Database Systems (CSF 212)

BITS Pilani
Hyderabad Campus

Prof.R.Gururaj
CS&IS Dept.

Indexing (Ch.17 of T1)

innovate achieve lead

Content

- *What is Indexing*
- *Primary and Secondary indexes*
- *Dense and Sparse Indexing*
- *Multilevel Indexing*
- *Designing Primary and Multilevel Indexes*
- *What is Tree Indexing*
- *B+ tree*
- *Inserting and deleting keys into B+ Trees*
- *Constructing a B+ tree*
- *Designing a B+ Tree node structure*

Introduction to Indexing

Indexes (Access Structures) are used to speed up the retrieval of records in response to certain search conditions.

Ex: Get all employees details where age=50;
Get employee details for eid=329; etc.

In real world databases, indexes may be too large to be handled efficiently.

Hence some sophisticated techniques are to be used.

The criteria for evaluating the hashing or indexing techniques –

- Access time
- Insertion time (new indexes or new records)
- Deletion time
- space overhead

Some times more than one indexing may be required for a file.

The attribute /field used for constructing index structure for a file is called an '*indexing field/attribute*' .

If the index field is a key, it is called as search key or indexing key.

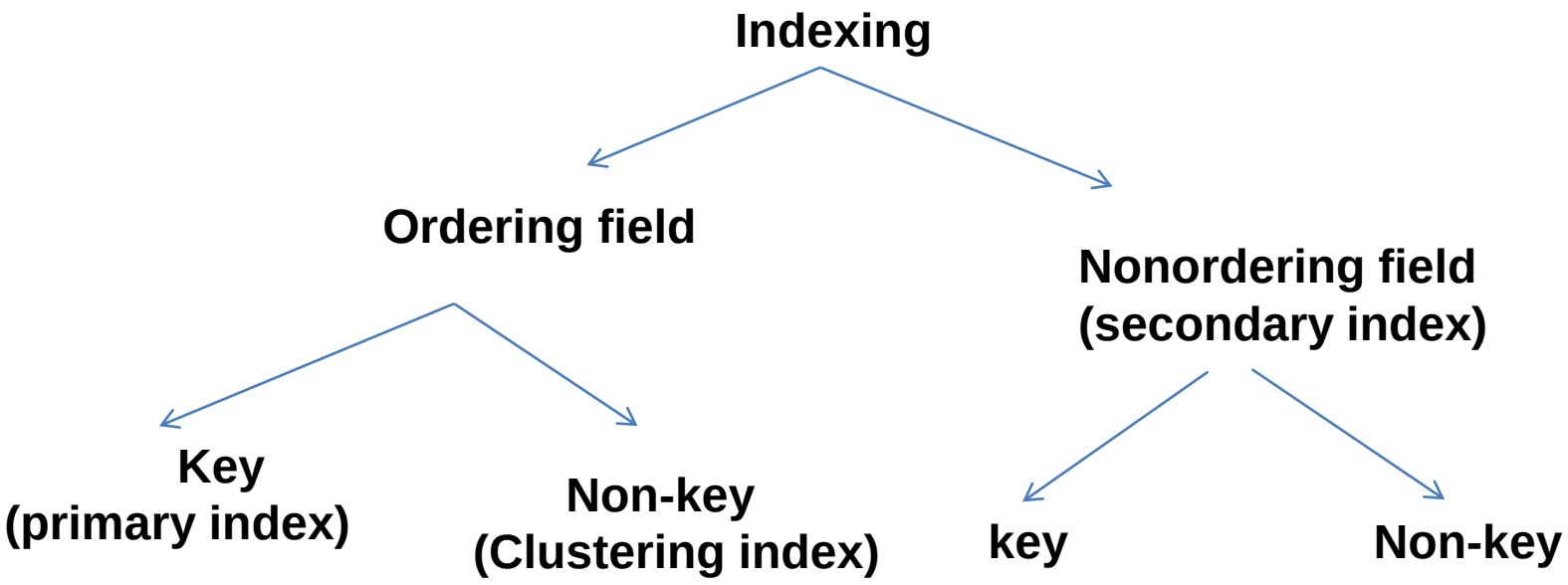
Indexes on key attributes:

1. Built on ordering key(PK) – Primary index
2. Non-ordering Key - Secondary index on key attribute

Indexes on non-key attributes:

1. Ordering non-key -- Clustering Index
2. Non-ordering non-key attribute – Secondary index on non-key

Hence, a file can have at most one primary index or one clustering index, but not both.


Indexing
Ordering field
Nonordering field
(secondary index)
Key
(primary index)
Non-key
(Clustering index)
key
Non-key

Data record: Similar kind of records(of a relation/table) are stored in a single file containing blocks. These are called data records and will have fields specified on the relation.

Index record: Like data records, index records are also stored in database. Any index record normally has two fields.

Value	Pointer
Key value	Location address of the record containing the key

Dense Index : In this, an index record appears for every data file record.

Sparse Index : Index records are created only for some data file records. This occupies less space.

A primary index and clustering index are non-dense.

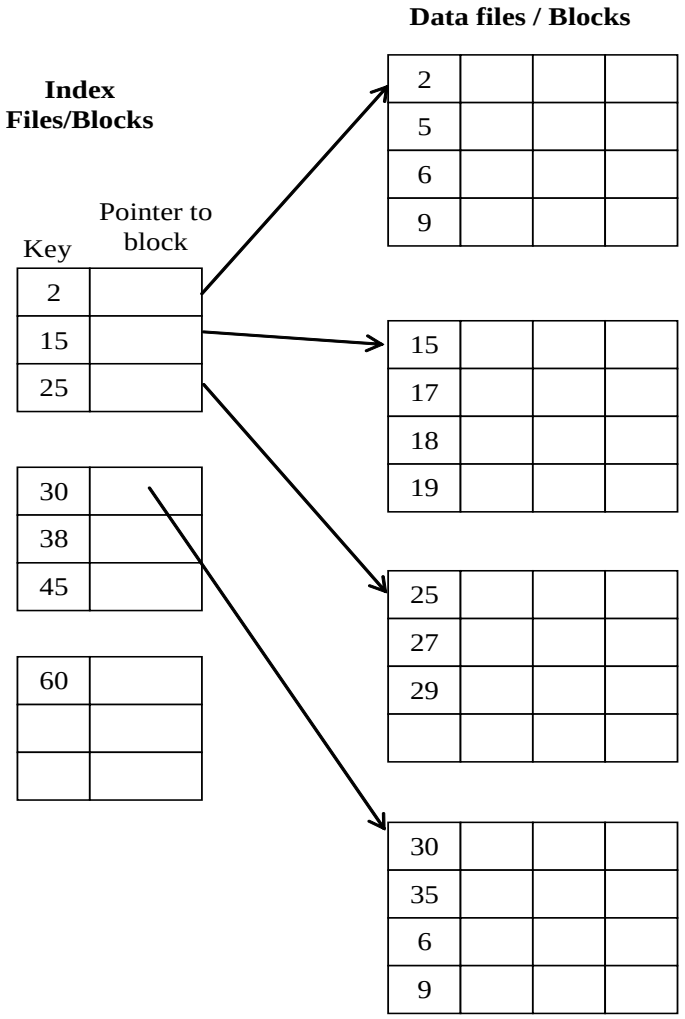
SQL Commands to create indexes:

