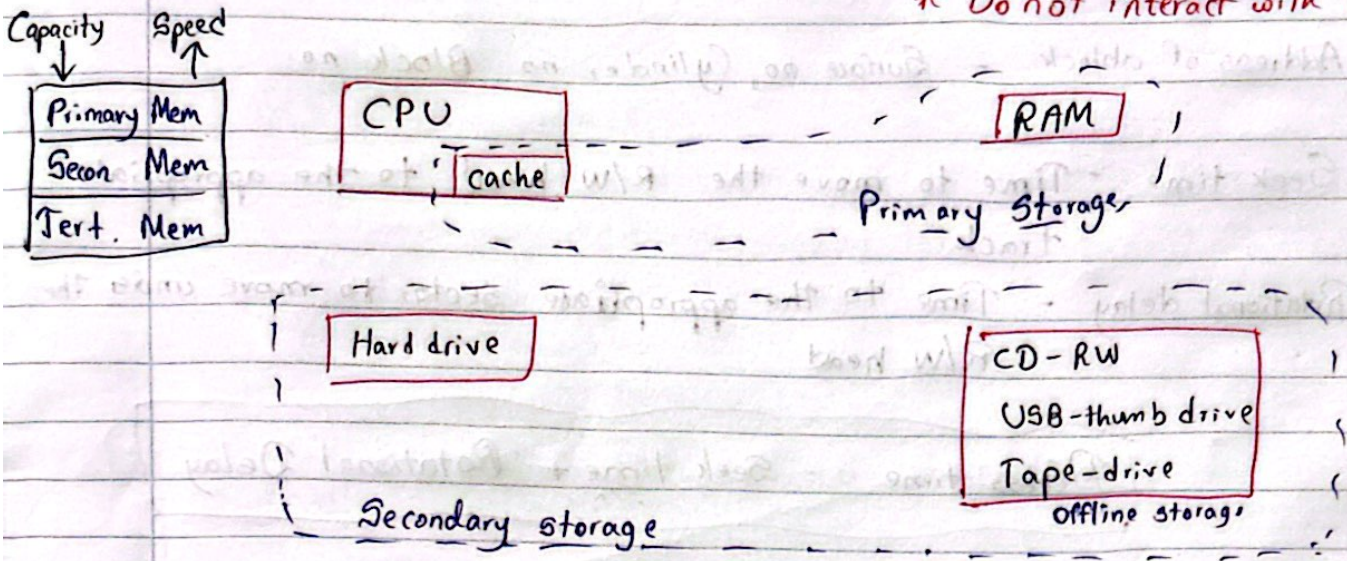
No: _____ Date: __/__/____

Database Notes

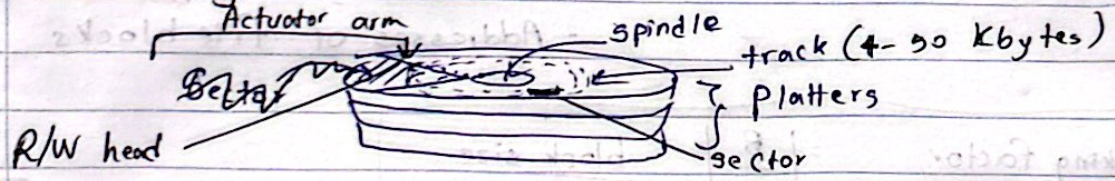
File organization and storage structures

- **Primary Storage** (Main memory)
  - * Fast, Volatile, Expensive
- **Secondary storage** (Files in disks or tapes)
  - * Non-volatile, Less-expensive
  - * Do not interact with CPU



* Databases are usually stored on magnetic disk secondary storage
* Data stored in disks as files of records.
* Record is a collection of data values that can be interpreted as facts about entities, their attributes and their relationships

Disk storage devices



* Block size is fixed for each system
* It is the smallest trasferable unit.
  Vary from 512 bytes to 4096 bytes.

No: Date: __/__/____

When trasfering data,

1) R/W head moves to the respective track
2) The platter rotates to the respective sector / block.

Is time consuming because of seek time s and rotational delay (rd) / latency

Blocking - Storing a number of records in one block
Blocking factor (bfr) - No. of records per block.

