FILE SYSTEM

* A data file is a collection of record, mostly homogeneous.
* A record is a collection of related field.
* A field is a collection of data.

In every record there may be multiple fields but one field has to be unique to distinguish b/w records, called the primary key. All other fields are called secondary keys.

Modes of Accessing a file.

Input
Output
Input / Output

Classification of files through their functioning

may be classified in following types

1. Master file
2. Transaction file
3. Program file
4. Work file
5. Text file
6. Report file

MASTER FILE : Represents static view of some organisation of data at a particular time.
e.g In a factory you may be having a master file of all

workers and product produced or a master file of all inventory of the factory.

TRANSACTION FILE : Temporary in nature and is used to bring changes in Master file
If we have to update a master file, so first of all the data has to be changed is first collected in transaction file and then through transaction file master file is updated.

PROGRAM FILE : Source code in HLL or in assembly language is stored in a program file which may further be translated to object code by some translation program such as compiler incase of HLL and assembler in case of assembly language.

WORK FILE : Temporary nature. e.g output produced by one phase of program may be stored in a work file which will afterwards become i/p for some other phase of program.

TEXT FILE : Created using some text editor program such as Windowwrite, MS Word

REPORT FILE : Used for presentation and are generated by some report writing software such as powerpoint.

FILE ORGANISATION TECHNIQUES

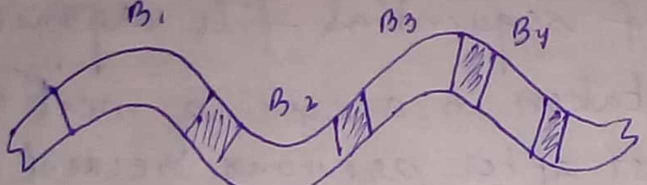
1. Sequential file Organisation
2. Relative file Organisation
3. Index seq. file Organisation
4. Multi-key file Organisation

In this file organisation technique all records are stored one after other in a sequence.
The advantage of sequential file organisation is that there is no time taken in accessing next record, as next record is just after previous record.
If record is stored at location 1 (start), then no time is taken in accessing it and if record is stored at the last location then full time is taken to access it. So on an average time taken to access a record is half the time taken to access the last record.
If we have to access nth record then we have to go through all previous (n-1) records, means we cannot access any record directly.

MEDIA USED FOR SEQUENTIAL file organisation

Magnetic tape is a suitable media for storing sequential files.
Records are stored in form of blocks. If blocking factor is 5, means every block has 5 records in it.
After every block there is some gap which is called inter block gap (IBG). This is provided because when stop command is given head stops reading but some part

of magnetic tape scrolls after the head gets directing
As magnetic tape take some time to become stationary
If we suddenly stop magnetic tape, there are chances
of breaking. Similarly when we give read cmd.
magnetic tape take some time. in attaining unifor
speed after that only head is activated. It means
again some part of magnetic tape will get scrolled
before head gets activated. To cover that IBG is
given.



IBG

RELATIVE FILE organisation
When there is a need to access a record directly
relative file organisation is used. In RFO there is
a predictive relationship b/w records key value.
(primary key) and physic address record.

R(Key value) → Address

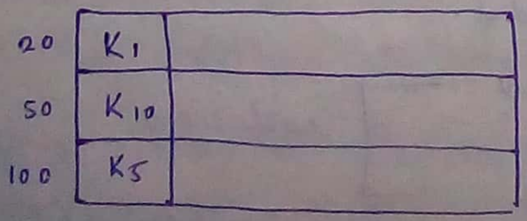
Addressing Technique used in RFO
following are 3 techniques used in RFO
① Direct Mapping
② Directory Look Up or Indexing
③ Calculations

DIRECT mapping : Key value itself becomes record's address / Relative address

* ADVANTAGES :
There is no time lost in calculating the address of the record as primary key itself is address / Relative address

* DISADVANTAGES :
This addressing technique has serious disadvantages which makes it practically impossible to use.
1. As key value itself is the address of record, so no logical / physical independence.
2. If a record's key value is changed then its physical location has to be changed as per the key value.
3. If a record is shifted from one part of memory to other so it's address will change, means it's key value also changes.
4. For every record we need to have a unique address and it has been observed all records are not required to be loaded in memory simultaneously.