Usually when we declare PK, an index is created automatically.

CREATE INDEX EMP_IND ON EMP(eid);

DROP INDEX EMP_IND;

Primary Indexing



Designing a Primary index

innovate achieve lead

Ex 1:

Assume that we have an ordered file with 80000 records stored on disk. Block size is 512 Bytes. Record length is fixed and it is 70 Bytes. Key field(PK) length is 6 Bytes and block pointer is 4 Bytes. Assume unspanned record organization

Design a Primary index on primary key.

Solution:

Sizeof disk block=512 Bytes; record length=70 Bytes

Block pointer=4 Bytes. Key field=6 bytes; total records=80000

No. records per block(*Bfr*)= floor (512/70)=7.31=7

No. of data blocks needed= ceil(80000/7)= 11429

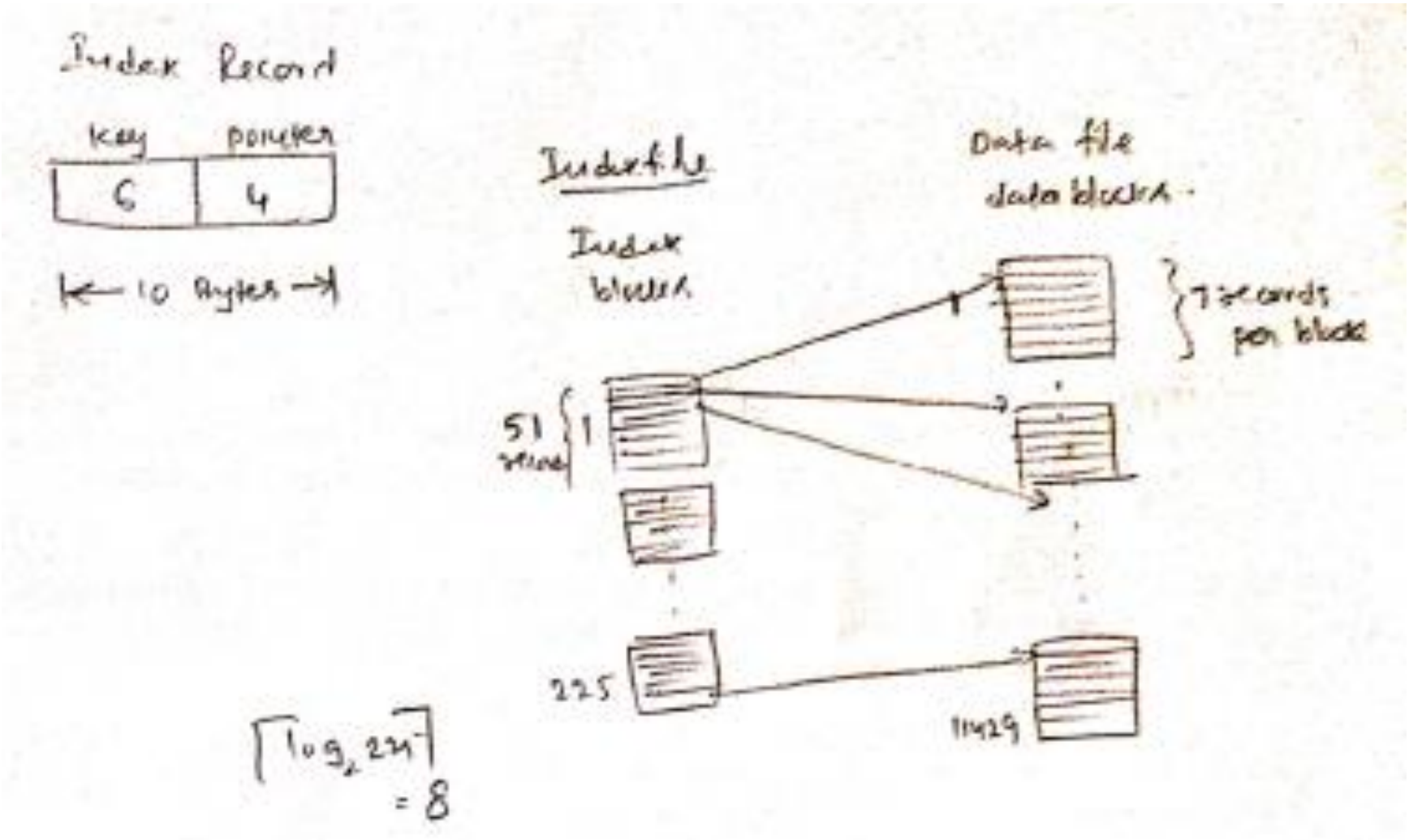
Index record length= key + pointer=6+4=10 Bytes

Blocking factor for index (Bfr_i) = floor(512/10)=51

(known as fan-out)

No. of index blocks = Ceil(11429/51)= 225

No. of block accesses= ceil of ($\log_2$ 225) + 1 = 8+1=9


Index Record
key
pointer
6
4
← 10 Bytes →
Index file
Index blocks
Data file
data blocks
51
1
225
11429
7 records per block
$\lceil \log_2 224 \rceil = 8$

Example
Designing a Primary index

innovate achieve lead

Ex 2:

Assume that we have an ordered file with 130000 records stored on disk. Block size is 1024 Bytes. Record length is fixed and it is 250 Bytes. Key field(PK) length is 10 Bytes and block pointer is 8 Bytes. Assume unspanned record organization

Design a Primary index on primary key.

