UNIT-6

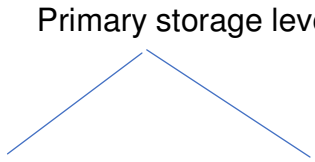
Introduction:

In computer storage media basically there are two types of storages:-

1.Primary storage:-The storage media is directly opened by the CPU.

EX: Main memory , it has small size and limited storage capacity but provides faster access to the data& cost is very high.

2.Secondary & Tertiary Storage:- Hard disk drives are classified as secondary storage ,where as removable media (EX: pendrive) are considered as tertiary storage.Data in these storages cannot directly processed by CPU.First the contents are copied to main memory and then processed cost is less and provide lower access to the data then primary storage devices.



Static random access memory(SRAM) (or) [cache]

Dynamic random access memory(DRAM) (or) [main memory]

->CPU uses cache memory to speed up the execution of the programs.

->DRAM provides the main work area for CPU for keeping programs & data.

-> Advantages of DRAM is low cost and drawbacks are volatility and lower speed compared to SRAM.

Secondary and tertiary storage includes

->disks,

->mass storage in the form of CD-ROM (compact disk read only memory).

-> DVD's (Digital video disk).

->tapes at least expensive.

Storage capacity is stored in kilo bytes (1 kb or 1000 bytes)

Mega bytes(1MB or 1 million bytes)

Giga bytes(1GB or 1 billion bytes)

Tera bytes(1000 GB)

- ➔ Programs execute and reside in DRAM.
- ➔ Generally large payment data base reside son secondary memory and portion of it are read into buffers of main memory when required .In some cases the entire

database is kept in main memory leading to main memory database .these are useful in real time applications that need form response time.
EX:-Telephone switching application.

➔ Between DRAM and magnetic disk another form of memory is called flash memory is there .this is non volatile.These are highly density and high performance memory using EEROM.

➔ Advantagesof flash memory is fats access .Disadvantages : if we want to delete data entire block must be erased .These are used in cameras MP3 players.

➔ CD-ROM disks store data optically and are read by LASER .These contain pre recorded data that cannot be overridden .we cannot erase the data we can write once read many (WORM) .

➔ DVD's will store 4.5 GB to 15 GB.

➔ Optical juke box memory use an array of CD-ROM platters ,which are loaded on to the drives on demand .capacity of these is in the 100's GB but retrival time slower than magnetic disks.

➔ Magnetic tapes are used for achieving and backup storage of data.

➔ Transient data structures presise for only a limited time during program execution .

➔ Most databases are stored perminantly on magnetic disk ,due to following reasons.

1. Databases are too large to ft in the main memory.
2. Secondary storage devices like disks are nonvolatilestorage but main memory is volatile storage .
3. Cost of storage per unit is less than for secondary storage than primary storage .

➔ Magnetic tapes can be stored for storing databases because costs are less than compare d to disks ,but accessing data on tape is quite low.

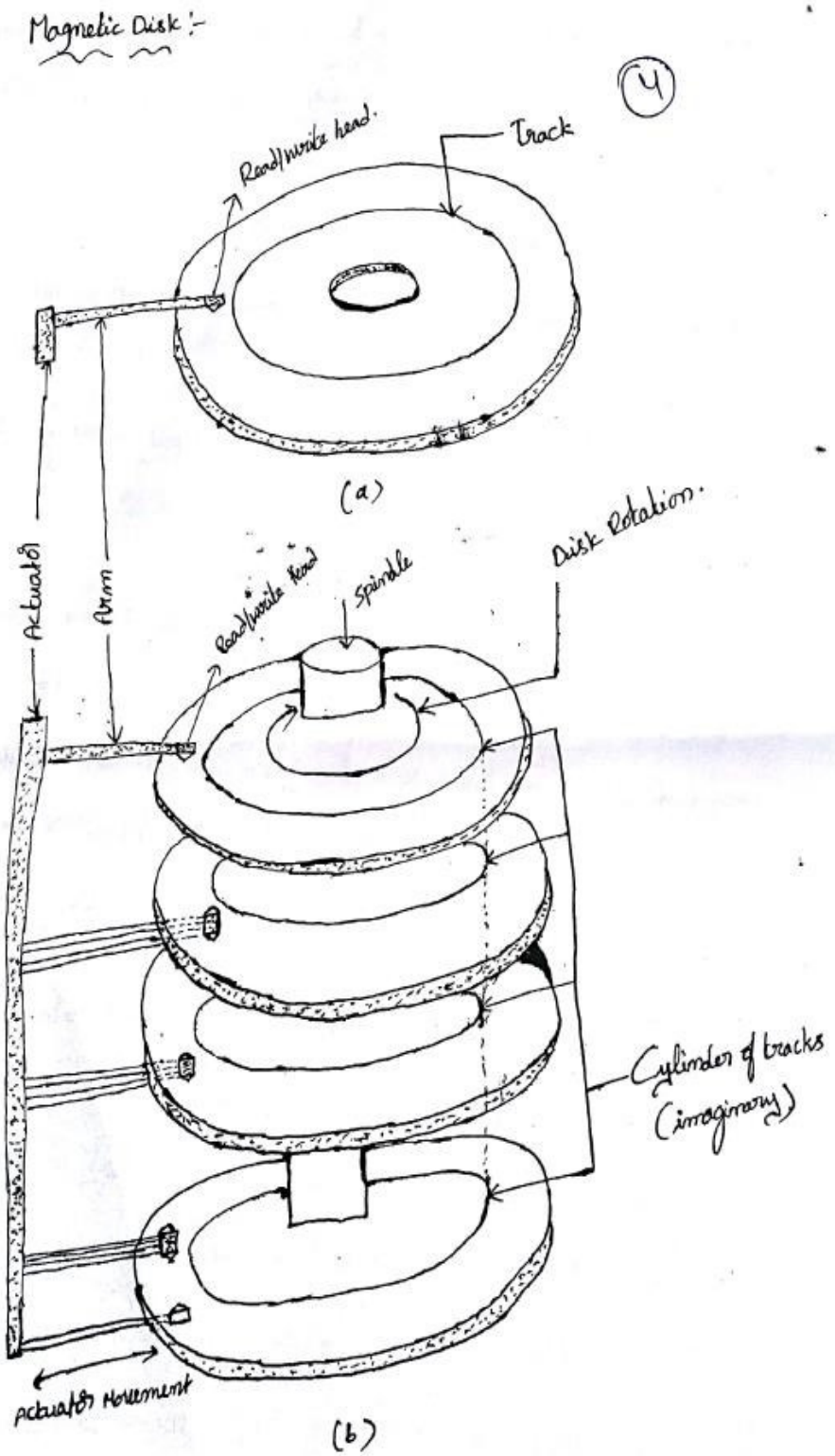
➔ Data stored on tapes is "off line" So an automatic loading device required to make the data on online .In constant disk are "online" devices can be accessed directly at any time.

➔ Databases applications need only a small portion of data base at a particular point of time for processing .When required that portion will be loaded from secondary memory to main memory for processing and re written to the disk if the data is changed .

➔ The data on the disk is organized as files of records .Each record is a collection of data base of entities there attributes and relation ship.

➔ Records must be stored on the disk in such an way that when need than has to be accessed efficiently.

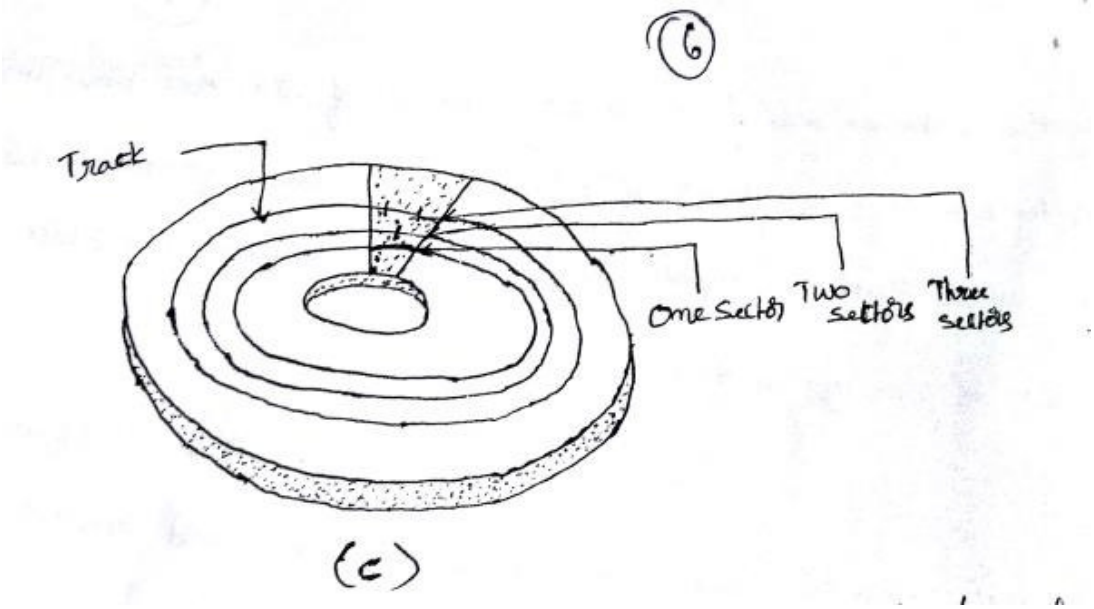
Magnetic disk :



Magnetic disks are used for storing the large amount of data .Most basic unitof data on the disk is a single bit of information To code information bits are grouped into bytes .based on thee computer the bit sizes are generally 4 to 8 bits .capacity of a disk is given by no of bytes it can store .