Address of ablock - Surface no, Cylinder no, Block no.

Seek time - Time to move the R/W head to the appropriate track

Rotational delay - Time to the appropriate sector to move under the R/W head.

Access time = Seek time + Rotational Delay

Seek & Rotational time >>> Transfer time ← Bottleneck in database

File - Sequence of records
File descriptor/File header - Contains details about the file
- Field names, Data types
- Addresses of file blocks

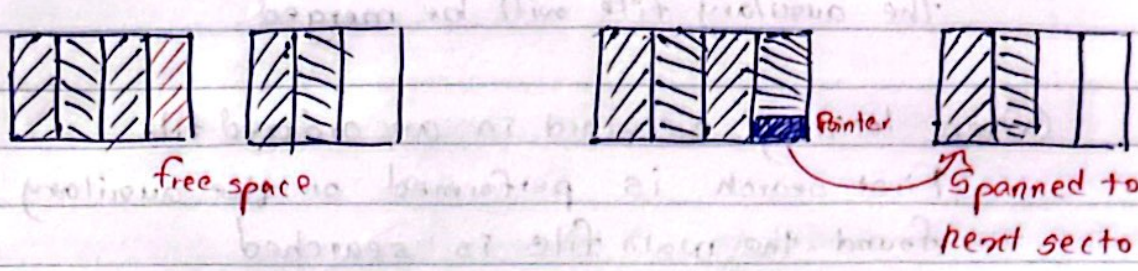
Blocking factor (bfr) $= \left\lfloor \frac{B}{r} \right\rfloor \cdot \frac{\text{block size}}{\text{record size}}$

Atlas

Date: __/__/____

Record spanning

Unspanned records: Spanned records



Types of block allocation

i, Contiguous allocation
- \+ Easy file scanning
- − Difficult to expand the file by including blocks in middle

ii, Linked allocation
- \+ Easy to expand
- − Difficult to scan

iii, Mixed allocation

File operations - Read from slides

Primary file organization methods

- Heap / Unordered file (Look at slides) Complexity (b/2
  - * No particular order
  - * Appends new records at the end of the file
- Ordered file (Sorted / Sequencial) Complexity ($\log_2 b$)
  - * Records are ordered by a field value (sort key)
  - Normal - b/2, Binary search - $\log_2 b$
  - Binary sear
- Hashed file
  - * Applies a hash function to a particular field to determine the placement
  - Chashke
- B-trees
  - * Use tree structures

When inserting a record in an ordered file

Insert the new record in an auxilary file and % when periodic file reorganization is happening, the main file and the auxilary file will be merged.

When locating a record in an ordered file

First search is performed on the auxilary file, if not found the main file is searched

Linear ordering queries

```
* Create table sales (
        prod_id   number(6) not null,
        cust_id   number not null)
    )
    CLUSTERING BY LINEAR ORDER (cust_id, prod_id);
```

* Alter tabe sales ADD CLUSTERING BY LINEAR ORDER (cust_id, prod
* Alter table sales order by prod_id ASC;

	Search Method	Average time to access a record
Heap / Unordered	Sequential	b/2
Ordered	Sequencial	b/2
Ordered	Binary	$\log_2 b$

Problem of a hash file

* Collisions occur when a new record hashes to a bucket that is already full. An overflow file is kept for storing such records.

Indexing

To do these calculations, the records should be fixed size and unspanned

2) Slide 02 - Page 05

$$bfr = \left\lfloor \frac{1024}{100} \right\rfloor = \lfloor 10.24 \rfloor = 10$$

$$\therefore \text{No. of blocks} = \frac{30000}{10} = 3000$$

* When optimizing query performance, the objective is to minimiz the number of disk / block accesses. Because the performance depends on how many block accesses it takes to perform the query

What is indexing ?

Defining an additional data structure which provide an alternative path to access data without affecting its original representation to improve database operation's performance.

Indexing can be done as

- Creating an index using a single field.
- Creating multiple indexes on multiple fields.
- Indexes on multiple fields.