Solution:

Sizeof disk block=1024 Bytes; record length=250 Bytes

Block pointer=8 Bytes. Key field=10 bytes; total records=130000

No. records per block(*Bfr*)= floor (1024/250)=4.096=4

No. of data blocks needed= ceil(130000/4)= 32500

Index record length= key + pointer=10+8=18 Bytes

Blocking factor for index (Bfr_i) = floor(1024/18)=56.88=56

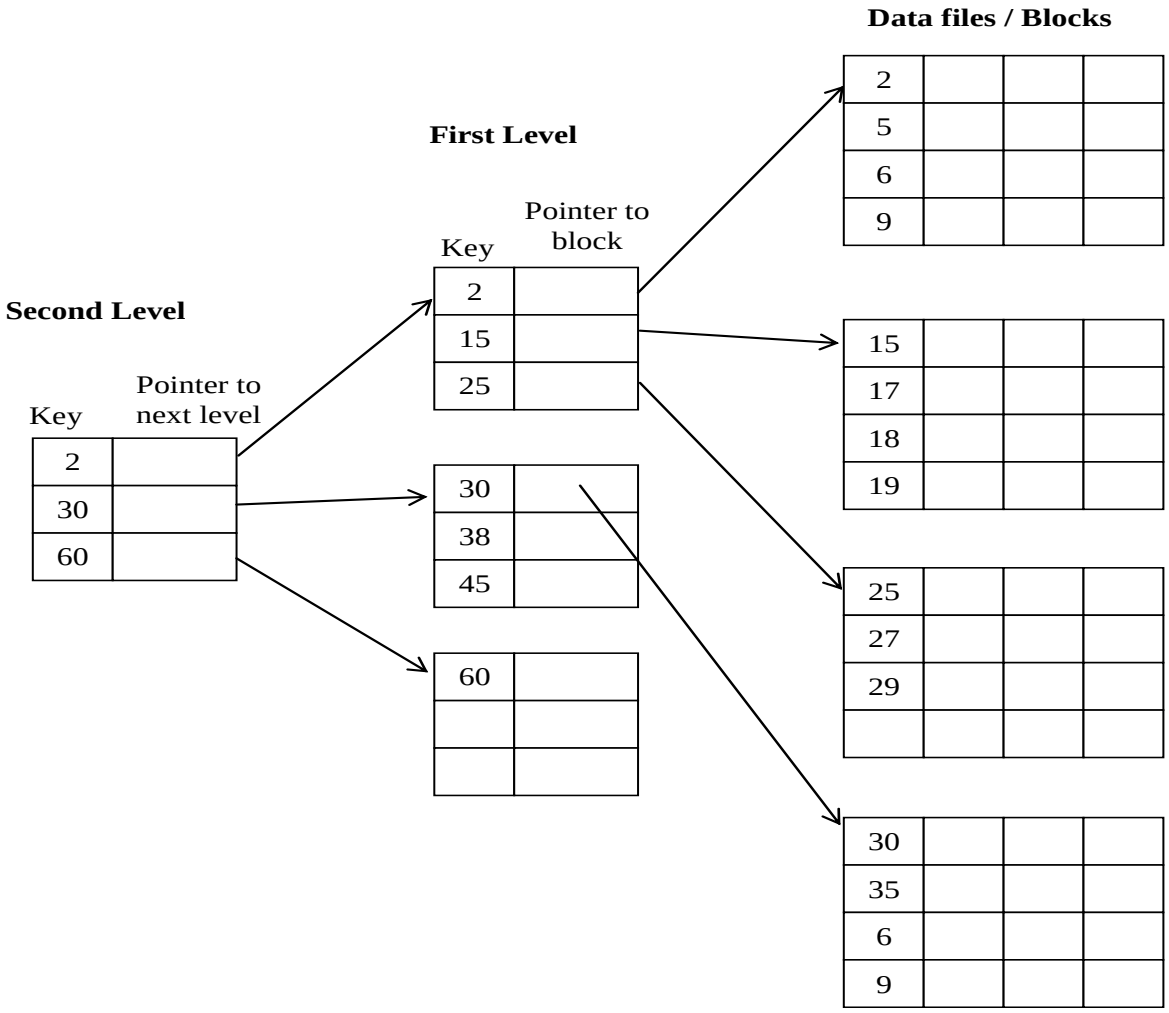
(known as fan-out)

No. of index blocks = Ceil(32500/56)= 580.35=581

No. of block accesses= ceil of ($\log_2$ 581) + 1 = 10+1=11

No. of block accesses without indexing= ceil of ($\log_2$ 32500) = 15

Multilevel Indexing (Two levels)



Designing a multilevel index

innovate achieve lead

Ex 2:

Assume that we have an ordered file with 80000 records stored on disk. Block size is 512 Bytes. Record length is fixed and it is 70 Bytes. Key field(PK) length is 6 Bytes and block pointer is 4 Bytes. Assume unspanned record organization

Design a multilevel index on primary key.

How many levels are there.

How many blocks are there in each index level.

