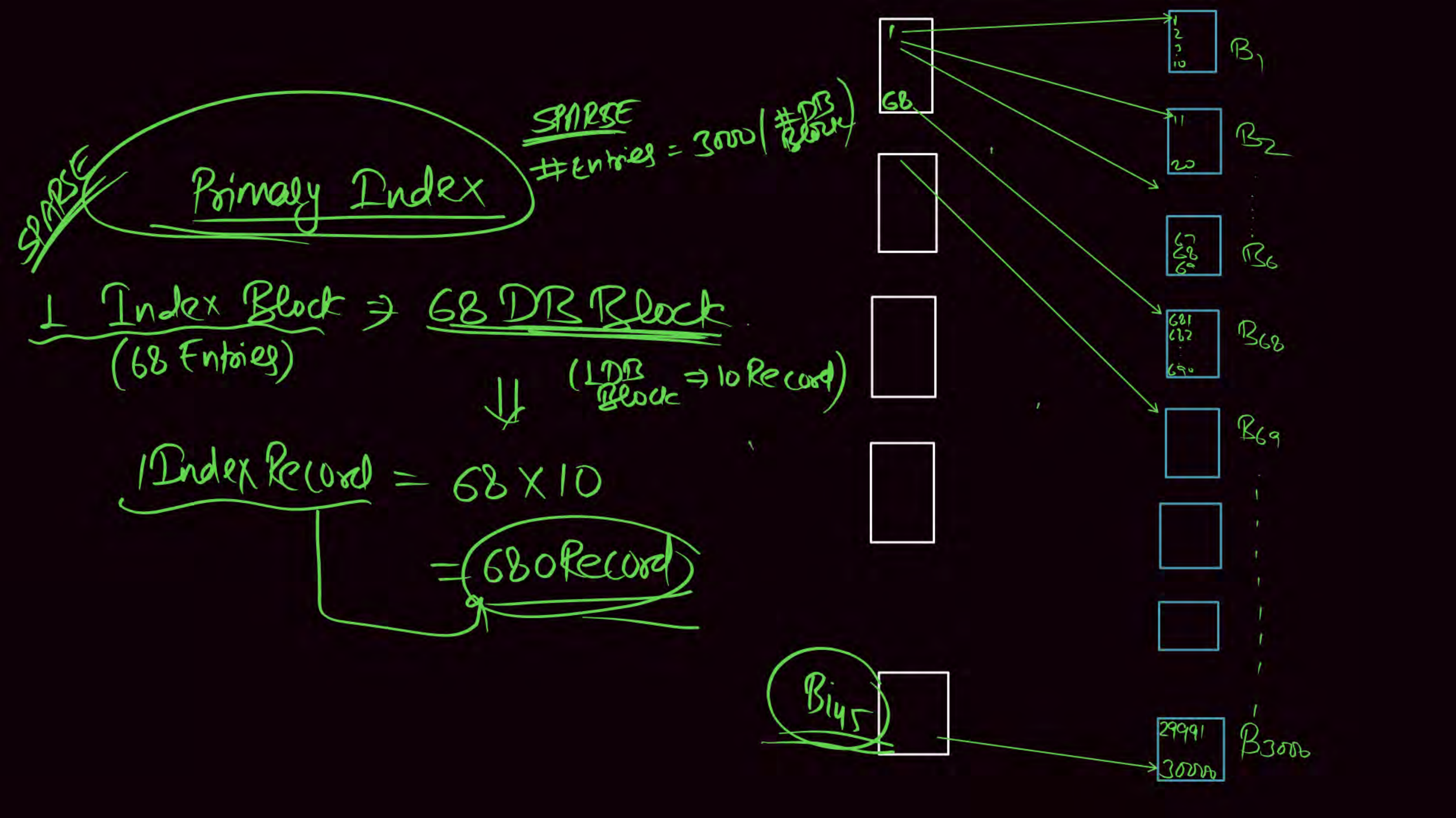
$$\text{Total number Index Block} = \left\lceil \frac{442}{68} \right\rceil = 7 \text{ Index Block}$$