As shown in fig(a) all disks are made of magnetic material shaped as a thin circular disk ,and protected by a plastic areascyclic cover . If a disk stores in formation in one side the it is called "single sided disk" .If both surface are used then it is called "double sided disk ". To increase storage capacity disks can be assembeled to get more surfaces as shown in fir(b). this is known as "disk pack". The information is stored on the disk surface in the form of concentric circles of small width . Each circle is having different diameter .Each circle is called a "track" .In disk packs tracks with some diameter on different surfaces is called a "cylinder" . because of the shape they would form it connected in space The data stored on one cylinder can be retrived much faster.

The number of tracks on a disk ranges from a few hundred to few thousand and the capacity of each track range from lens of k bytes t150 k bytes . A track is divided into smaller blocks or sectors The division of a track into sectors is hard - coded on disk surface and can not be changed .



Sectors of the tracks will subtents fixed angles at the center of circle A track may be divided into any no . of sectors .For Ex: Some cylinders may have one sector per track and some may have two per track and so on . While formatting disk the operating system sets division of tracks into equal sized disk blocks (or pages). Block size is fixed during initialization and can not be changed dynamically disk block size ranges from 512 and 892 bytes .sectors sub divided into blocks during initialization blocks are separated by fixed sizes "inter block gaps" . these will have specially coded control information written during disk initialization .

A disk is a random access addressable device transfer of data between main memory and disk takes place in units of disks blocks . The hard ware address of the bock is given by a combination of a cylinder number track number (surface number with in the cylinder in which the track is located) and block number (with the track) blocks are held by contiguous resourced area in main storage called buffers for the "read " command the bock from disks is copied into buffer where as for a write command the contents of buffer are copied into the disk block .some times several contiguous blocks may be transferred as a unit known as "cluster ". in this case the buffer size is adjusted as the no. of bytes in the cluster.

The mechanism that read or write s a block is the disk read /write head which is part of a system called disk drivers .A disk pack is mounted in the disk drive ,which includes a monitor that rotate the disks .read/write head includes an electronic component attached to a mechanical arm .once read/write head is positioned on the right track and the block specified is the block addressed moves under the read/write heads the electronic component of the read/write head is activated to transfer the data .some disk units have fixed read/write heads with as many heads as there are tracks .these are called fixed head disks , where disk units with an actuator are called movable head disk .A disk controller embeded in the disk drive , controls the disk drive and interface it to the computer system. The disk controller first mechanically position the read/write head on the correct track the time required to do this is called "seek time ". typical seek times are 7 to 10 msec on desktops and 3 to 8 ms an server . There is another delay called "rotational delay " or "latency" while the beginning of the desired block rotated into position under read/write head .Finally some additional time is needed to transfer data that is called the "block transfer time ". Time needed to locate and transfer a disk block is in the order of milliseconds (EX: 9 to 60 msec).

Magnetic tape:

Disks are random access secondary storage devices because an arbitraty disk block may be accessed at random once we specify its address .magnetic tapes are sequential access device .To n^{th} block on the tape first we must be scanned the preceding n-1 blocks .Data is stored on real of high capacity magnetic tape ,some what to audio tapes or video tapes ,A tape drive is required to read/write the data to a "tape reel".

A read/write head is used to read or write data on tape .Data record son tape also stored in block as like disks . The main characteristics of a tape is its requirement that

we access the data blocks in “sequential order” .To get a block in the middle of s rel tape the tape is mounted and then scanned until the required block gets under the read /write head so tape access is slow and mostly not used for online storage of data .Tapes serve a very important function called “backing up ” the database .The reason for backing up is to keep copies of backup files in case of data is lost due to disk crash .So disk files are copied periodically tot tape .EX: airline reservation system >database files that are seldome (rarely) used or are out dated but required for historical record keeping can be achieved on tapes,now a days ,backing up enterprises databases so that on transaction information is lost is a majorundertaking.

Unit – 6

Records

Data is usually stored in the form of records. Each record contains a collection of related data values or items corresponds to a particular field of the record .Records usually describe entities and their attributes.

Ex: employee record represents an employee entity and each field value in the record specifies some attributes of that employee, such as name date of birth, salary. A collection of field names and collection of field names and their corresponding data types constitutes a record type or record format definition. A data type specifies the types of values a field can take. Data types include numeric (integer, long integer, floating point), string of characters(fixed or varying), Boolean (0,1 or true or false) and date and time. An integer my require 4 byte, for long integer 8 bytes, real number 4, Boolean 1, date 10,etc. To store image, video files, etc. BLOB (Binary Large Object) data type is used.

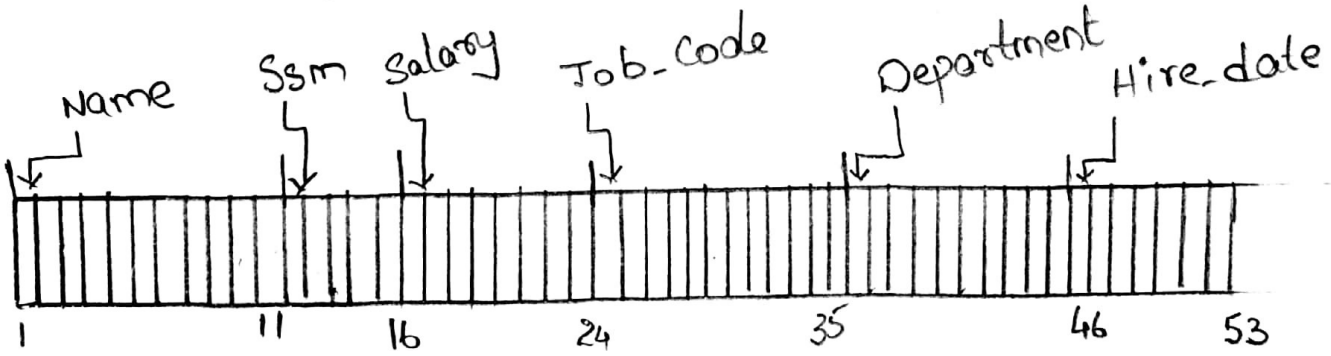
Files, Types of Records

A file can be defined as sequence of records. All records in a file are of the same record type.

Fixed length records: If the records of a file has exactly same size (in bytes) then the records are known as fixed length records.

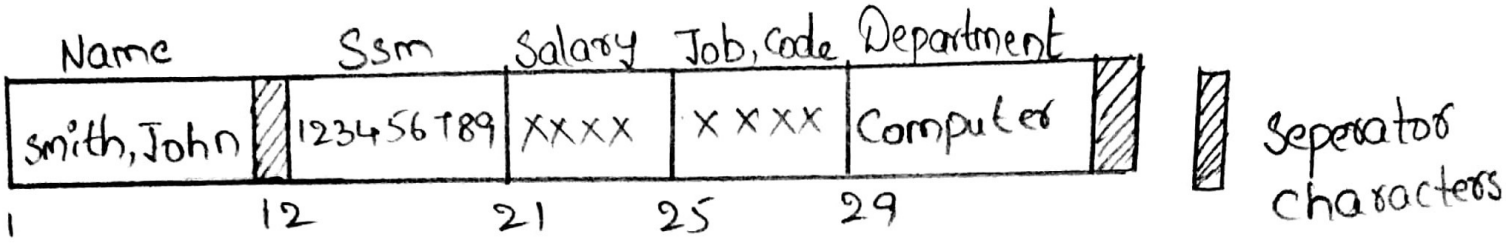
Variable length records: If the records of a file have different sizes those record are known as variable length records.

Ex: for fixed length records

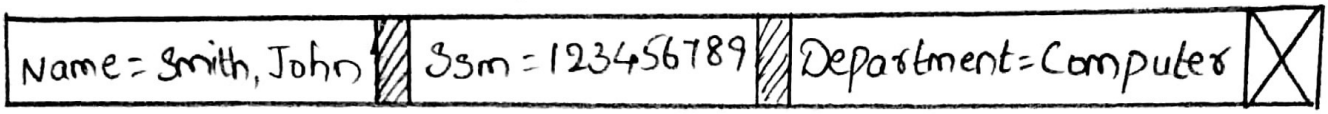


The above record is of size 52 bytes. Every record has the same fields and the field lengths are fixed. In this type system can indentify the starting byte position of each fixed form from the starting. So accessing of field values is easy.