Solution :

Size of the disk block=512 Bytes; record length=70 Bytes

Block pointer=4 Bytes. Key field=6 bytes; total records=80000

No. records per block(Bfr)= floor (512/70)=7.31=7

No. of data blocks needed= ceil(80000/7)= 11429

Index record length= key + pointer=6+4=10 Bytes

Blocking factor for index = floor(512/10)=51 - fanout

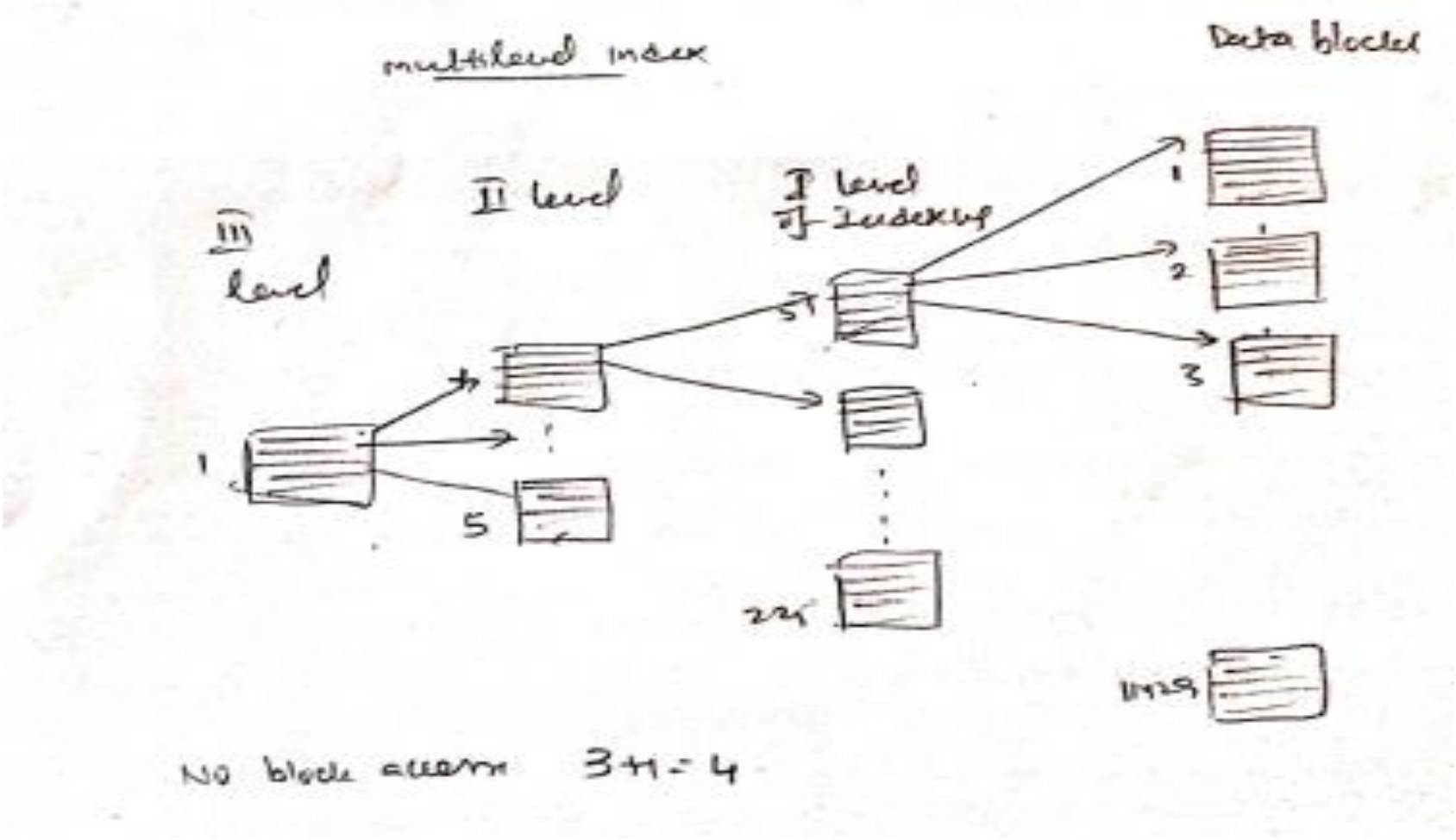
No. of index blocks in first level= Ceil(11429/51)= 225

No. of index blocks in 2nd level= Ceil(225/51)= 5

No. of index blocks in 3rd level= Ceil(5/51)= 1 top level

No. of levels in indexing structure=t=3

No. of block accesses= No. index levels + 1= t+1=4


multilevel index
Data blocks
III level
II level
I level of Indexing
1
5
5
1
2
3
224
1129
No block access 3+1=4

Example
Designing a multilevel index

innovate achieve lead

Ex 2:

Assume that we have an ordered file with 130000 records stored on disk. Block size is 1024 Bytes. Record length is fixed and it is 250 Bytes. Key field(PK) length is 10 Bytes and block pointer is 8 Bytes. Assume unspanned record organization

Design a Primary index on primary key.

Solution:

Sizeof disk block=1024 Bytes; record length=250 Bytes

Block pointer=8 Bytes. Key field=10 bytes; total records=130000

No. records per block(*Bfr*)= floor (1024/250)=4.096=4

No. of data blocks needed= ceil(130000/4)= 32500

Index record length= key + pointer=10+8=18 Bytes

Blocking factor for index (Bfr_i) = floor(1024/18)=56.88=56

(known as fan-out)