DIRECTORY LOOK UP or INDEXING
In this technique whenever a record is loaded at an address a seperate directory / index of record's key value and address where it is loaded is maintained.

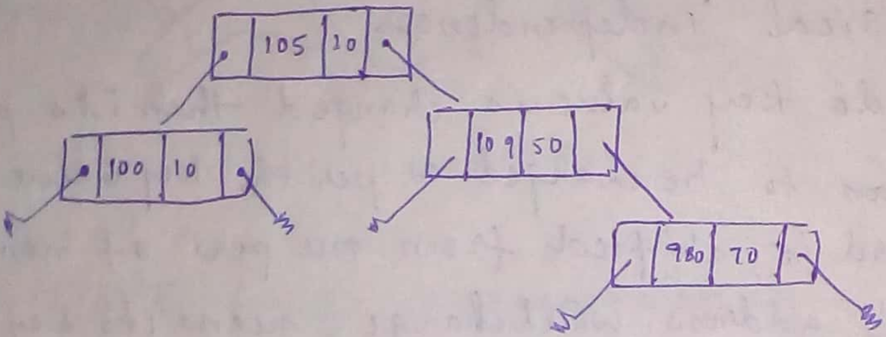


Records in memory

key	Add
K1	20
K10	50
K5	100
⋮	⋮

A directory or index may be in form of a table, which has to be searched sequentially or making search faster, the index may be in form of a BST or a B-Tree as an index.

	P. key	S. key . . .
20	105	
50	109	
10	100	
70	980	



BST as an index

CALCULATIONS

One way to implement R (key value) ⟶ address is to do some calculation on primary key to get address / Relative address of records.

It means we are applying some calculation or some relationship on primary key to get address.

key	Add.	
1234	10	
1204	7	collision
4321	10	

The process of applying calculations on key is called hashing and the calculation used is called hash function.

In this technique it is quite possible that more than one key values after applying the hash function will get the same address, this described. as follows :

$$R(K_1) = R(K_2)$$
$$\text{while } K_1 \neq K_2$$

means two diff. keys K_1 and K_2 have generated the same address. This event is called collision.

Since a relatively larger set of key values is mapped to a relatively smaller set of addressed space. Collisions are bound to happen for which we have collision resolving techniques. This hashing techniques are widely in use as it has been obsersued thatatany given point of time we are processing only certain records we need to have memory space for all records.

HASHING TECHNIQUES

1. Division Remainder
2. Mid Square
3. Hashing by folding

DIVISION REMAINDER HASHING

In this hashing technique the key value is divided by a suitable divisor. The remainder of the division becomes the address / relative address of the record.

Choice of a suitable divisor is very critical. A suitable divisor is a no. which generates less no. of collision. It has been observed that prime no. are better divisors

Key value	Divisor	Address / Rel Add.
1234	13	12] collision
1325		12]
1230		08

MID SQUARE HASHING

In this hashing technique key value is squared and then certain no. of digits from centre/ middle are taken as address.

Key value	square	Add.
120	14400	440
130	16900	690
122	14884	488

HASHING BY FOLDING

In this technique the key value is partitioned and then folded and after folding overlapped no. are added. The result becomes (after truncating the most significant bit, if required) address of the record.

$\boxed{1}\ \boxed{2\ 3\ 4\ 5}\ \boxed{6\ 7\ 8\ 9}$ - Key value

looking for a 4 digit address, so key value will be partitioned into blocks of 4

```
    2 3 4 5
    9 8 7 6
    1
  1 |3 2 2 1|
```

3221 - Address

COLLISION RESOLVING TECHNIQUES

As a relatively larger set of key values is competing for a relatively smaller set of add. space collisions are bound to happen.

When more than one keys $K_1, K_2, K_3 \ldots$ are being calculated the same home address. The event is called collision. Only one key may stored at the home address for all other records some other add. have to be found using some collision resolving techniques.

The collision resolving technique which we use to resolve collision while storing records. The same technique has to be used for retreiving the records.

for synonyms either space is found in the same file called open addressing or they are stored in some seperate overflolw file.
Two popular collision resolving techniques are

1. Linear Probing
2. Double Hashing

LINEAR PROBING

Uses open addressing method for resolving collisions.