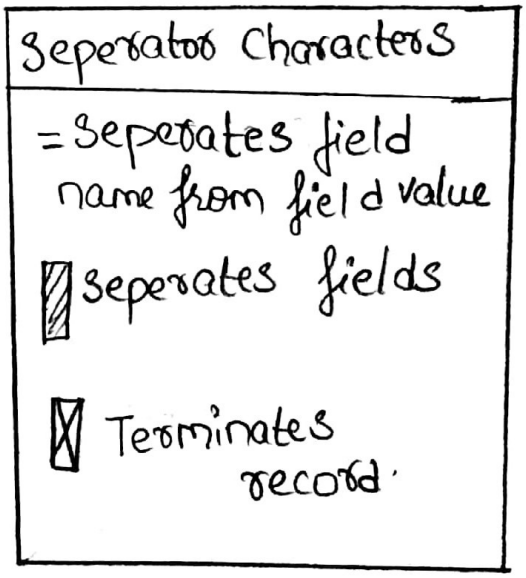
Ex: for variable length records



fig(a)



fig(b)



For variable length fields, each record has a value for each field, but we do not know the exact length of some field values. To determine the byte that represent for each field value we can use special separator character (?,%.$) which do not appear in any field value.

A filed of records with optimal fields can be formatted in different ways. If the total number of fields for record type is large , but the number of fields that actually

appear in a typical record is small, we can include in each record a sequence of <field_name, field_value> pairs rather than just the field values as shown in fig(b)

Spanned vs unspanned Records

A block is the unit of data transfer between disk and memory. So the records of a file must be allocated to disk blocks. When the block size is larger than the record size , each block can contain so many records. There may be some files which can have large records that cannot fit in one block.

Suppose that the block size is B bytes. For a field length records of size R bytes. Then bfr=[B/R] gives the record per block and called as "blocking factor" for the file. In general

R may not divide B exactly. So e have some un used space in each block i.e.,

B-bfr*R)byytes

To use this unused space, we can store part of a record in on block and the rest on another. A pointer at the end of the first block points to the block having the remaining part of the record, if it is not in the next consecutive block on disk. This type of organization is called "spanned organization" .If records are not allowed to cross block boundaries, that is called unspanned organization. This is used with fixed length records having B>R

Record organization fig:

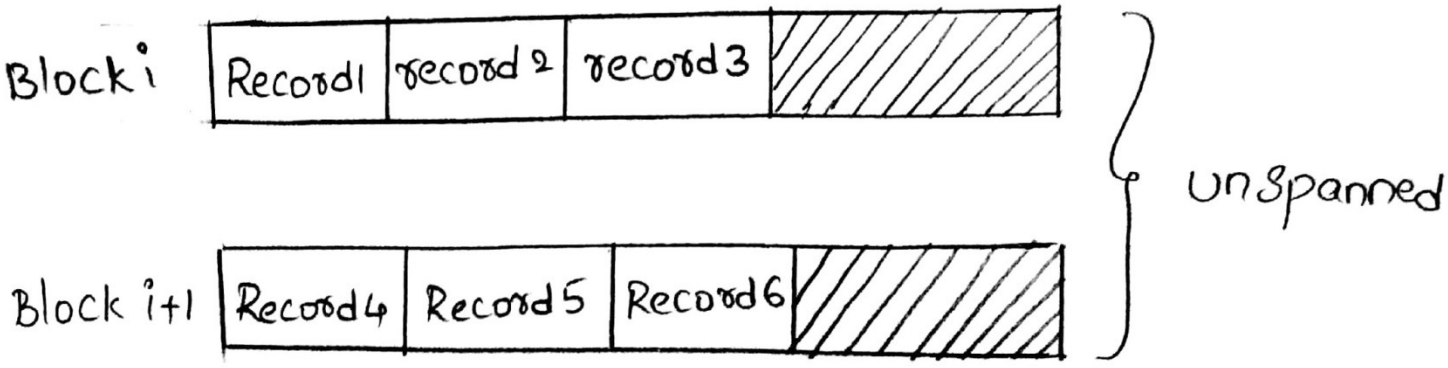


fig (a)

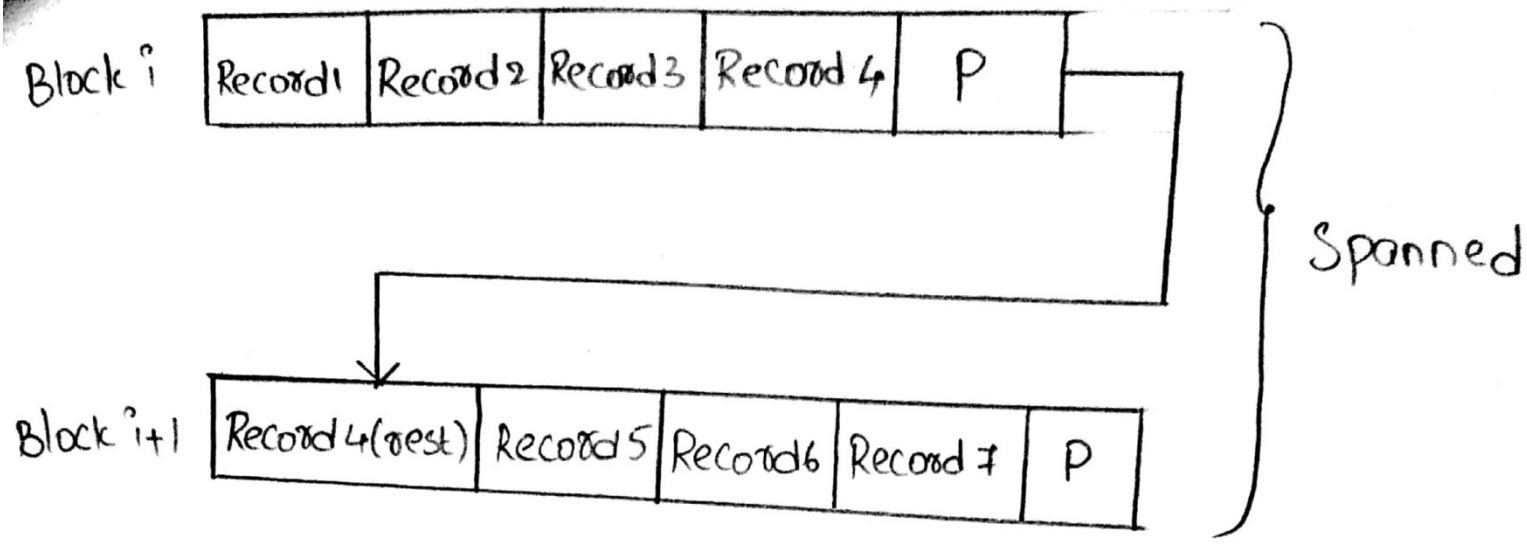


fig (b)

Operations on fields: operations on fields are usually grouped into retrieval operations and update operation. Retrieval operations do not change any data in the file. Update operations change file by insertion or deletion of records or updation of field values.

Operations:

Open: prepare the file for reading and writing. Allocates buffer to hold file blocks and retrieves the file header. Sets the file pointer to the beginning of the file.

Reset: sets the file pointer of an open file to the beginning of the file.

Find(locate): Depending upon search condition searches the first record. Transfer the block containing that record in the buffer and it becomes the current record.

Read (or get): Copies the current record from the buffer to a program variable in the user program. This command also advance the current record points to the next in the file, which may read the next file block from the disk.

Find Next: Searches the next record in the file that satisfies the search condition. Transfer the block containing that record into main memory buffer. This record becomes the current record.

Delete: Deletes the current record and updates the file on disk.

Modify: Modifies some field values for the current record and updates file on disk.

Insert: Inserts a new record in the file by locating the block where the record is to be inserted, transferring that block into main memory buffer, writing the record into the buffer and return the buffer contents(modified) into disk.

Close: Completes the file access by releasing the buffers and performing cleaning up operations.

The above operations are called "record-at-a-time" operations, because each operation to a singe record. These are some additional "set-at-a-time" operations to deal with multiple records at a time.

Find All: Locates all the records in the file the search condition.

Find n(Locate n): Searches for the first record that satisfies search condition and then continuous to locate the next n-1 records satisfying the same condition. Transfers the block containing the n records to the main memory buffer.

Find Order: Retrieves all the records in the file in some specified order.

Reorganize: Starts the reorganized process. Because some file organizations require periodic reorganization.

File organization: It refers to the organization of the data of a file into records, block, and access structures. This includes in which way records and blocks are placed on the storage medium and interlinked.

Access Method: It provides a group of operations (all the above) that can be applied to a file.