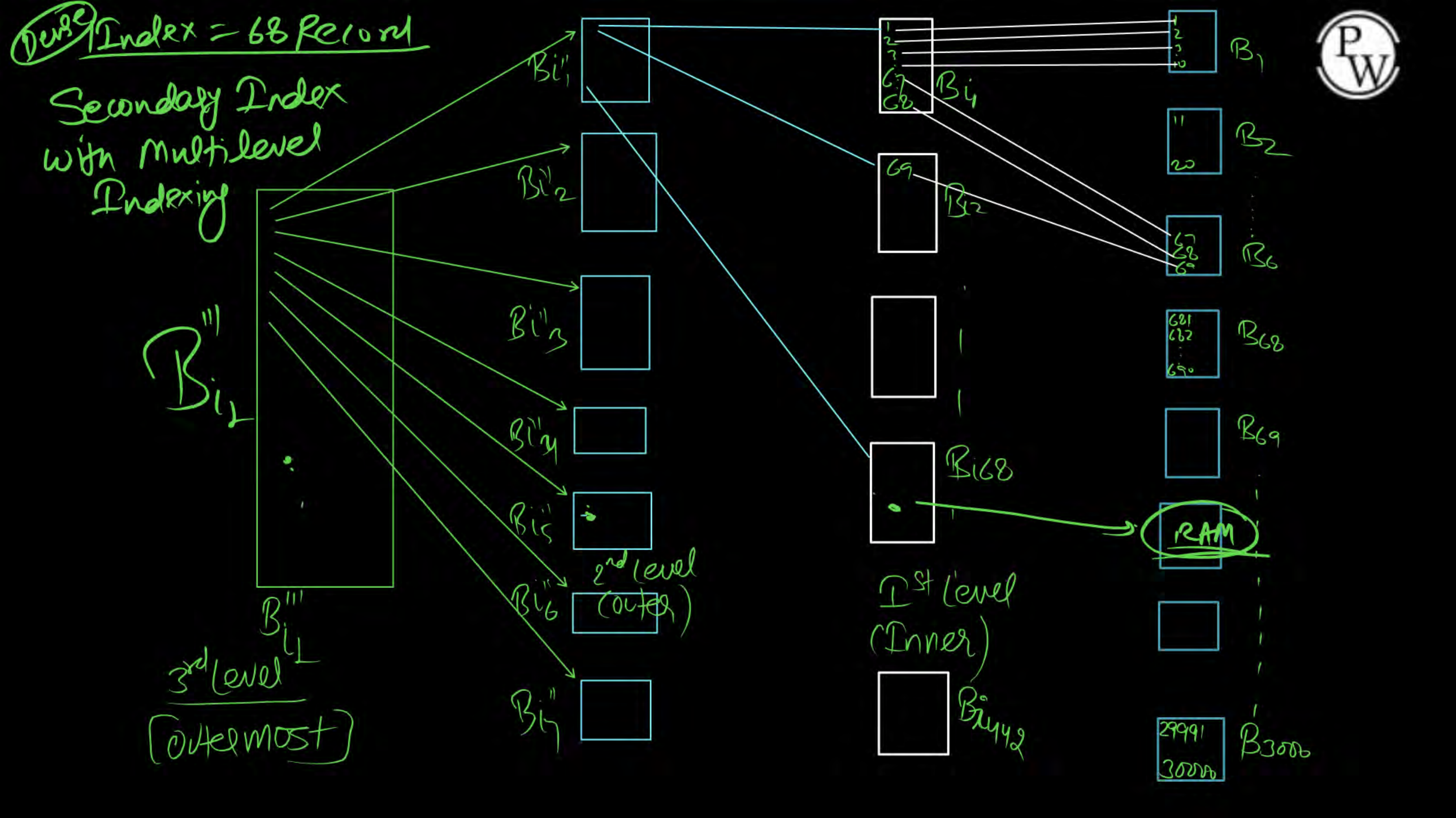
IIIrd Level: Number of Index Record = 7 (number of 2nd level block)

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

3rd 2nd 1st Data Block

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


SPARSE
Primary Index
SPARSE
#Entries = 3000 (# DB Blocks)
1 Index Block ⇒ 68 DB Block
(68 Entries)
(1 DB Block ⇒ 10 Record)
1 Index Record = 68 X 10
= 680 Record
1
68
1
2
3
10
11
20
67
68
69
681
682
690
B_1
B_2
B_6
B_{68}
B_{69}
B_{145}
29991
30000
B_{3000}

