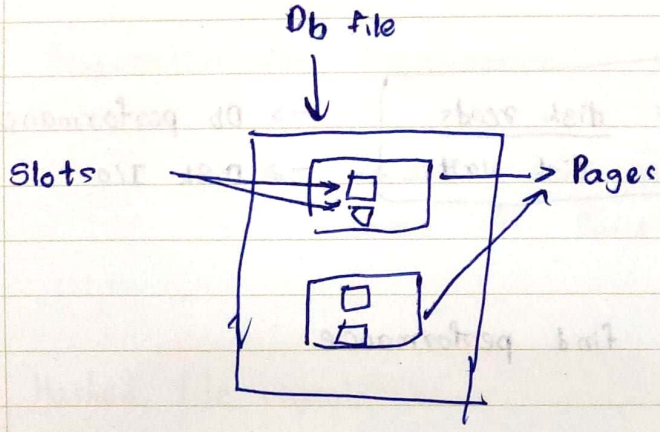
to optimize db we need file organization.

File organization & indexes

Date No

lec 7



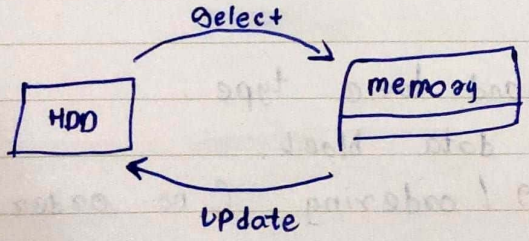
- Db file is a collection of pages. page is a collection of slots.

- in Slots store data, each slot have a single data.
- Page is a collection of slots.

if there are 100 records there will be 100 slots.

- actually data stored in HDD. if we run a query to Select data it will read data from HDD and send to memory.

- in update store to memory as Select and update is the memory. and write back in HDD.



1 disk block = 1 db file page.

- at a time bring one page at a time to memory

(HDD $\xrightarrow{\text{read}}$ memory)

Date No

db permormance Calculated based on disk read and disk write. €

Performance = disk reads / disk writes

-> Db performance
-> Disk I/O

1000 -> records
10 records in 1 page find performance

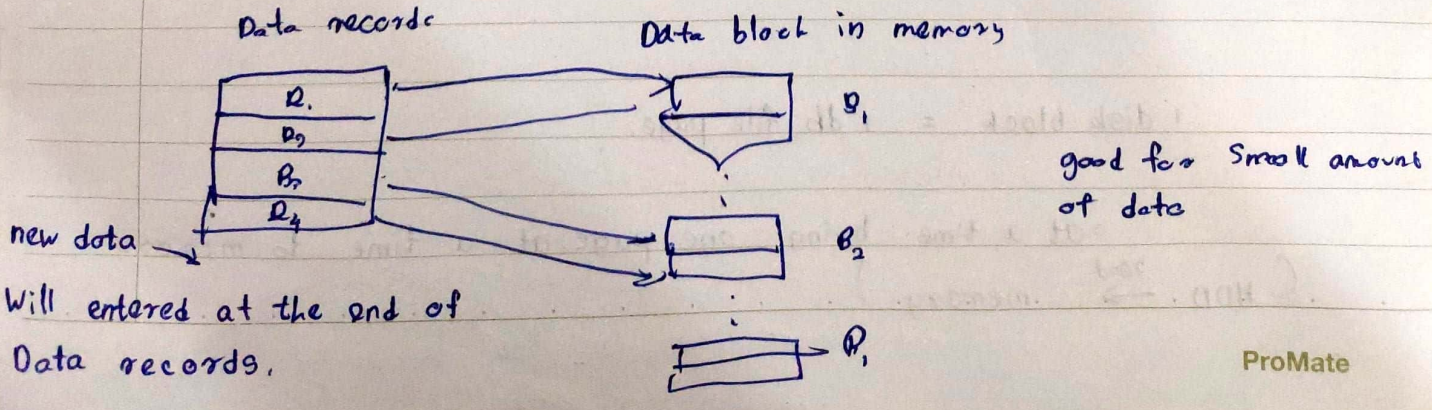
$= \frac{1000}{10}$

$= 100$ (read : write)

3 type of file organization
1. Heap file
2. Sequential file } to optimize the db.
3. Hashing file

① Heap file organization

- Simplest and basic type.
- Work with data block.
- No sorting / ordering. (no order to store data)