TYPES OF FILE ORGANIZATIONS

1. Files o unordered records(Heap files): This is simplest and most basic type of organization. In this the records are placed in the file, the order in which they are inserted. So new records are inserted at the end of the file. Such an organization is called "heap" or "pile fie".
   In this organization inserting a new record is very efficient because last disk block is copied into buffer, new record s added, then that block is rewritten back to disk. The address of the last file block is kept in the file header. In this searching for a record using any search condition involves "linear search" through the file block by block. It is an expensive procedure. If only one record satisfies the search condition, then an average a program will read into memory and search half the file blocks before it fields the records. For file off b blocks, this require searching of b/2 blocks on average. If no or several records satisfy search condition, the program must read and search all bb blocks in the file.

To delete a record, a program must first find its block, copy the block into a buffer, delete the record from the buffer and finally rewrite the block back to the disk. This leaves one unused space in the disk block. So if we delete large no. of records that results in wasted storage space.

Another technique used for record deletion is to here an extra byte or bit, called deletion marker, stored with each record. A record is deleted by setting the deletion marker to a certain value. The value of market indicates a valid (not deleted) record. Whenever the block is accessed only these records will be taken into consider. Both of the deletion techniques require periodic reorganization of the file make free the unused space of the deleted records. We can use either spanned or unspanned organization for an unordered file, and it may be used ith either fixed length or variable-length records.

Files of ordered Records (sorted files): We can physically order the records of a file on disk based on the value of one of their files called the "ordering field". This leads to an "ordered" or "sequential" file. I f the ordering field is also a key filed of the file (key field: a filed to have unique value in each record), then the field is called the "ordering key" for the file. Ordering records have some advantages over unordered files.

1. Reading the records in order of the ordering key value becomes extremely efficient because no sorting is required.
2. Finding the next record from the current one requires no additional block access because the next record is in the same block unless the current record is the last one in the block .
3. Using a search condition based upon the value of an ordering key field results in faster access when the binary search is used, which gives an important over linear searches. A "binary search" for disk files can be done on the blocks rather than on the records. Suppose that the file has b block numbered 1,2,3,.....,b. the records are ordered by ascending value of their ordering key field, and we are searching for record whose ordering key field value is k. Assuming the disk address of the file header, the binary search can be described by algorithm shown below. A binary search usually access $\log_2(b)$ bocks.

Algorithm:

```
l<- 1, v<-b;(b is the o. of file blocks)
while(v>=l) do
begin i<-(l+v) div 2;
 read block I of the file into the buffer;
 if k<(ordering key field value of the first record in block i)
 then v<-i-1
 else if k>(ordering key field value of the last record in block i)
 then i<-i+1
 else if the record with ordering key field value=k is in the buffer
 then goto found
 else goto notfound
  end;
  goto notfound;
```

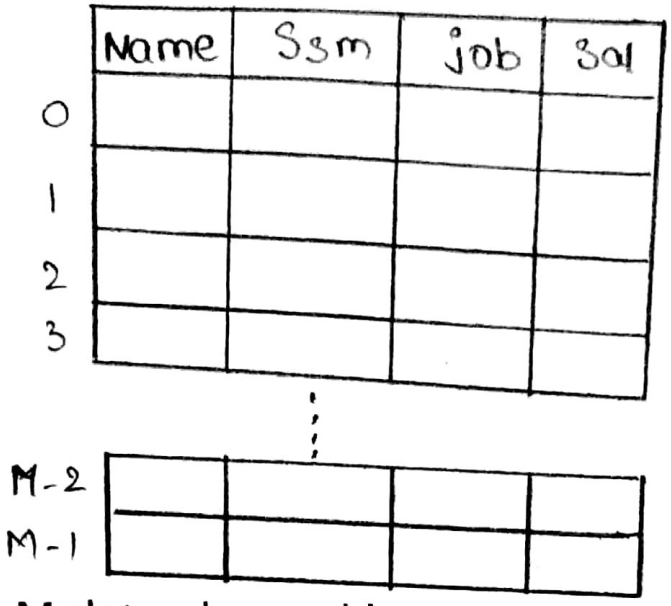
A search condition including >,<,>=,<= on the ordering field is quite efficient. Inserting and deleting records are expensive operations for an ordered file because the records must remain physically ordered. To insert a record, we must find its correct position in the field based on is ordering field value. After that makes space in the file to insert the record in that position. For deletion the problem may be less compared to insertion if we make use of deletion markers and periodic re organization.

Hashing Techniques:

Another type of primary file organization is based on hashing, which provides very fast access to records under certain search conditions. This organization is called as hash file organization. The search condition must be an equality condition on a single file called the "hash field". In most cases, the hash field is also a "key field" of the file, in which case it is called the "hash key". The idea behind hashing is to provide a function h, called "hash function" or "randomizing function", which s applied to

the field value of a record yields the address of the disk block in which the record is stored.

Internal Hashing: For internal files hashing is typically implanted as a "hash table" through the use of an array of records. Suppose that the array index range is from 0 to M-1 as shown in the below fig.

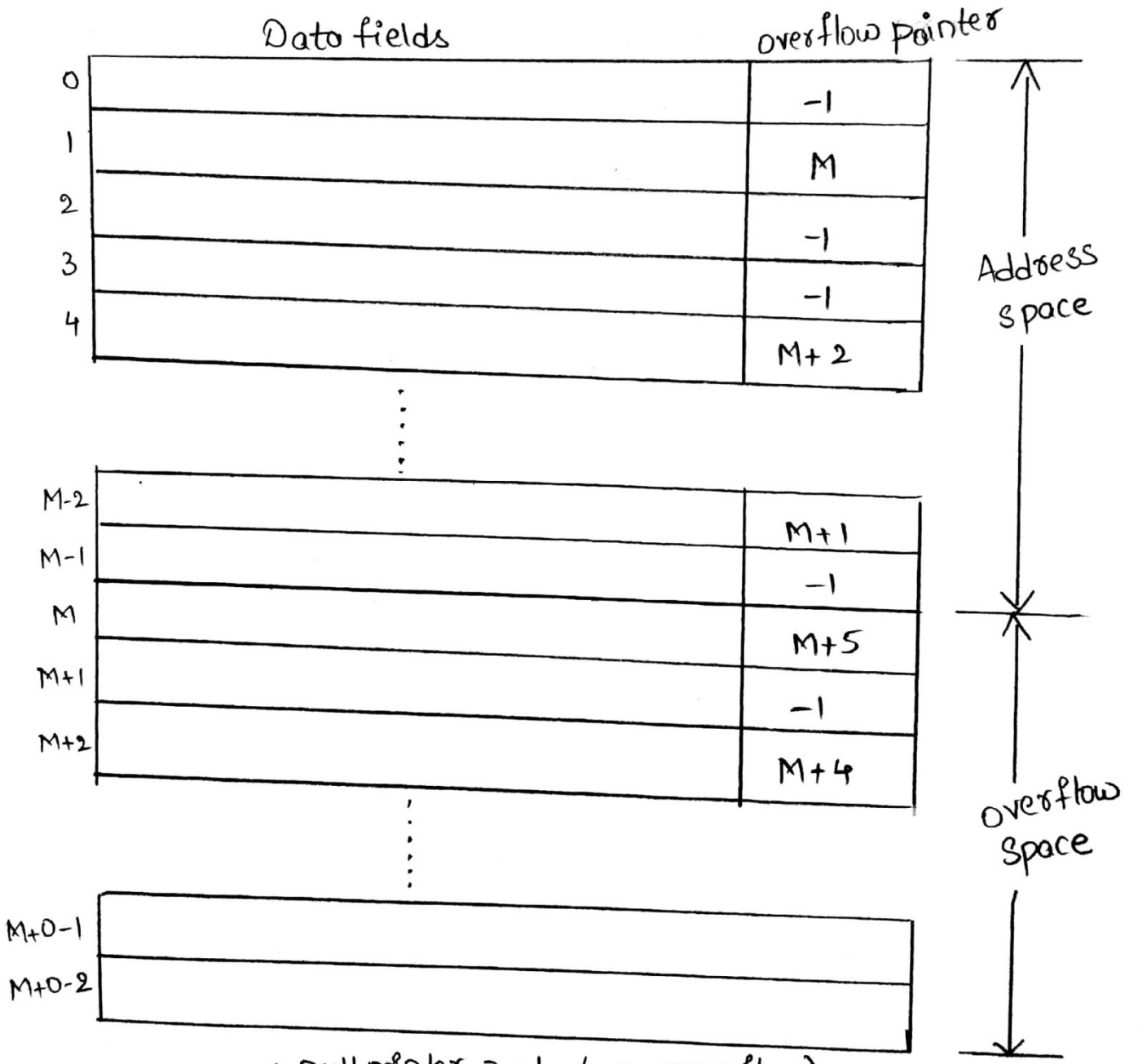


We have M slots whose address corresponds to the array indexes. We choose a hash function that transforms the hash field value into an integer between o and M-1, one common hash function is h(k)=k mod(M), which gives the remainder of an integer hash field value k after division by M; this value is then used for record address, called folding, includes applying an arithmetic function such as "addition" or a logical function such as "exclusive or" to different portions of the hash field value to calculate the hash address. The problem with most hash functions is that they will not generate distinct values to hash to distinct address. The no. of possible values a hash field can take is called "hash field space". Generally this is much larger than the "Address space" (no. of available address for records)

A collision occurs when the hash field value of the record hashes (coinsides) to an address that already contains different record. In this situation, we must insert the new record in some other position, since its hash address is occupied. This process of finding another position is called "collision resolution". Some o the collision resolution methods are:

Open addressing: The program checks for the positions until an unused (empty) position is formed proceeding from the occupied position by the hash address.