Suppose we have a file where max. no. of records can be stored from Location 1 to max. and suppose 2 keys K_1 and K_2 are being calculated same address h, then the record having key K_1

will be stored at home address n. for 2nd
record having key K2, next available memory
location will be found. from location 1 to max-1
we will sequentially search for a empty memory
location. Wherever it is found first the record
have key value K_2 will be stored. Suppose we are
not able to found an available memory location from
location 1 to max-1, then we will be searching
from location 1 to n-1, still if no empty location
is found for storing this record, the we'll display
the file is full.

Let us assume ~~Since~~ address for a record having Key K_2, which is not being
stored ~~is being found in the same file afterwards~~
~~it may happen that some other reco~~
at home address n is being found n+5. n+5
where this record having key K_2 is being stored is
not its home add. So afterwards it may happen
~~for~~ some other add ~~is ca~~ K_3, home add is
calculated n+5 where record having key value K_2 is
already stored. So this may invite more no. of
collision.

DOUBLE HASHING

If a record is not stored at its home address then a
2nd hash function is applied to the combination of
key value and result of 1st hash function. The
target address space may be the same file or a
seperate overflow file.

Date: / / /

Approaches to improve functioning of Hash f^n.

① Synonym chaining
If Keys K_1, K_2, K_3 . . . generate the same add. then ~~sa~~ record having Key K_1 will be stored at home address for record K_2, K_3 and so on some other add. will be found using collision resolving techniques. These keys are called synonym. We can maintain a chain (linked list) of addresses of all synonyms, means Key K_1 is stored at home address ~~and~~. Similarly K_2 will have add. of K_3 and so on.
While storing the records we won't gain anything using synonym chaining but while retrieving the records the entire system becomes much faster suppose we are searching of record Key K_3. we apply hashing and got a home add. n : ~~At~~ that home add. record K_1 is stored but K_1 will give lind add. to K_2.

② Bucket Addressing
Instead of having a single add. for a single record, we may assign a bucket; so if bucket size is 5 then in a common add. 5 records may be stored. This will reduce no. of collision while storing the records only. Reduces collision upto a large extent, but it may also lead to wastage of memory.

INDEXED SEQUENTIAL FILE ORGANISATION

When there is a need to access a record sequentially as well as directly indexed seq. file organisation is used using the same key.

e.g In a ~~banking~~ university system all results are processed using enrollment no. so this can be done sequentially. Now if a guardian comes to university and asks about his ward's performed again the data base is accessed through ~~dat~~. enrollment no. but now it is done directly.

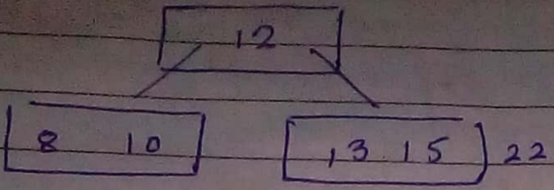
An indexed seq. organisation can be implemented using a B+ Tree.

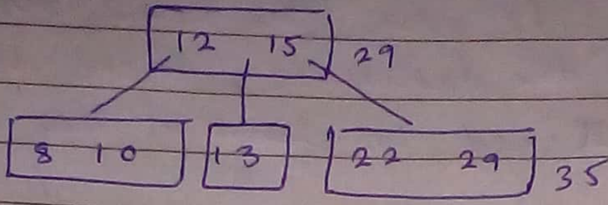
A B+ Tree is a variation on a B-Tree. In a B+ Tree a sequence set consisting of a linked list of all nodes is mai~~ntained at~~ in leaf nodes. So a B+ Tree has two parts index part and sequence set. When we want to ~~access~~ the record directly we go through index part and when we want to access all records we go through sequence set.