No. of index blocks (1st level)= Ceil(32500/56)= 580.35=581

No. of index blocks (2nd level)= Ceil(581/56)= 10.375=11

No. of index blocks (3rd level)= Ceil(11/56)= 1

No. of block accesses= No of levels+ 1 = 3+1=4

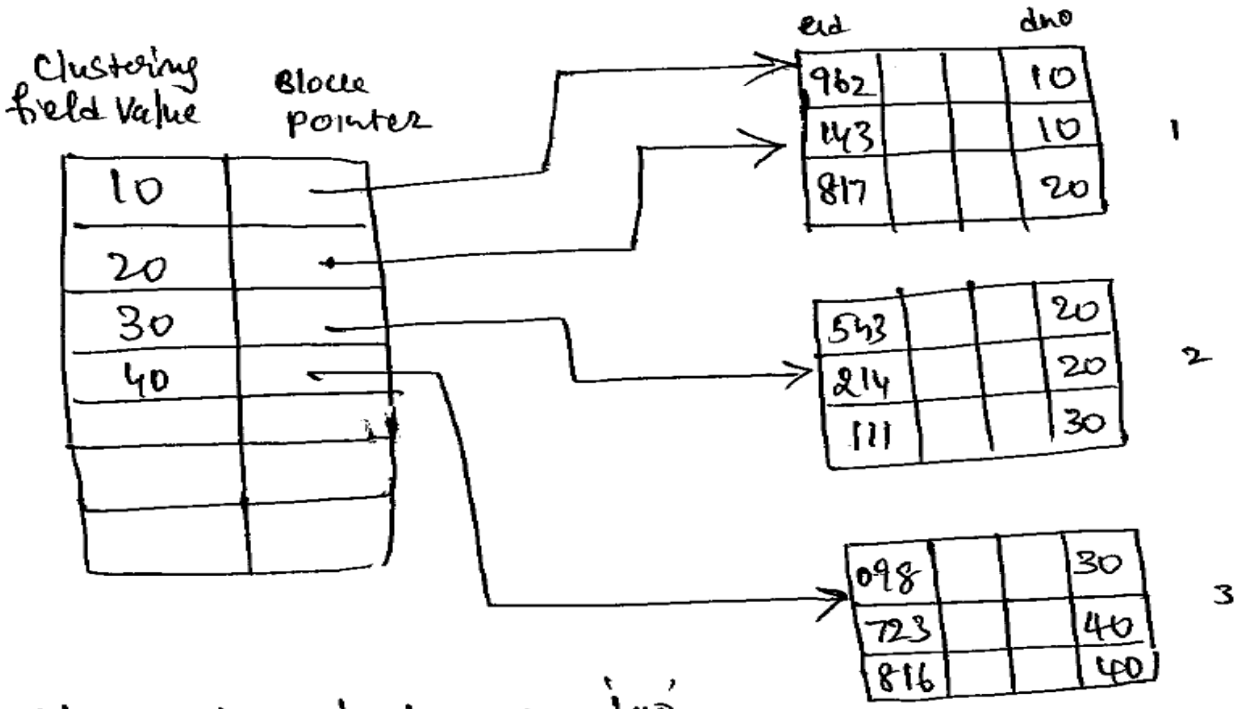
Note:

No. of block accesses without indexing= ceil of ($\log_2$ 32500) = 15

No. of block accesses with only one level= ceil of ($\log_2$ 581) +1 = 10+1=11

No. of block accesses with only two levels= ceil of ($\log_2$ 11) +1 = 4+1+1=6

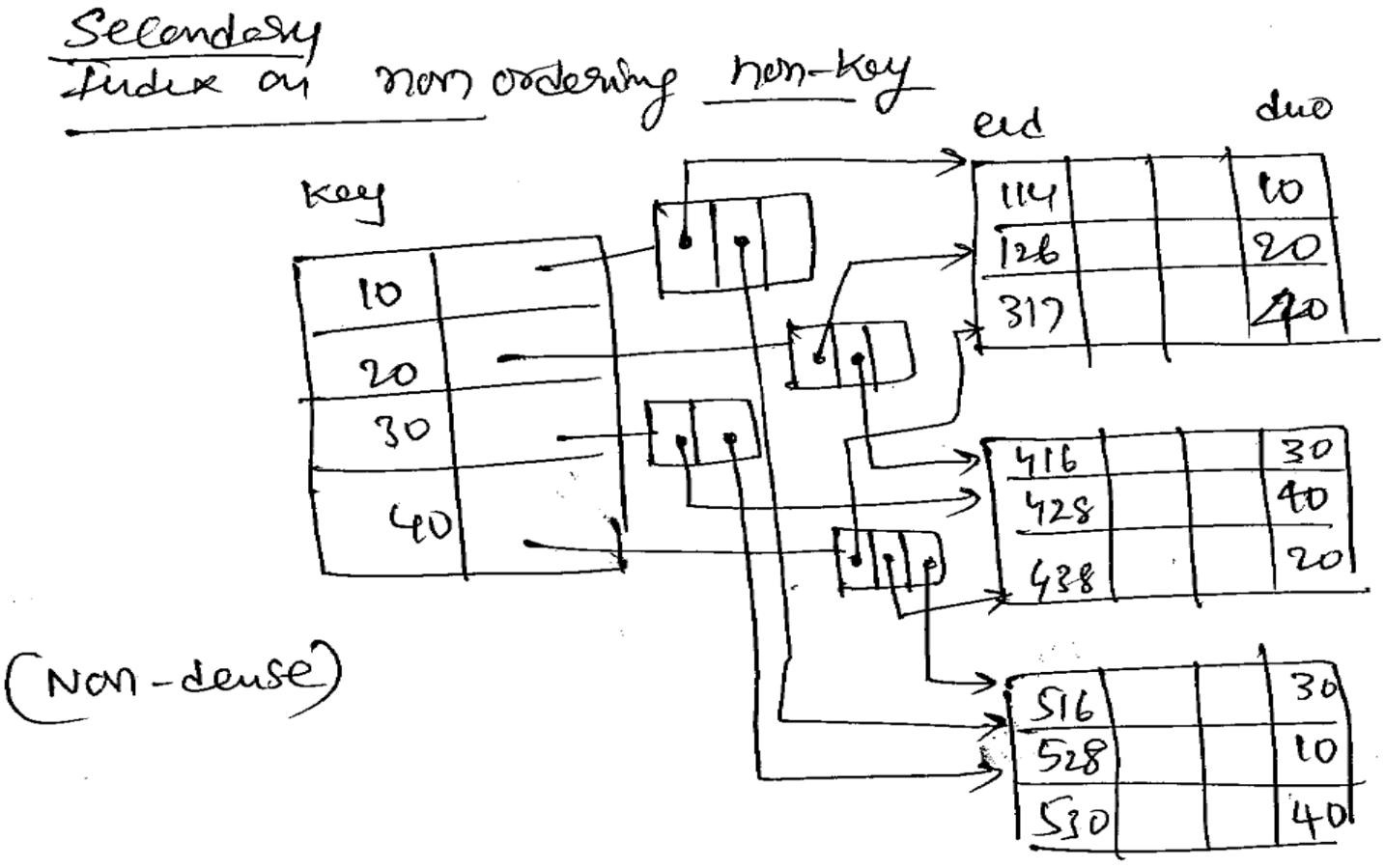
Clustering Index on Ordering Non-key



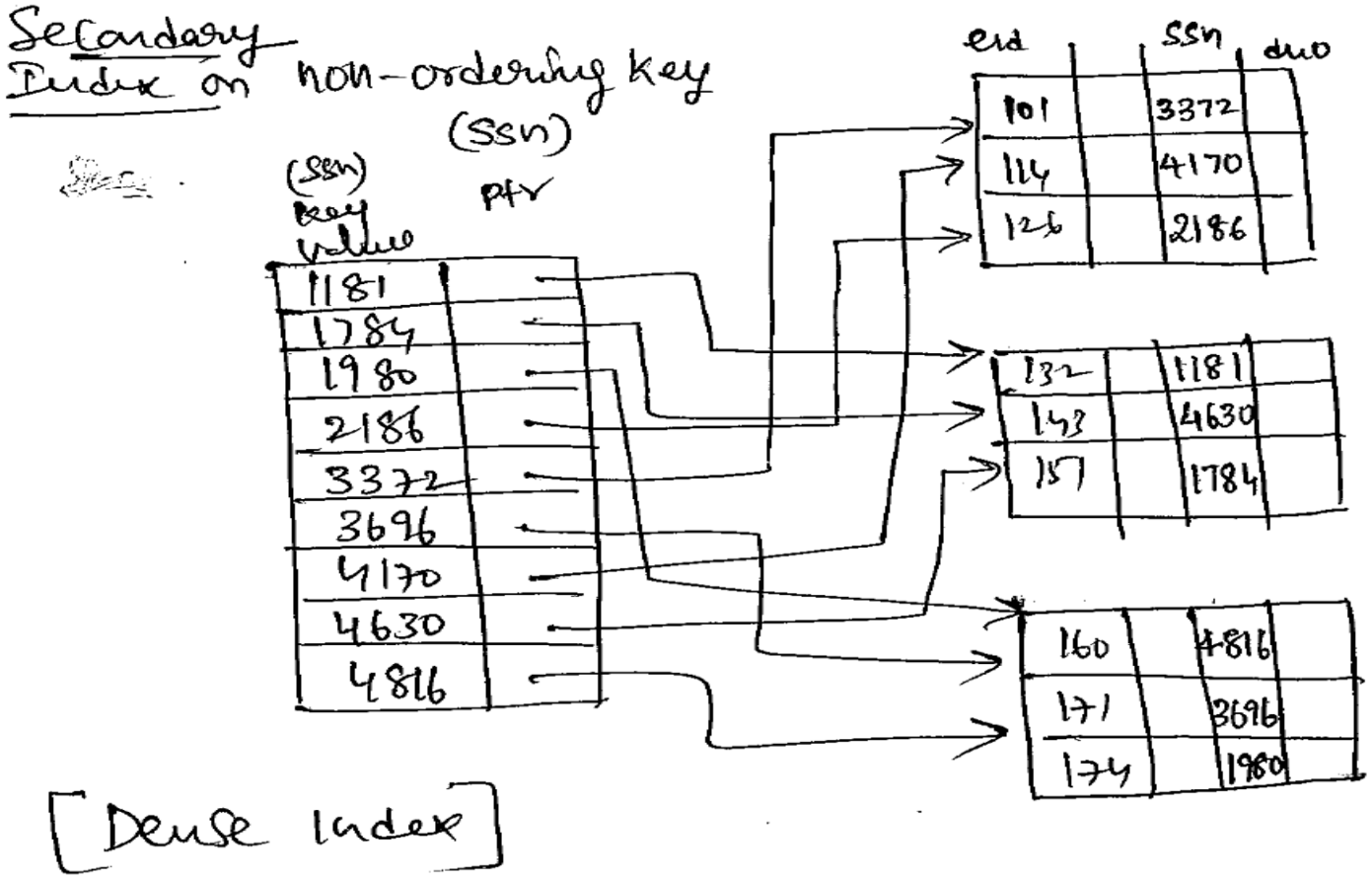
Clustering Index on dno of Employee records.

Non Dense

Secondary Index on Non-ordering, Non-key



Secondary Index on Non-ordering key



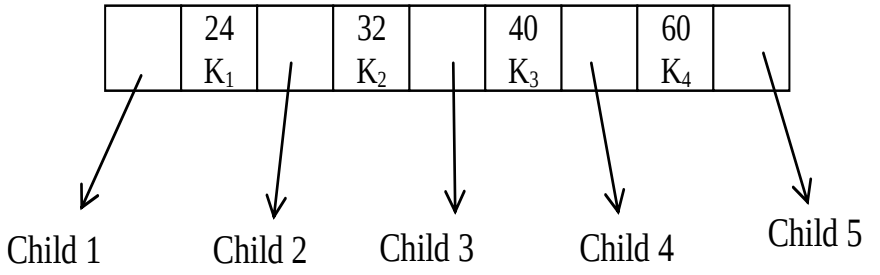
B+ Tree Indexing

B+ Tree is a multilevel search tree used to implement dynamic multilevel indexing. The primary disadvantage of implementing multilevel indexes is that the performance degrades as the file grows. It can be remedied by reorganization, but frequent reorganization is not advisable. B+ tree is best suited for multilevel indexing of files, because it is dynamic.

B+ Tree of Order *p*

It is a balanced tree, (all leaves are at same level).

Each internal node is of the form-



Constructing a B+ Tree

Ex 3:

Construct a B+ tree with given specifications. The order of the tree, p=3 and p_{leaf} =2. The tree should be such that all the keys in the subtree pointed by a pointer which is preceding the key must be equal to or less than the key value , and all the keys in the subtree pointed by a pointer which is succeeding the key must be greater than the key.

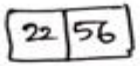
Insert the following keys in same order- 56, 22, 78, 42, 102, 90, 35. Show how the tree will expand after each insertion, and the final tree.

Next, delete 56, 46, 22 in the same order and show the status of the tree after each deletion.