Chaining: For this method, various overflow locations are kept, usually by extending the array with a number of overflow positions. A additionally a pointer field is added to each record location. A collision is resolved by placing the new record in an unused overflow location and the setting pointer of the occupied hash address location to the address of that overflow location.

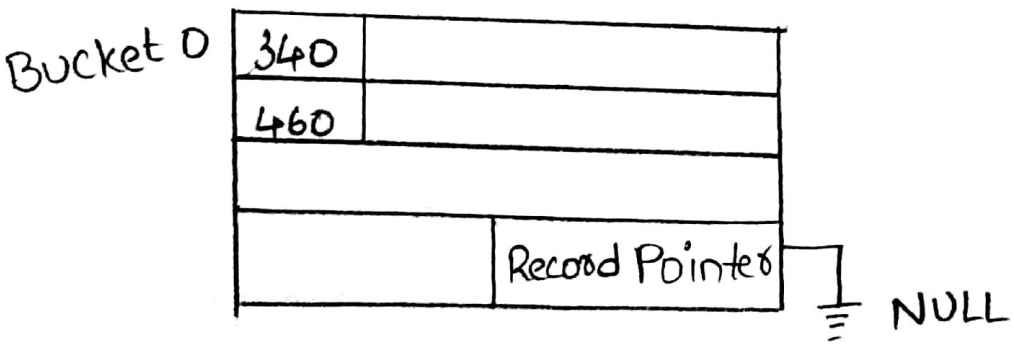


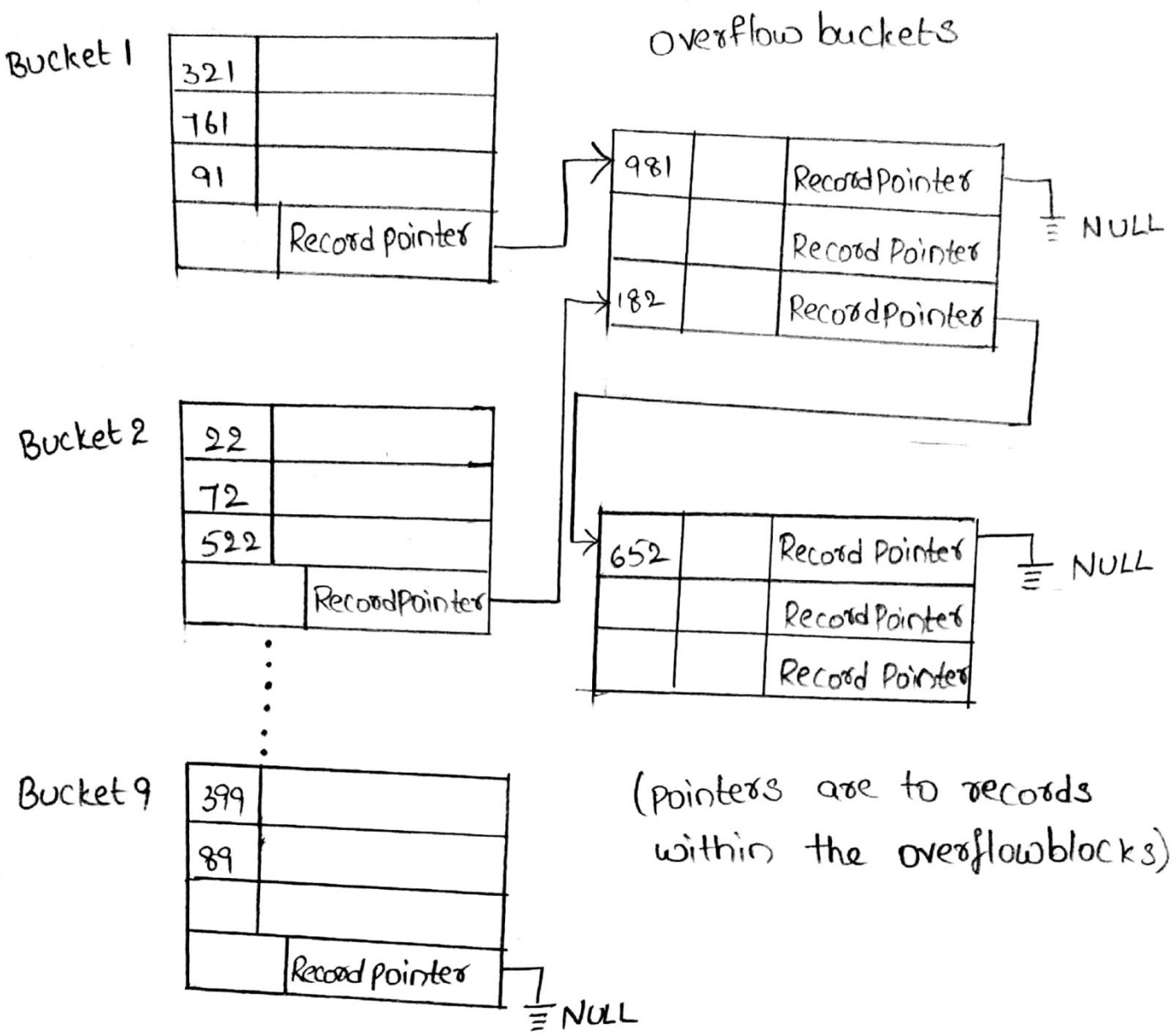
Multiple hashing: The program applies a second hash function if the first result in a collision and so on.

External Hashing for Disk File:

Hashing for disk files is called external hashing. According to the characteristics of disk storage, the target address space is made o buckets, each of which holds multiple records. A bucket may be on disk block or a cluster of contagious blocks. The hashing function gives relative bucket number, rather than block address. In file header one table is maintained to convert the bucket into corresponding block address.

The collision is less with buckets, because how many records a bucket can take, those many records only we will fit into a bucket. However we must take care about the case where a bucket is filled and a new record hashes to that bucket. We can use chaining process in which a pointer is maintained in each bucket to a linked list of overflow records for the bucket as shown in below figure. The pointer in the linked list should be "record pointers", which includes both a block address and a relative record position with in the block.





Hashing provides the fastest possible access for retrieving an arbitrary Record given the value of its hash field. The above described hashing scheme is called "static hashing", because a fixed no. of buckets M is allocated. This is very serious drawback of dynamic files. Suppose we allocate M buckets for the address space and let on be the maximum number of records that can fit in one bucket, then at most (m * M) records will fit in the allocated space. I f the no. of records fewer than (m * M), so many collisions will result.

Hashing Techniques that allow Dynamic File expansion:

Major drawback of the static hashing scheme is that thee hash address space is fixed. Hence it is difficult to expand or shrink the file dynamically. This can be overcome by using two schemes.