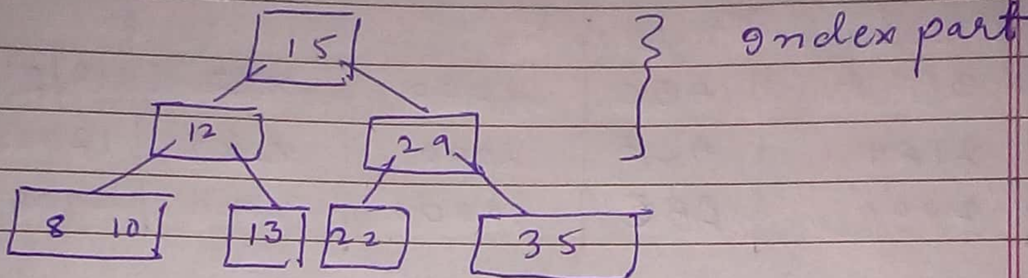
Q. Draw a B+ Tree of order 3 with ~~follong~~ ~~of~~ following data.

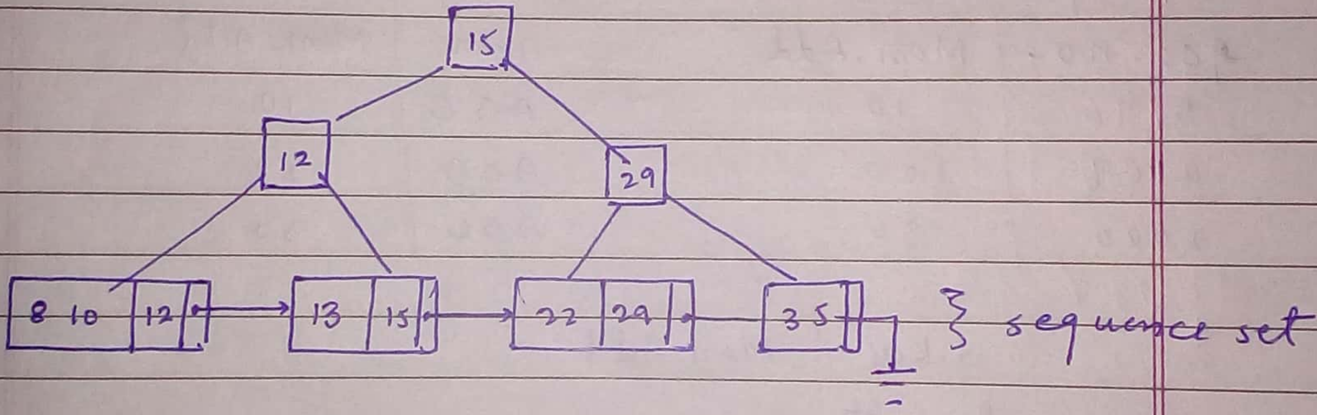
12 10 13 15 8 22 29 35

$\boxed{10 \quad 12}$ 13









MULTI KEY FILE ORGANISATION

When there is a need to access the records through different key a multikey file org. is used.

e.g In a banking system user accesses account using account no.; now if a bank manager is interested to know who all account holders have more than 10 Lac rupees balance so he'll be accessing the record through balance. Similary some

Investigating agency may be interested in knowing the what is the list ~~l~~of account holders belonging to Clement Town

In a multi key file org. the keys through which we want to access the database, there seperate indexes will be maintained in form of key add. pair.

Mem. add.	Acc. No.	name	Amount	Add.	DOB
10	3196	ABC	1000	X.Y.Z	12/2/17
20	2169	ACB	2000	x.z.y	120868
30	2000	ABC	10000	r.m.n	200678

P. Key

Acc. no.	Mem. Add.
3196	10
2169	20
2000	30

S. Key

name	Mem. Add
ABC	10
ACB	20
ABC	30

S.key Amount	Mem. Add
1000	10
2000	20
10000	30

One index is in form of primary key-memory add pair all other ~~of~~ indexes are in form of sec. key-memory add. pair. Now if data is moved from one location to other then we will have to update all above list and all memory address will change.

This can be avoided by having indexes for secondary key and primary key inplace of primary key and memory add.. Now there will be only one index which will be having add. ~~that~~ Now whenever data. is moved only one list and primary key have to be updated

The keys for which indexes have been maintained may be removed from actual database to save memory. This process is called inversion, means a completely inverted list will not have any record in actual database as all entries will be in form of indexes. This is a theoritical concept as we are not maintaining every field.

Scanned by CamScanner