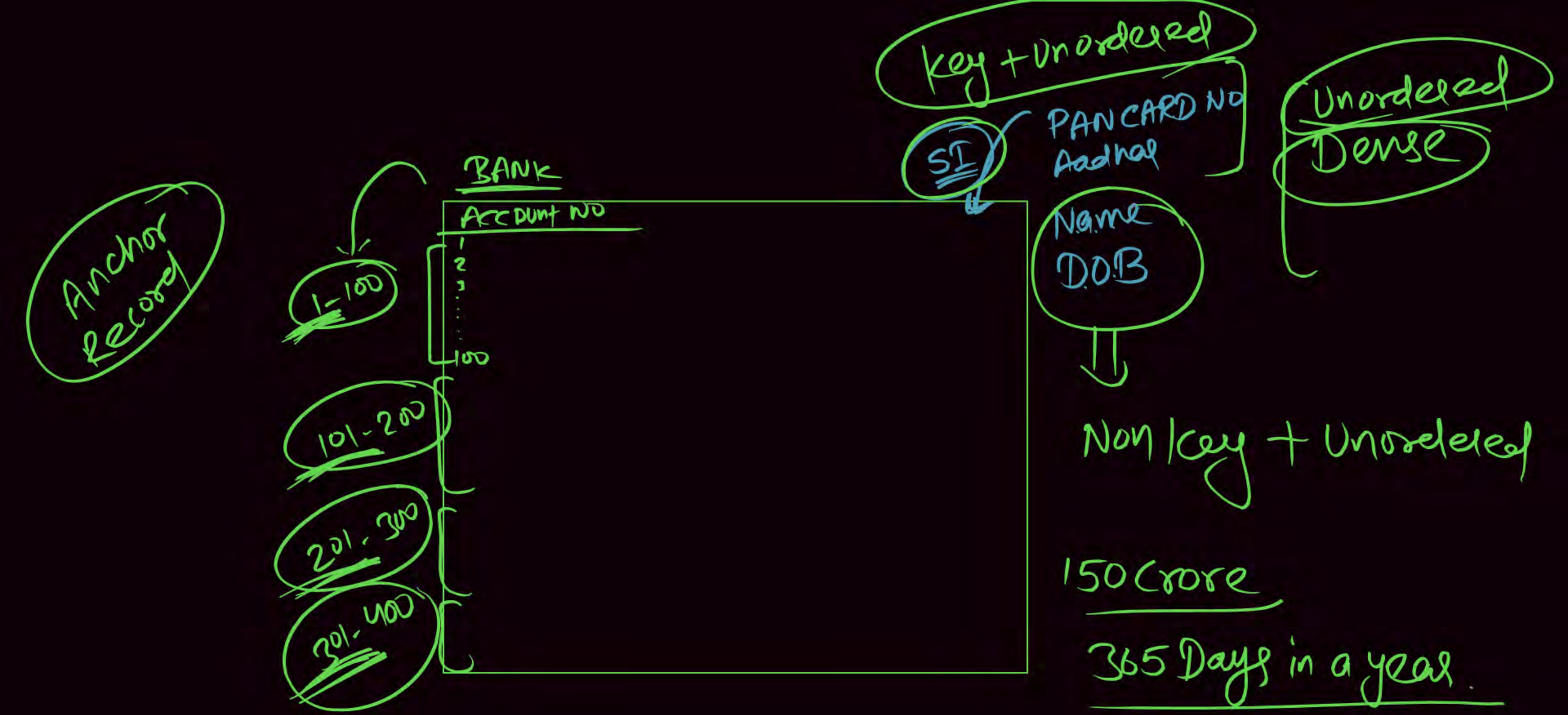

Dense Index = 68 Record
Secondary Index with Multilevel Indexing
B_{i_1}''
B_{i_1}'''
3rd level (Outermost)
2nd level (outer)
1st level (Inner)
RAM
PW