1.Extensible Hashing

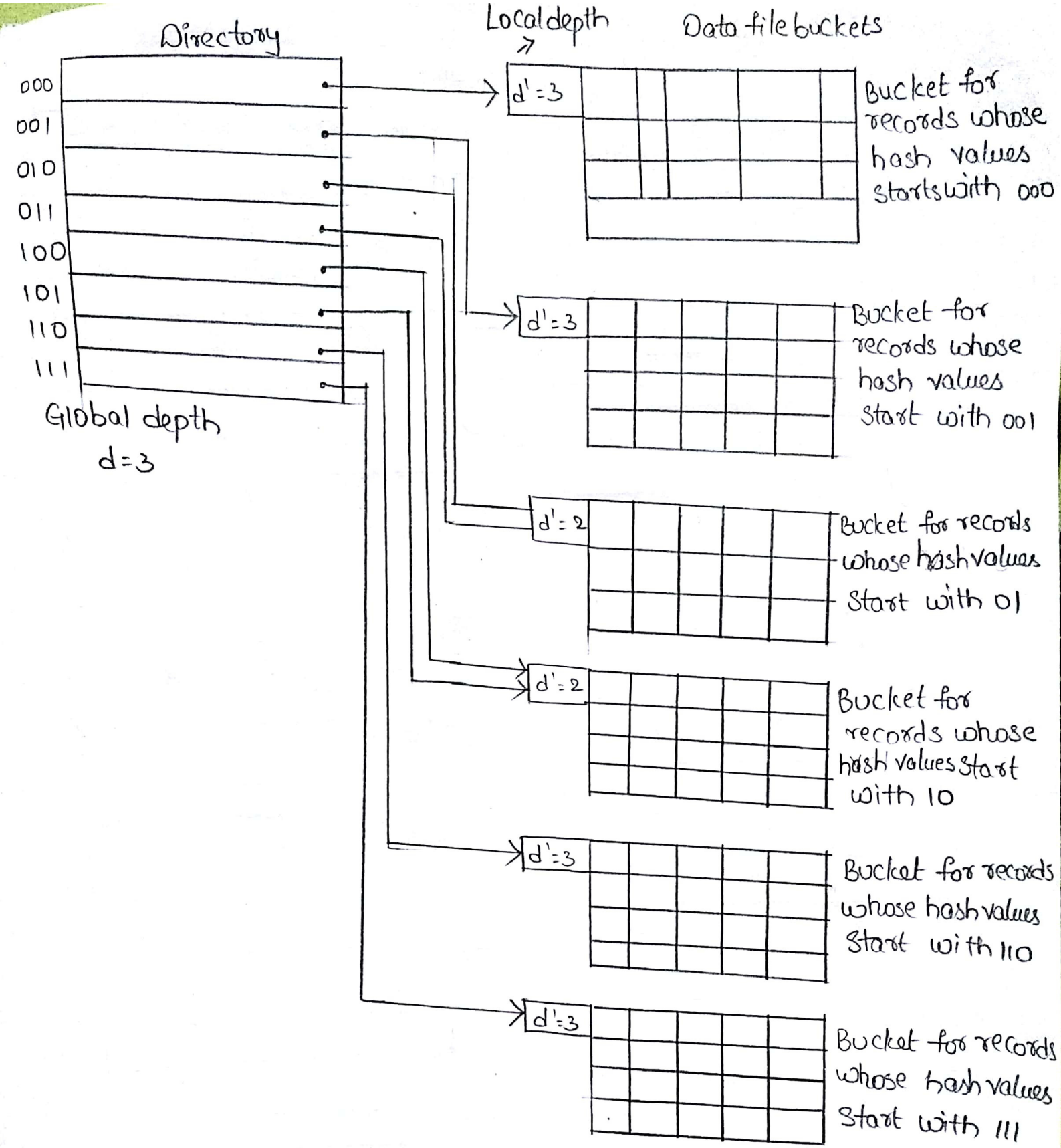
2. Linear Hashing

These schemes takes the advantage of the fact that the result of applying hashing function is a non negative integer, so we can represent it in a binary number. The access structure is built on the binary representation of the hashing function result. This is called as "hash value" of a record. Records are distributed among buckets based on the values of the loading bits in their hah values.

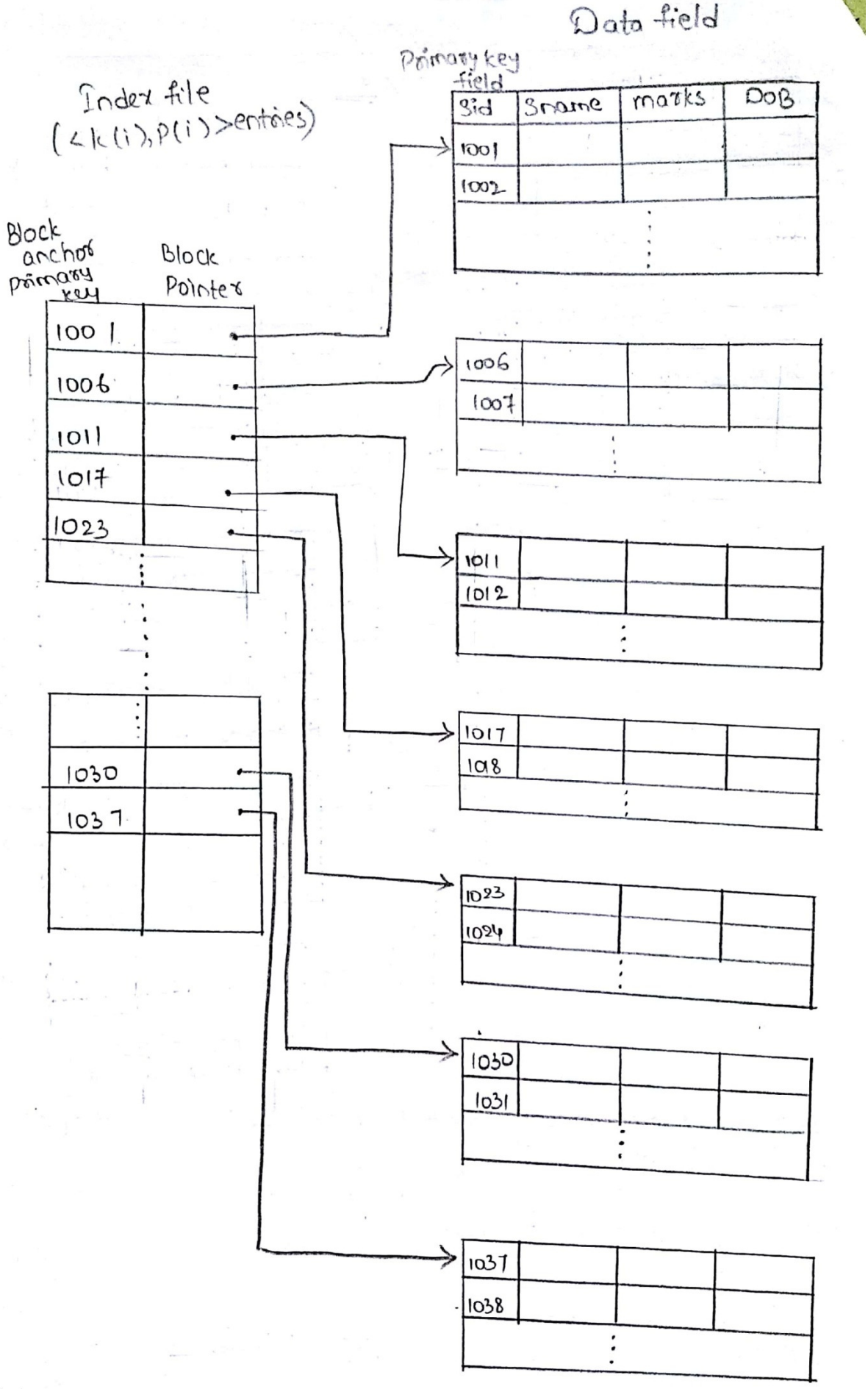
Extensible Hashing:

In this an array of $_2$d bucket address is maintained which is called as "directory", where 'd' is the global depth of the directory. The integer value corresponding to the first (high-order) d bits of a hash value is used as an index to the array to determine a directory entry, and the address in that entry determines the bucket in which the corresponding records are stored. There may not be different buckets for each of the 2d directory locations. A "local depth d" stored with each bucket, specifies the number of bits on which the bucket contents are used. The below fig. shows directory with global depth d=3,

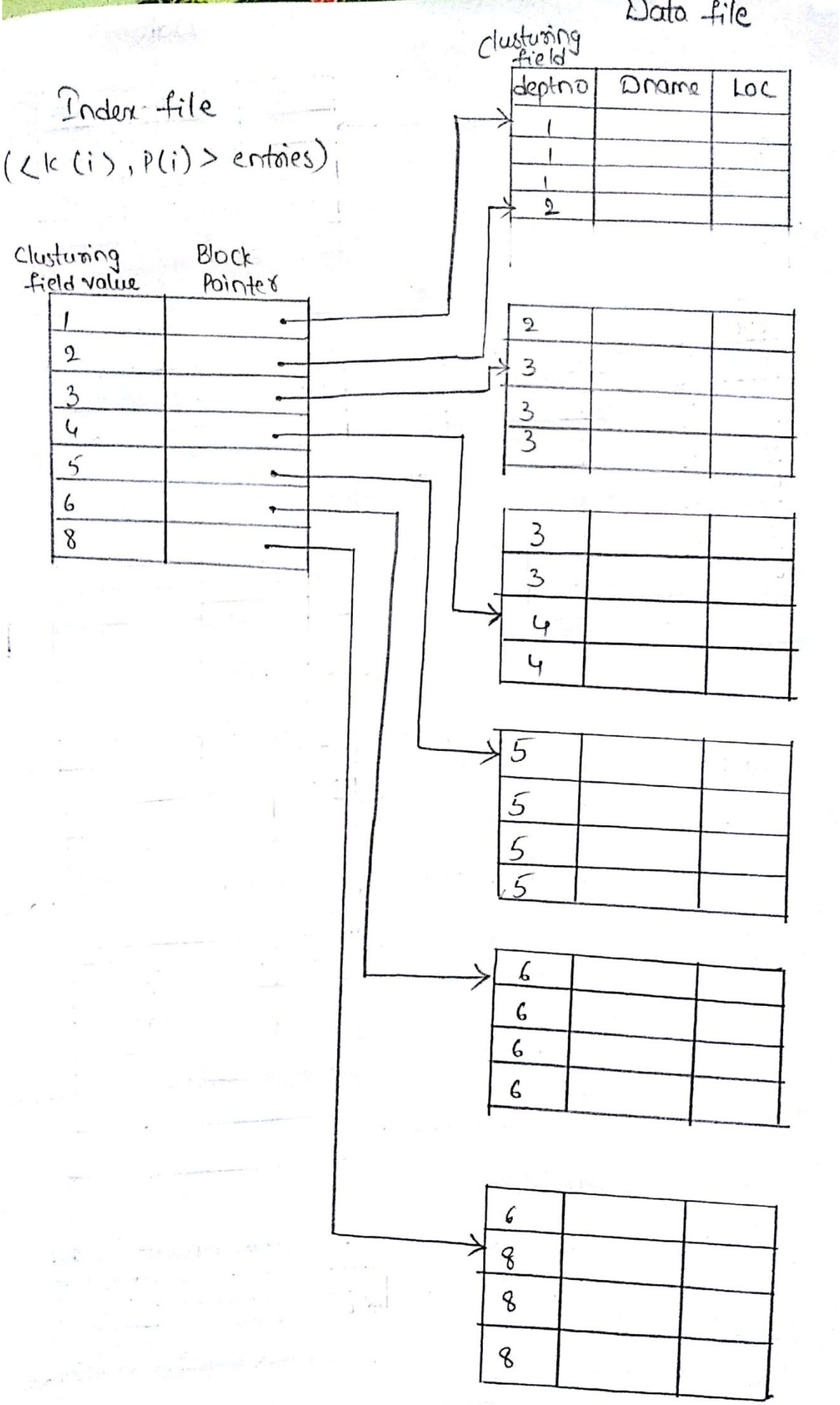
The value of d can be increased or decreased by one at a time so that doubling or halving the number of entries in the directory array. Doubling is needed if a bucket whose local depth 'd' is equal to the global depth d, overflows. Halving occurs if d>d'.