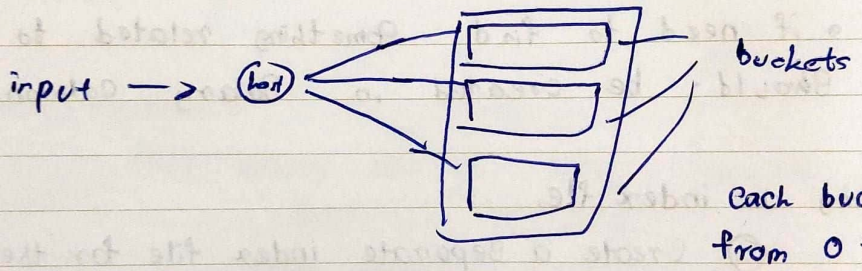
- Update / delete time concuming (need to search)
- not good for large files.

02) Sequential file organization.

- records are stored in a order. (Asc / Desc)
- based on some column - Can store data.

03) Hashed file organization.

- insert our input through a hash function. and taking a hash output, and assigned to a buckets.



Each bucket has lable from 0 to 9 one. according to the remainder it will allocate for bucket

$$\text{bucket} = \frac{\text{Value}}{\text{bucket count}}$$

$$= \frac{70\,000}{10}$$

= take the remainder only '0'

= insert to '0' bucket.

- Hashed file organization not sorted well.
- Store in different buckets acording to remainders.

INDEX.

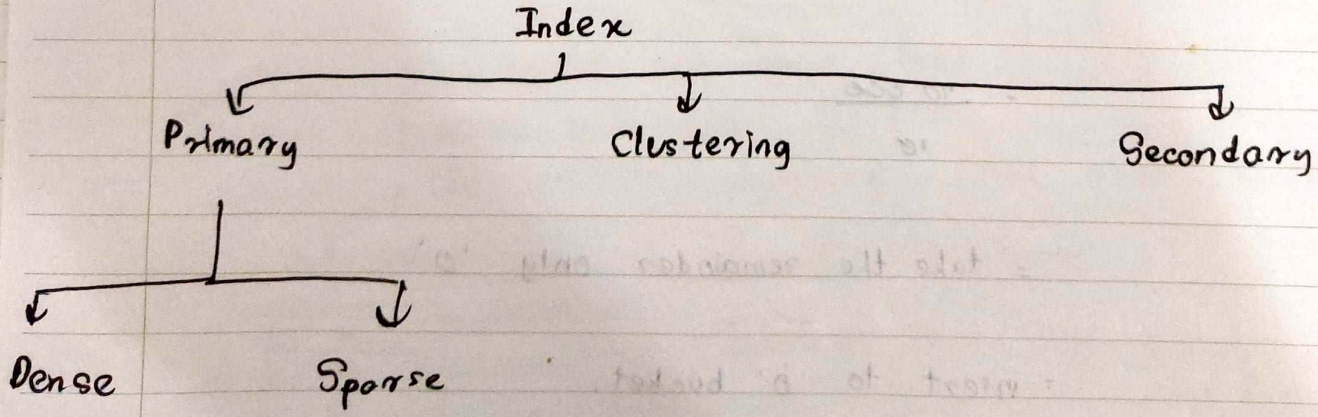
index mainly 2 part
- Search key (key)
- data reference (value)

- together called index. Used to optimize the Speed of db.

- it reduce the read and write to the HDD. to create a index we can ~~your~~ use more than 1 column.

- if need to find something related to Salary index should be created in Salary column.

Creating an index file.
1. Create a separate index file for the column (s).
2. Enter all values to index file.
3. Sort the values (ASC / DESC).



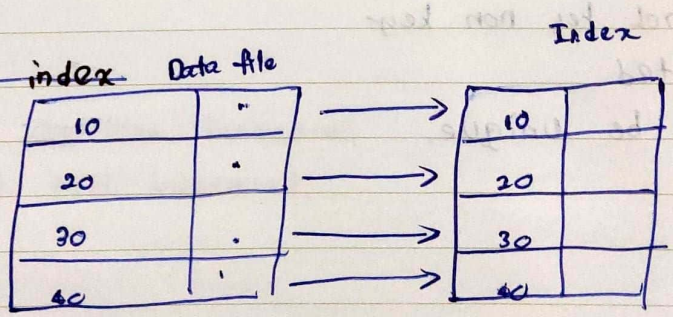
Primary index.