An index entry is of format

$$< K(i), P(i) >$$

K(i) ↑ Search key

P(i) ↑ Block pointer / Row pointer

Indexing field - The field used to order the index records in the index file.

What could be the reason for the number of block accesses being less compared to data file

i) An index ~~entry~~ record length is comparably smaller than a data record.

ii) We can perform binary search on the index file always

Indexes

Sparse

* Contains records for only some of the search key values
Applicable to ordered data files only

Dense

* Contains a record for every search key value.

Single level indexes

Primary Index
Ordering key field of an ordered file

Clustering Index
* Ordering non key field of an ordered file

Secondary Index
* Any non ordering field.
May be key or non-key field
* Ordered / Non ordered file.

Primary Index (Ordered data file, Ordered on a key field)

* Includes one index entry for each block in the data file
* Index entry has the key field value for the first record in the block (block anchor)
* Sparse index.

Accessing a record = $\log_2 b$ + 1

$\log_2 b$ ↑ Binary search

1 ↑ Accessing the block

Q) Slide 2, Page 21

$$bfr = \left\lfloor \frac{1024}{100} \right\rfloor = \lfloor 10.24 \rfloor = 10 \text{ records per block}$$

~~No. of block accesses to search~~

No. of blocks = $\frac{30000}{10} = 3000$ blocks.

∴ No. of block accesses to search = $\lceil \log_2 b \rceil$

$= \lceil \log_2 3000 \rceil$

$= \lceil 11.55 \rceil$

$= 12$

*)

Index record size = 9 + 6 = 15 bytes

No. of records = No. of blocks = 3000

$$bfr = \left\lfloor \frac{1024}{15} \right\rfloor = 68$$

No. of blocks = $\lceil \frac{3000}{68} \rceil = 45$

∴ No. of block access = $\lceil \log_2 45 \rceil$

$= 5.4$

$= 6$

No: Date: __/__/__

Indexing Part II

2) Slide 03, Page 04

$$bfr = \left\lfloor \frac{2048}{80} \right\rfloor = \lfloor 25.6 \rfloor = 25$$

No of blocks : $\frac{40000}{25} = 1600$ //

3) Page 06

*) $bfr = \left\lfloor \frac{4096}{100} \right\rfloor = 40$

No. blocks = $\frac{300\,000}{40} = 7500$

Block accesses = $\log_2 7500 = 13$

*) Index record size = 9 + 6 = 15 bytes

$$bfr = \left\lfloor \frac{4096}{15} \right\rfloor = 273$$

No. of blocks : $\left\lceil \frac{7500}{273} \right\rceil = 28$

∴ No. of block accesses for index = $\lceil \log_2 28 \rceil = 5$

No. of total block access = 5 + 1 = 6 //

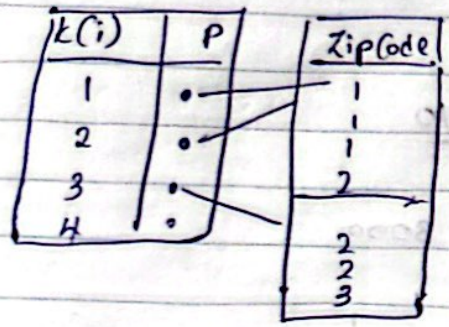
Problems of primary indexing

* Inserting and deleting records in the main file must move other records, since it's ordered
* Some insertions/deletions must also change index entries, the anchor records change.
* There can be only one primary index on a file.

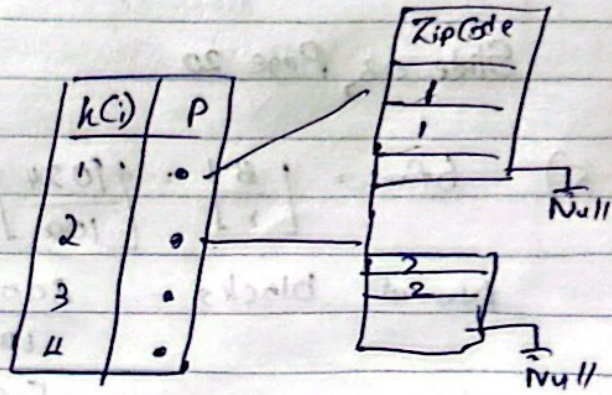
Clustering index (Ordered data file, Ordered non key field)

* Includes one index entry for each distinct value of the field
* Sparse index
* Index entry points to the first data block that contains record with that field value.