Suppose a new record causes overflow in the bucket whose hash values start with 01 – the third bucket in fig.



Primary Index on ordering key field (Sid)



Fig(i) clustering Index on deptno

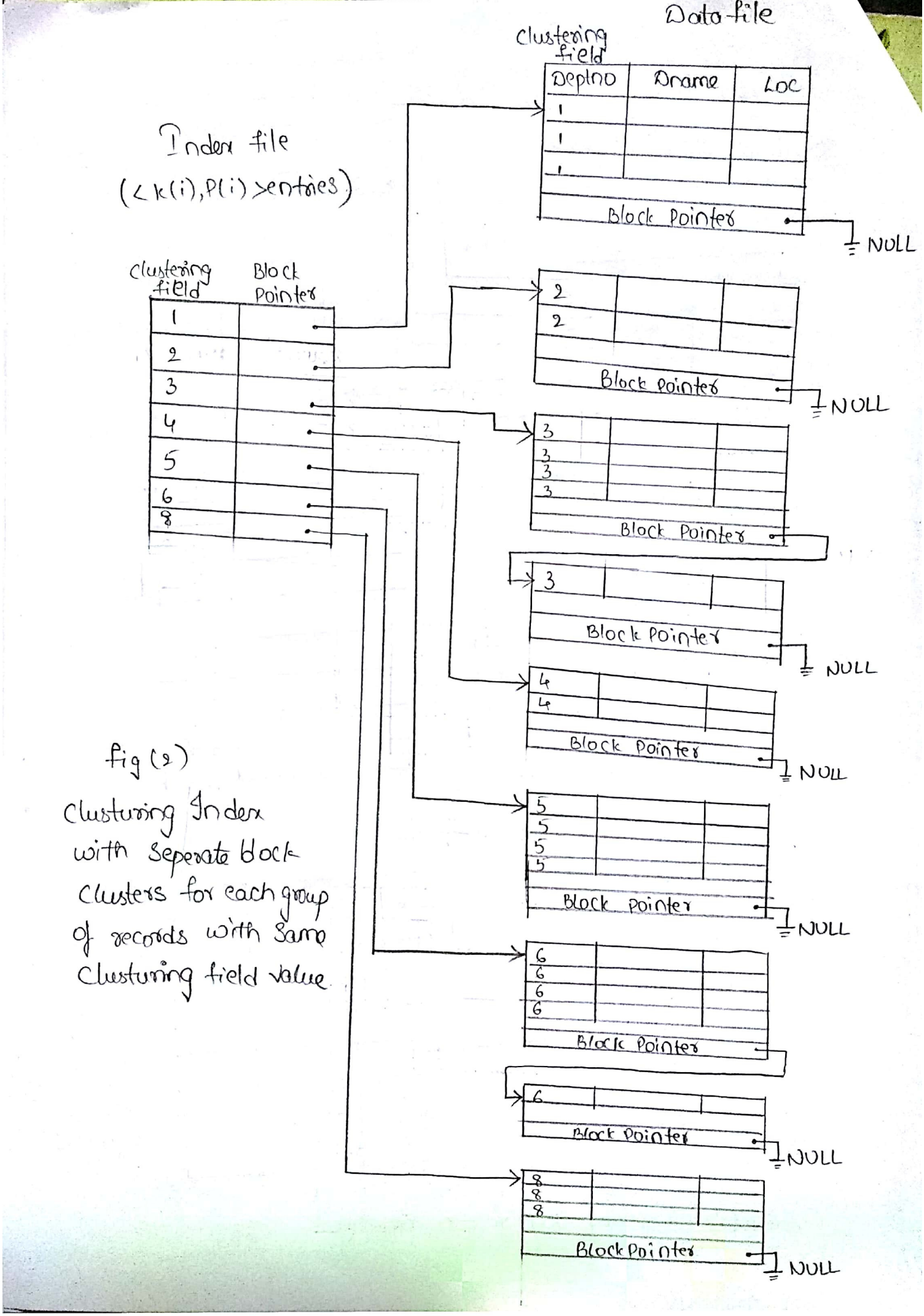
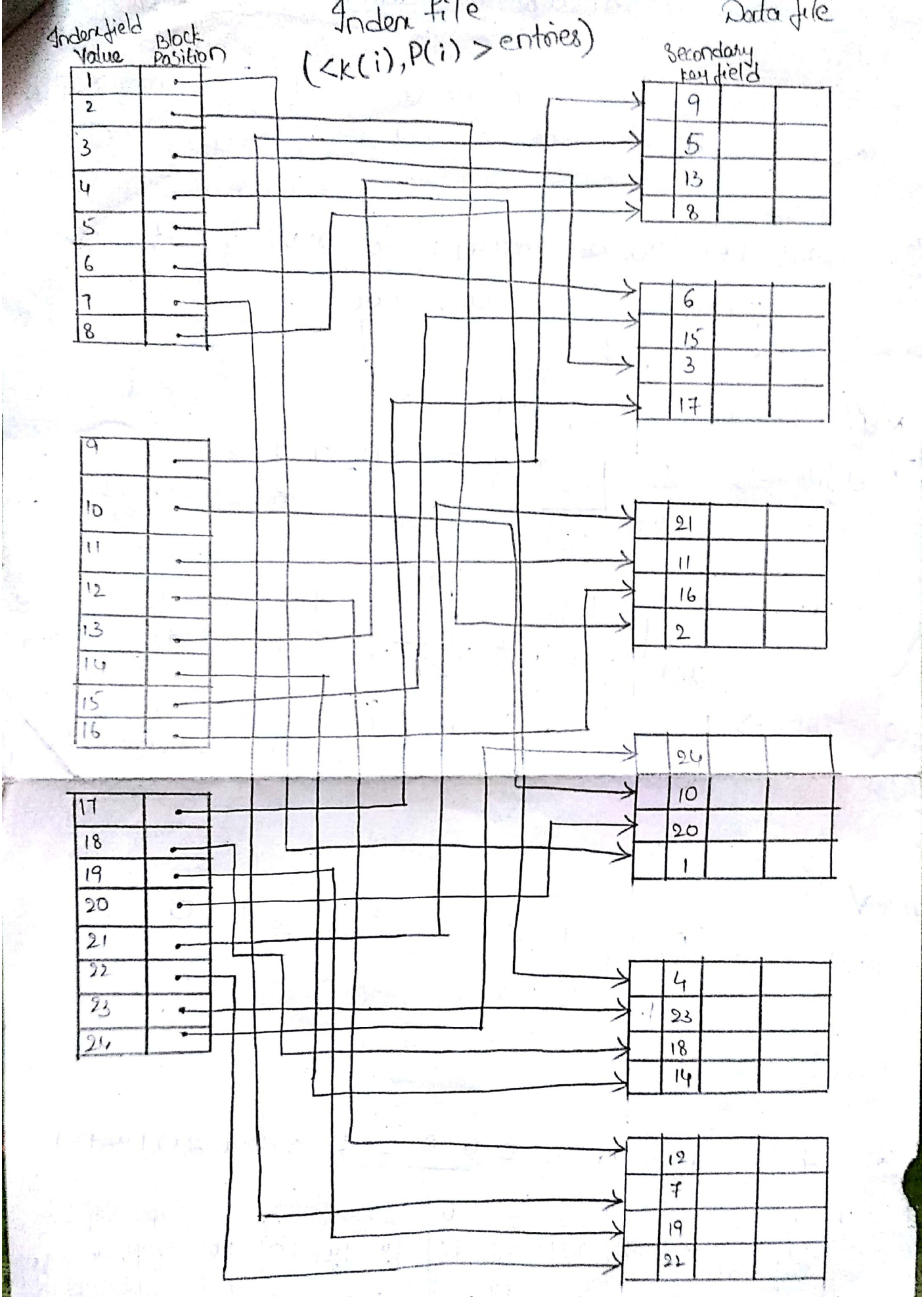