- is a ordered file which is fixed length size with tow fields. (ordered file)
- Created using primary key.
- 1 to 1 relationship between the entries in index table.
- Search key is
  - Sorted
  - not null
  - fast

Dense index

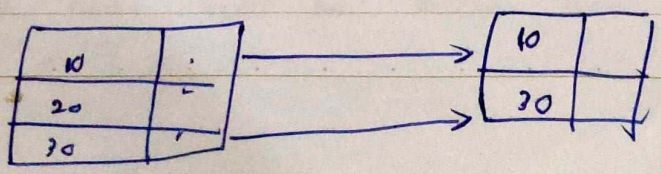
- for every search key record is created
  - fast
  - accurate
  - need more storage.

Sparse index



Sparse index.

- appeared only for some B.Search key.



Secondary indexing

- Can be generat by a Candidate key, with a level indexing.

- data is not Sorted. non clustured index.

- more time required
- not Sorted
- Search key cant be null
- Slower than primary and faster than clustured.

Clustering index.

- used multiple related records found at one place.

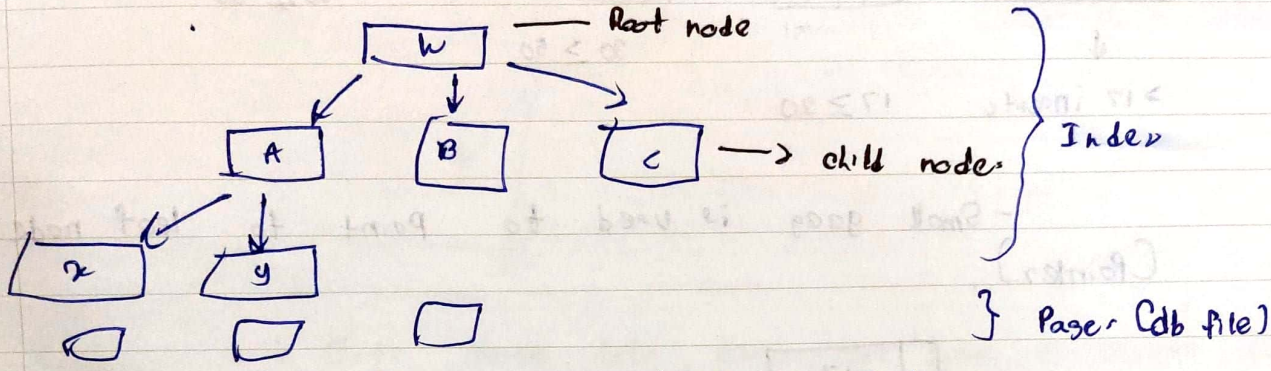
- Sorted data.
- records stored in indexes.
- Created in non-prim columns.
- Can group columns.

- Search key non key
- Sorted
- not be unique.

Using B+ tree.

for range searches.

- Use to impliment the db indexes (B+ trees)



- x Parent A, A Parent Root

- upper byer Parent, lower layer child.

- last node are leaf nodes. We actually stor db file records in leaf node.

- Can use for insert, update, delet.

Insert.

- hight of B+ tree increased.
- Width of B+ tree increased.

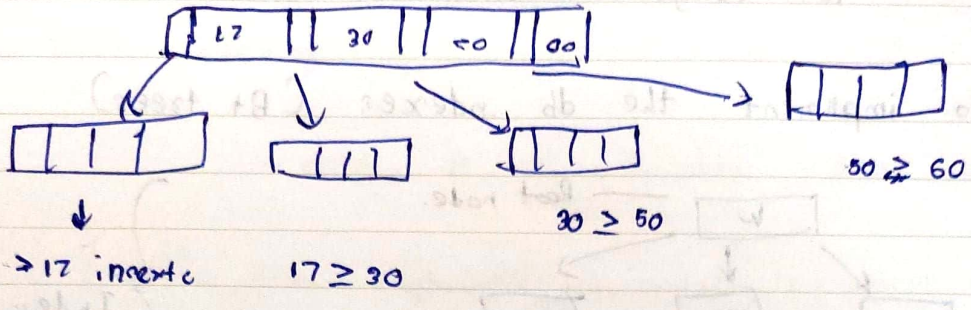
Delete.

- hight ↓
- Width ↓

- each node must be filled 50%. without root.
- Cant avoid it.

Date No

B+ tree



- Small gaeg is used to Point to leaf node. (Pointer).