Two types of clustering index



* New values start with the same block



* New values start with a new block

Page 15

Q) Index entry length = 5 + 6 = 11 bytes

$$bfr = \left\lfloor \frac{4096}{11} \right\rfloor = 372$$

$$\text{No. of blocks in index file} = \left\lceil \frac{1000}{372} \right\rceil = 3$$

$$\text{No. of block accesses} = \lceil \log_2 3 \rceil = 2$$

No: _____ Date: __/__/____

Secondary index (Non ordering key/non key field)

* There can be many secondary indexes defined for a single file.

If the index field is a key field (Secondary key)
↳ Then index is dense

If the index field is a non key field
↳ Then index is sparse

Slide 03, Page 20

a) $bfr = \lfloor \frac{B}{r} \rfloor = \lfloor \frac{1024}{100} \rfloor = \underline{10}$

No. of blocks = $\frac{30000}{10} = 3000$

No. of block access = $\lceil \log_2 3000 \rceil = \underline{12}$ ✗ Cannot do binary as it is non ordered

$\therefore \frac{3000}{2} = 1500$

#) Index entry length = 9+6 = 15 bytes.

~~No of blocks = 30000~~

$bfr = \lfloor \frac{1024}{15} \rfloor = 68.$

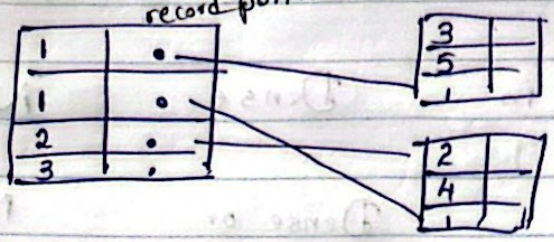
No. of blocks = $\lceil \frac{30000}{68} \rceil = 442$

No. of block accesses = $\lceil \log_2 442 + 1 \rceil$

$= \underline{10}$

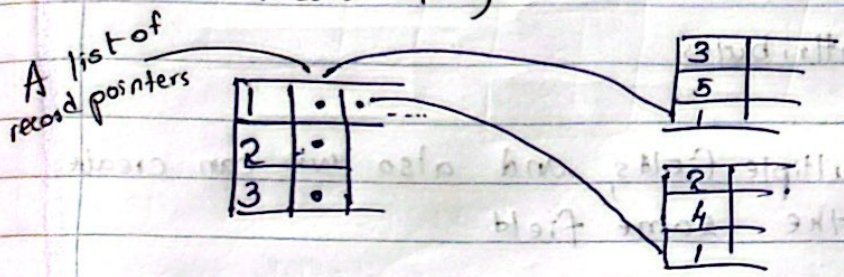
If the field is a non-key field, which means there can be duplicate values, there are 3 ways that we can implement secondary index.

01/ Several indexes fo same K(i) value, hence a dense index



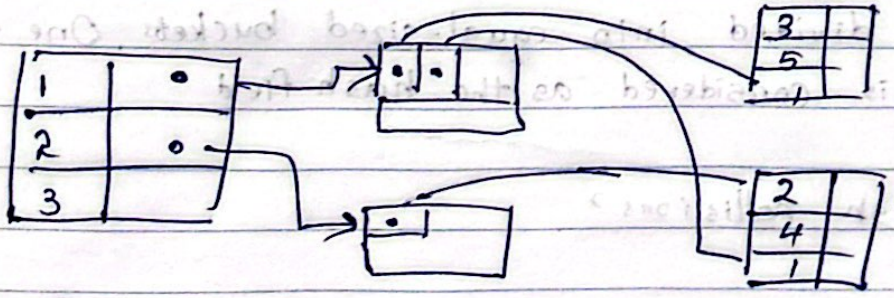
* In this case we have record pointers instead of block pointers.
* Dense index