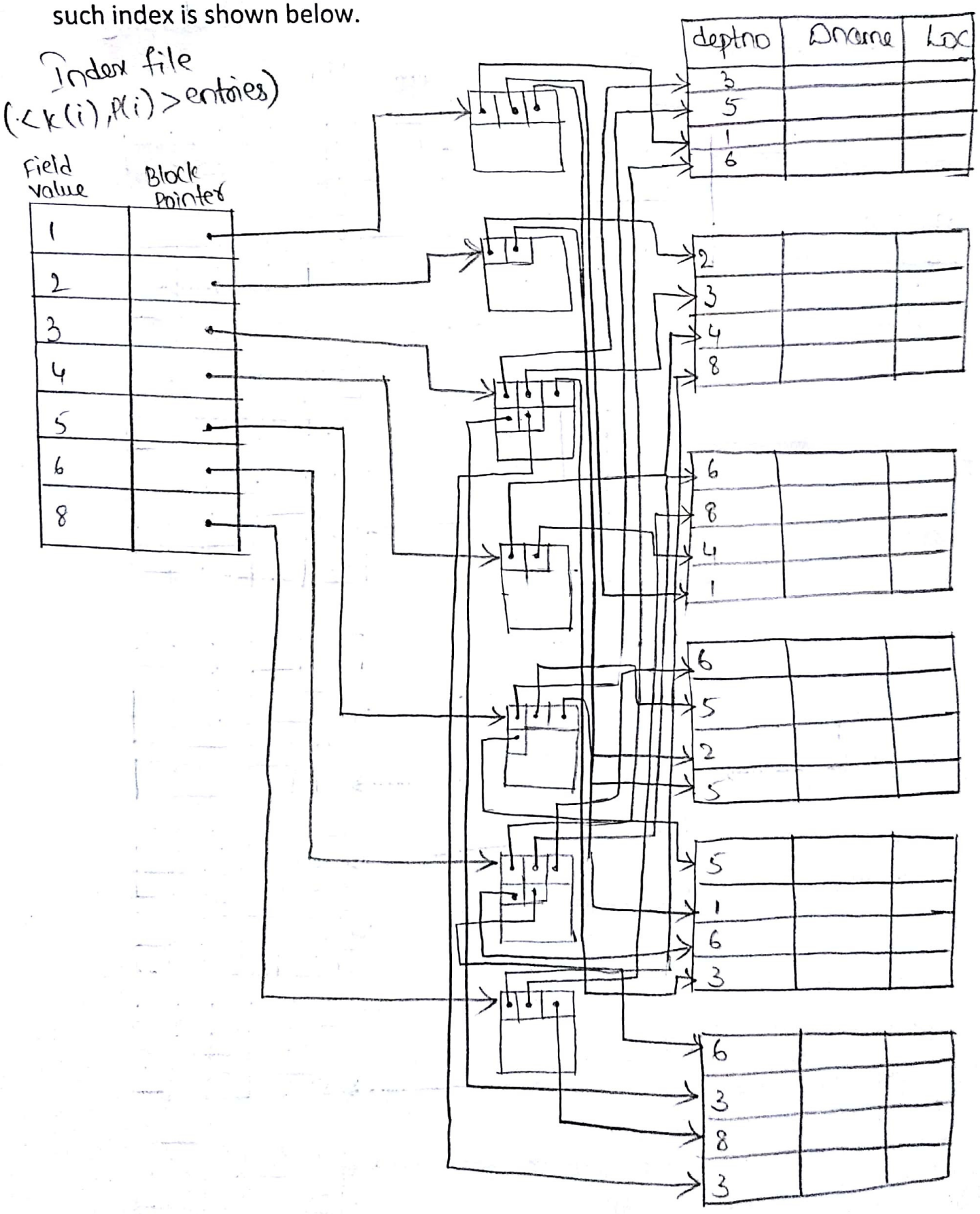
fig (2)

Clustering Index with Seperate block Clusters for each group of records with Same Clustering field value



Dense secondary Index on new ordering key field

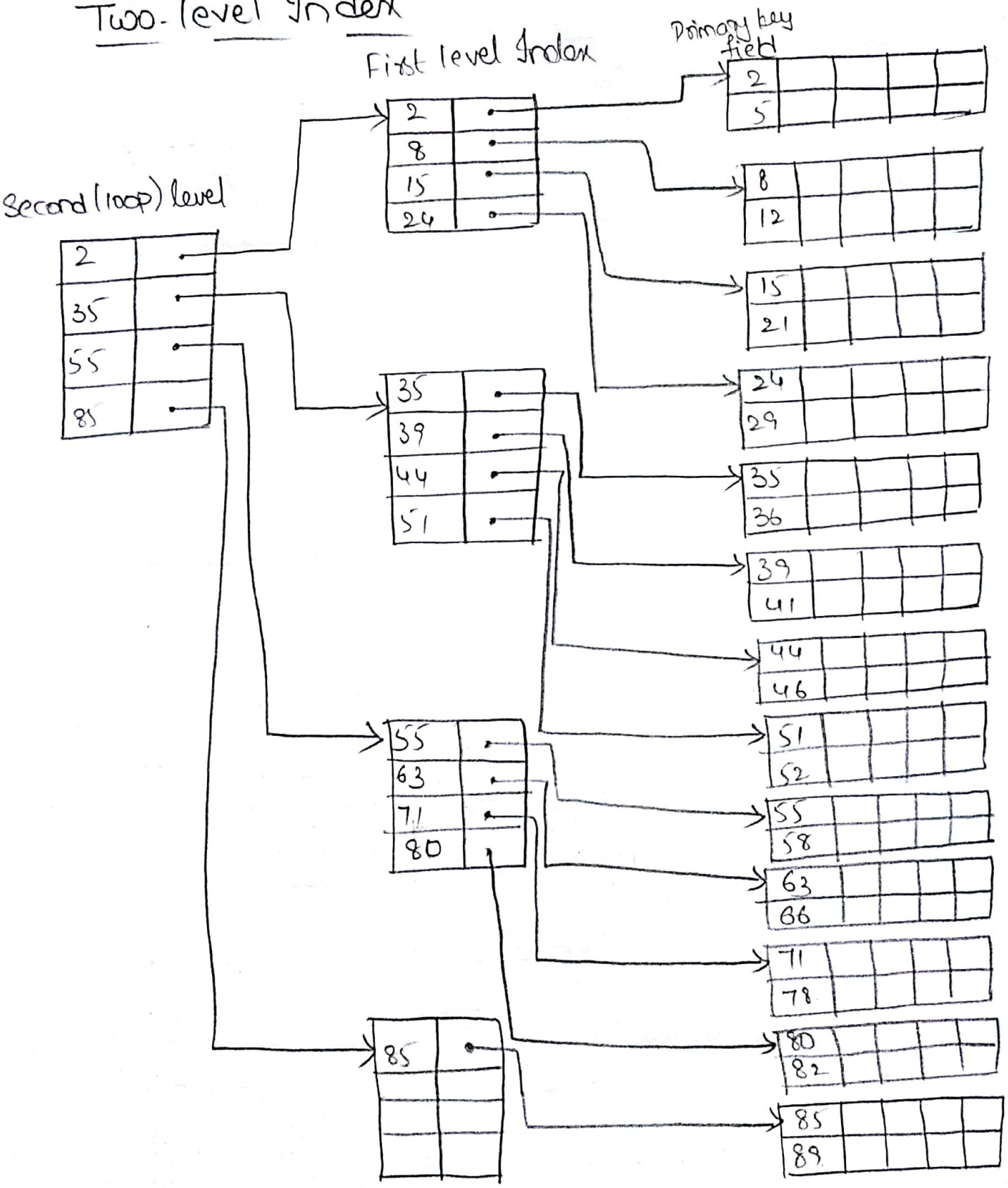
We can also create secondary Index on a non key field file. In this many records in data file can have the same value for the indexing field one way to implement such index is shown below.



Secondary index on a non key field implemented using one level of indirection so that index entries re of fixed length and h1ave unique field values

Datafile

Two-level Index



Two level Indexing