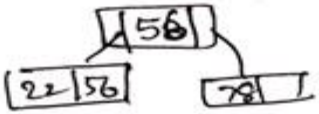
B+ Tree P = 3 Pleaf = 2

Keys - 56, 22, 78, 42, 102, 90, 35

Insert 56, 22

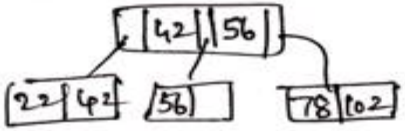


Insert 78

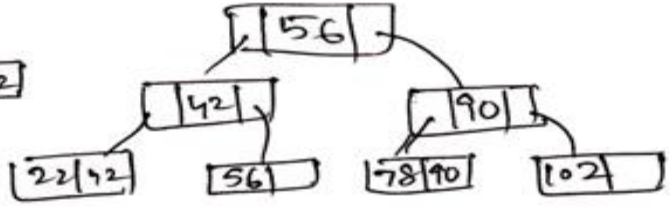


Insert 42

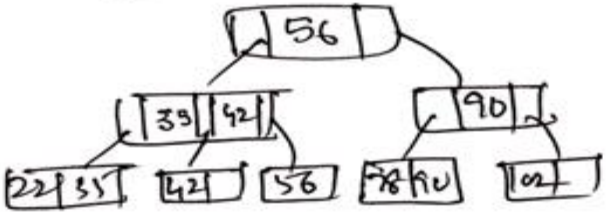
Insert 102



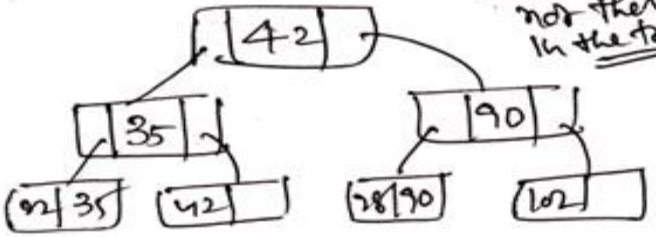
Insert 90



Insert 35



Delete 56

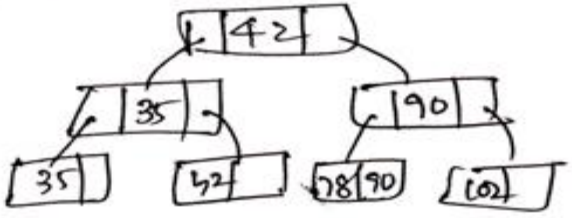


Delete 46

not there in the tree

Final tree

Delete 90



final tree

Node design for B+ tree

Example:

We need to design a B+ tree indexing for Student relation, on student_id attribute; the key of the relation. The attribute student_id is of 4 bytes length. Other attributes are- student_age(4 bytes), student_name(20 bytes), student_address(40 bytes), student_branch(3 bytes). The Disk block size is 1024 Bytes. If the tree-pointer takes 4 bytes, for the above situation, design the best possible number of pointers per node(internal) of the above B+ tree. Each internal node is a disk block which contains search key values and pointers to subtrees.

Solution:

Disk block size=1024 Bytes

Size of B+ tree node= size of disk block

Each tree pointer points to disk block and takes 4 Bytes.

Each key (student_id) takes 4 Bytes

In a B+ tree node, No. of pointers = no. keys +1

Assume that no. keys = n

Then no. pointers= n+1

Then min. size for a node= {(no.Keys* size of each key)+
(no.pointers * size of each pointer)} <= 1024

$(n*4)+(n+1)*4 <= 1024$

$4n+4n+4 <= 1024$

$8n+4 <= 1024$

$8n <= 1024-4= 1020$

$n <= 127.5$ or 127

hence In each internal node, no. keys=127; and no. pointers=128

Note

In a B+ tree record pointer for a record with given key can be found only at leaf node.

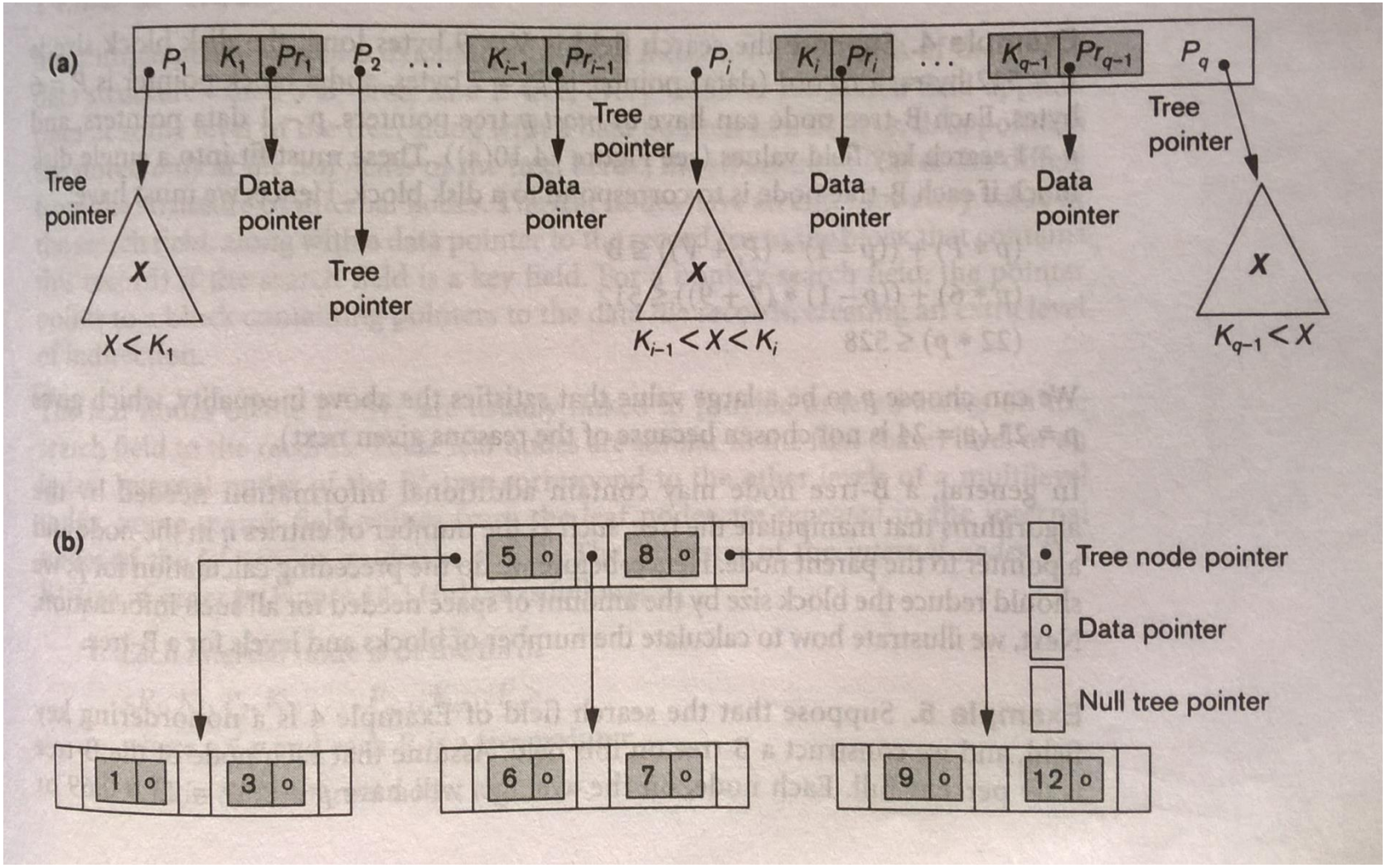
But if it is in case of B-tree it can happen at intermediate node also.