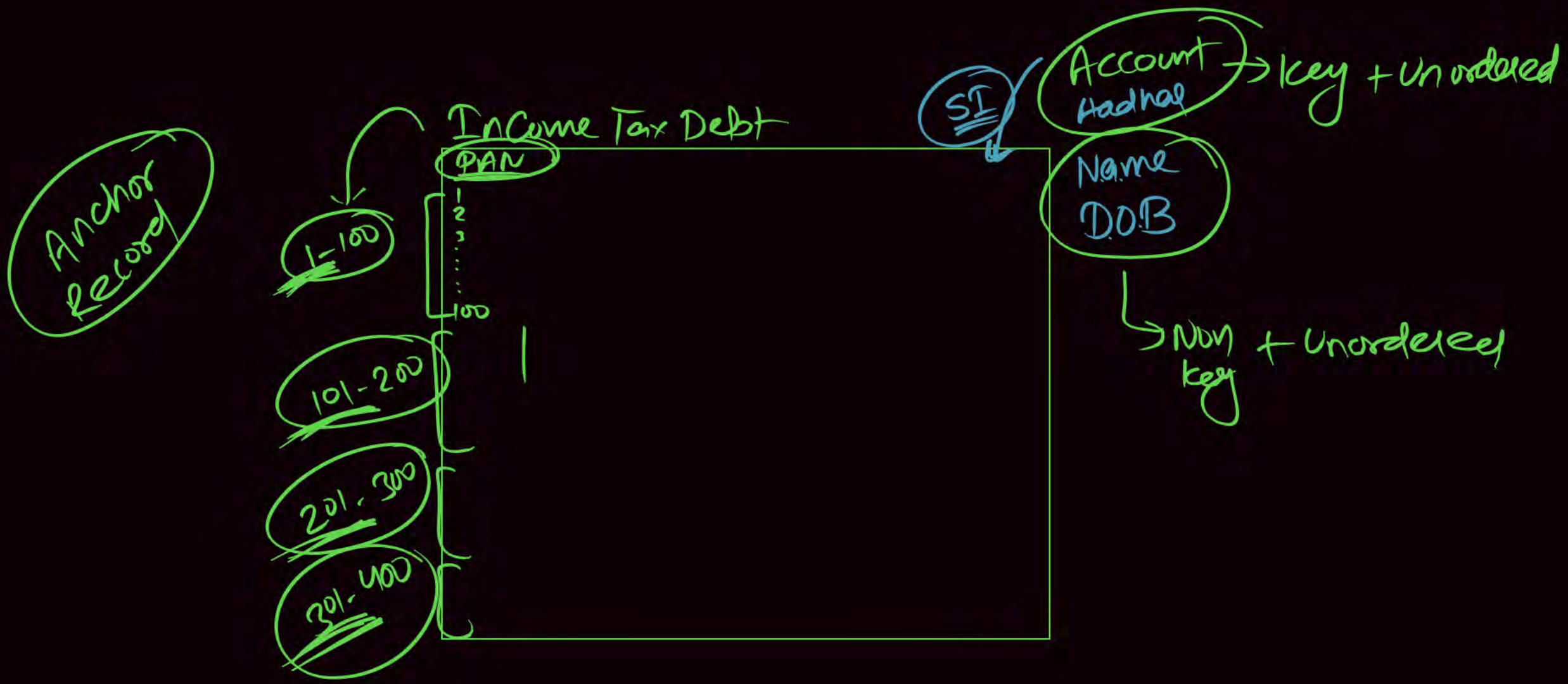
PI ✓ CI X

CI ✓ PI X

PI ✓ SI ✓

CI SI ✓


Key + Unordered
SI
PANCARD NO
Aadhar
Unordered
Dense
BANK
ACCOUNT NO
Name
D.O.B
Anchor record
1-100
101-200
201-300
301-400
Non key + Unordered
150 Crore
365 Days in a year


Anchor record
Income Tax Debt
PAN
1
2
3
100
1-100
101-200
201-300
301-400
SI
Account
Hadhar
key + Unordered
Name
DOB
Non key + Unordered

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

[GATE-2008 : 1 Mark]

(A) Non-key and ordering (Clustering Index)

(B) Non-key and non-ordering [SI with non key]

(C) key and ordering [PI]

(D) key and non-ordering [SI with key]

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

Total #Record = 16,384 [2^{14}] Key = 6 Byte BP = 10 Byte, Block Size = 1024 Byte

One Index Record Size = Size of Key + Size of BP = 6 + 10 = 16 Byte.

Block factor of Index file = $\frac{1024B}{16B} = \frac{2^{10}}{2^4} = 2^6$ → 64 Index Entries Per Block.

SI (Dense) Total # Index entries = 16384 (2^{14}) (# DB Records)

Ist Level Total # Index Block = $\frac{2^{14}}{2^6} = 2^8$ = 256 Index Block. Ans

---

IInd Level: Total # Index entries = 256 (# of Ist Level Block)

Total # Index Block = $\frac{256}{2^6} = \frac{2^8}{2^6} = 2^2$ = 4 Index Block Ans

3rd Level

Total # Index entries = 4 (# 2nd Level Block)

$$\text{Total \# Index Block} = \left\lceil \frac{4}{64} \right\rceil = 1 \text{ Index Block}$$

So Stop Here

Total 3 Level Required.

To Access a Record Using Multilevel Index = n+1 = 3+1 = (4) Ans

$\Rightarrow 1 + 1 + 1 + 1 = (4)$ Ans

- 1 → 3rd Level Index
- 1 → 2nd Level Index
- 1 → 1st Level Index
- 1 → Data Block

Any Doubt ?

THANK YOU!