02/ One index for one K(i) value, but variable index record length



* Record pointers, and index records are variable length
* Sparse index

03/ Have two indexes, the first index have fixed size length and a block pointer. The second index have a list of record pointers



* Here the first index is of fixed length and it has block poin
* Non dense index

first index second index

Type of Index	No. of first level index entries	Dense or Nondense	Block anchoring on the data file
Primary	No. of blocks in data file	Nondense	Yes
Clustering	No. of distinct index field values	Nondense	Yes / No.
Secondary (key)	Numbe of records in data file	Dense	No.
Secondary (nonkey)	No. of records or No. of distinct index values	Dense or Nondense	No

Indexing on multiple attributes

An index can have multiple fields, and also we can create multiple indexes for the same field.

Create index person_fn_idx on Person (first_name, last_name)

Hash index

Blocks are divided into equal-sized buckets. One of the record field is considered as the hash field.

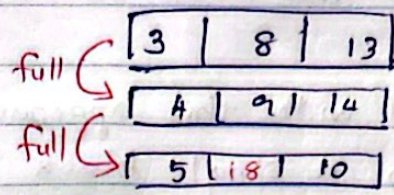
What is hash collisions?

When a new record is hashed to a bucket which is already full, it is known as a hash collision.

What are the hash collision resolution mechanisms?

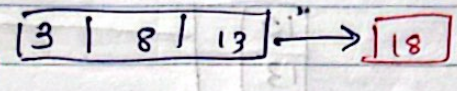
i) Open addressing - If the bucket given by hashing function is already full, move to the adjacent buckets until we find an empty slot

h(18)
= 18 mod 5
= 3rd bucket



ii) Chaining - Put the value in an overflow location and set a pointer to the intended location

h(18)
= 18 mod 5
= 3rd bucket



iii) Multiple hashing - Apply a second hashing function if the first results in a collision.

* Hash index is also a type of secondary index.

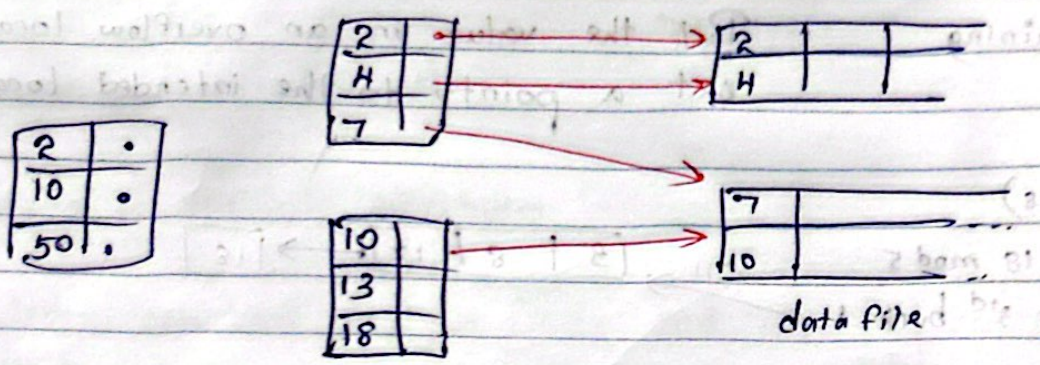
Multi-level indexing

Since an index is ordered, we can create a primary index to the index file.

The initial index file is called the first-level index

Index for the index is called the second-level index

* We can proceed with this approach until the ~~first~~ top level index fit into a single block



Second index First index

$$\text{Number of levels needed} = \lceil \log_{fo}(r_1) \rceil$$

fo = fan out (blocking factor for index)

r1 = Number of index entries in the first level index

A file organization that uses multi-level indexing is called indexed sequential file

How do we handle insertion in multilevel indexing?

Create an overflow file, store it there and merge it to the main file periodically.

The indexes are recreated when reorganizing.

Q) $Bfr_i = 68 \qquad r_1 = 442 \times 68$

# of second level blocks $= \left\lceil \frac{442}{68} \right\rceil = 7$

# of third level blocks $= \left\lceil \frac{7}{68} \right\rceil = 1$

∴ There are 3 levels

No. of block accesses : $3 + 1$

$= 4$