Hence in B+ tree search, success or failure can be declared only after reaching leaf_level.

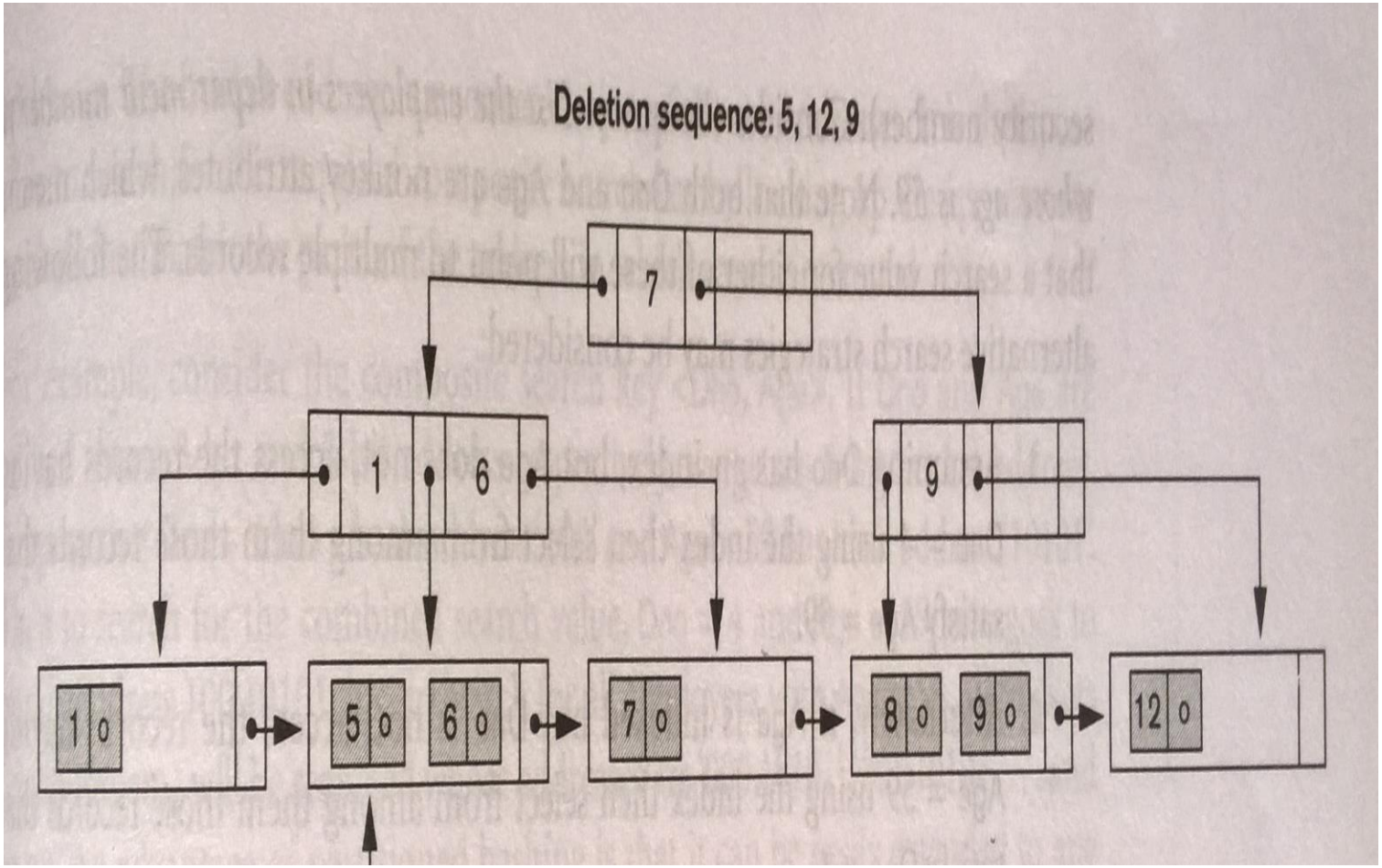
Where as in B-tree search can be successful at intermediate level as well.

On failure we reach the leaf level.

B Tree


(a)
P_1 K_1 Pr_1 P_2 ... K_{i-1} Pr_{i-1} P_i K_i Pr_i ... K_{q-1} Pr_{q-1} P_q
Tree pointer
Data pointer
Tree pointer
Data pointer
Tree pointer
Data pointer
Data pointer
Tree pointer
X
$X < K_1$
$K_{i-1} < X < K_i$
$K_{q-1} < X$
(b)
5 8
1 3
6 7
9 12
Tree node pointer
Data pointer
Null tree pointer

B+ Tree



Example for B+ tree : Page 644 of T1

Reading assignment:

Bitmap Index: Ch.17.5.2 of T1 (page-649)

Summary

- *What is Indexing and its importance*
- *How Primary and Secondary indexes work*
- *Examples of Dense and Sparse Indexes*
- *What is Multilevel Indexing*
- *Some example problems on designing Primary and Multilevel Indexes*
- *What is Tree Indexing*
- *B tree and B+ tree concepts*
- *Constructing a B+ tree (Insert/Delete operations)*
- *Designing a B+ Tree node structure*
- *Bitmap index(reading assignment)*