UNIT 5 – FILES

The File Data Structure

A file is a collection of records. The term file, however, is usually reserved for large collections of information stored on devices outside the computer's internal memory.

It usually implies that the records are stored in secondary storage in the computer's external memory, on tapes or disks.

As a result, the ways in which the file must be organized so that operations on it can be carried out efficiently are dependent on the characteristics of the secondary storage devices used to implement the file.

File Attributes

A file's attributes vary from one operating system to another but typically consist of these:

Name: Name is the symbolic file name and is the only information kept in human readable form.

Identifier: This **unique tag** is a number that identifies the file within the file system; it is in non-human-readable form of the file.

Type: This information is needed for systems which support different types of files or its format.

Location: This information is a pointer to a device which points to the location of the file on the device where it is stored.

Size: The current size of the file (which is in bytes, words, etc.) which possibly the maximum allowed size gets included in this attribute.

Protection: Access-control information establishes who can do the reading, writing, executing, etc.

Date, Time & user identification: This information might be kept for the creation of the file, its last modification and last used. These data might be useful for in the field of protection, security, and monitoring its usage.

File Operations

A file is an abstract data type. For defining a file properly, we need to consider the operations that can be performed on files. The operating system can provide system calls to create, write, read, reposition, delete, and truncate files. There are six basic file operations within an Operating system. These are:

Creating a file: There are two steps necessary for creating a file. First, space in the file system must be found for the file. We discuss how to allocate space for the file. Second, an entry for the new file must be made in the directory.

Writing a file: To write to a file, you make a system call specify about both the name of the file along with the information to be written to the file.

Reading a file: To read from a file, you use a system call which specifies the name of the file and where within memory the next block of the file should be placed.

Repositioning inside a file: The directory is then searched for the suitable entry, and the 'current-file-position' pointer is relocating to a given value. Relocating within a file need not require any actual I/O. This file operation is also termed as 'file seeks.'

Deleting a file: For deleting a file, you have to search the directory for the specific file. Deleting that file or directory release all file space so that other files can re-use that space.

Rename or Change the Name of File.

Copy the File from one Location to another.

Sorting or Arrange the Contents of File.

Move or Cut the File from One Place to Another.

Execute Means to Run Means File Display Output.

File Organizations

File organization ensures that records are available for processing. It is used to determine an efficient file organization for each base relation.

For example, if we want to retrieve employee records in alphabetical order of name. Sorting the file by employee name is a good file organization. However, if we want to retrieve all employees whose marks are in a certain range, a file is ordered by employee name would not be a good file organization.

Types of File Organization

There are three types of organizing the file:

1. Sequential access files organization

2. Direct access files organization

3. Indexed sequential access files organization

1. Sequential file organization

In sequential access file organization, all records are stored in a sequential order. The records are arranged in the ascending or descending order of a key field.

Sequential file search starts from the beginning of the file and the records can be added at the end of the file.

In sequential file, it is not possible to add a record in the middle of the file without rewriting the Sequential File Organization

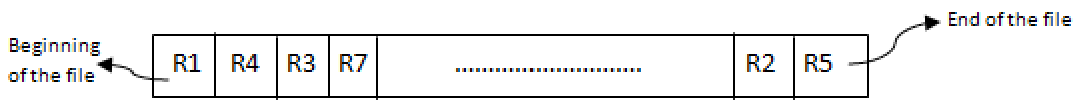
It is one of the simple methods of file organization. Here each file/records are stored one after the other in a sequential manner. This can be achieved in two ways:

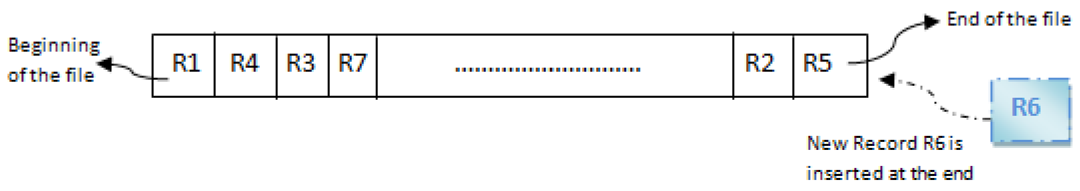
In the first method:

Records are stored one after the other as they are inserted into the tables.

When a new record is inserted, it is placed at the end of the file.

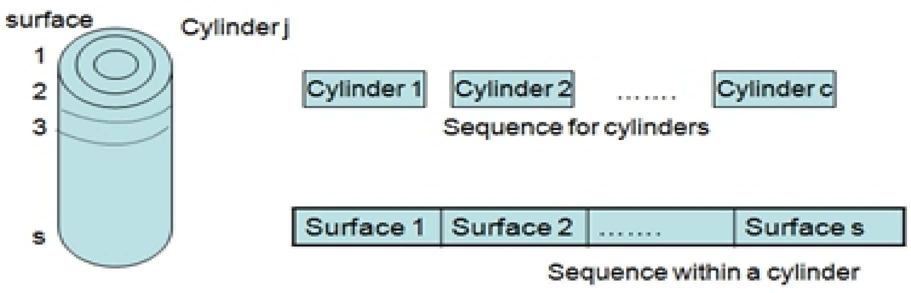
In the case of any modification or deletion of record, the record will be searched in the memory blocks. Once it is found, it will be marked for deleting and new block of record is entered.





In the second method, records are sorted (either ascending or descending) each time they are inserted into the system. This method is called **sorted file method.** Sorting of records may be based on the primary key or on any other columns. Whenever a new record is inserted, it will be inserted at the end of the file and then it will sort – ascending or descending based on key value and placed at the correct position. In the case of update, it will update the record and then sort the file to place the updated record in the right place. Same is the case with delete

Example:



Advantages of sequential file

It is simple to program and easy to design.

Sequential file is best use if storage space structure is also sequential in nature.

Disadvantages of sequential file

Sequential file is time consuming process.

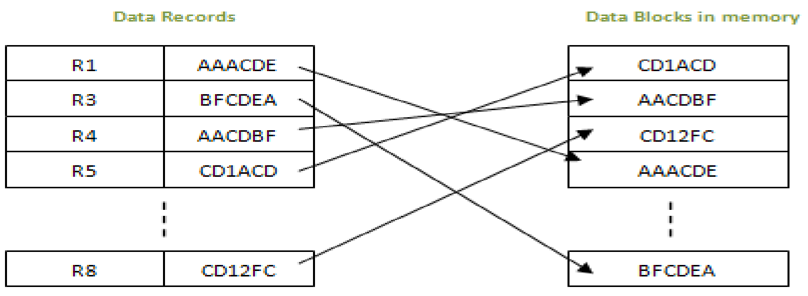
It has high data redundancy.

Random searching is not possible.

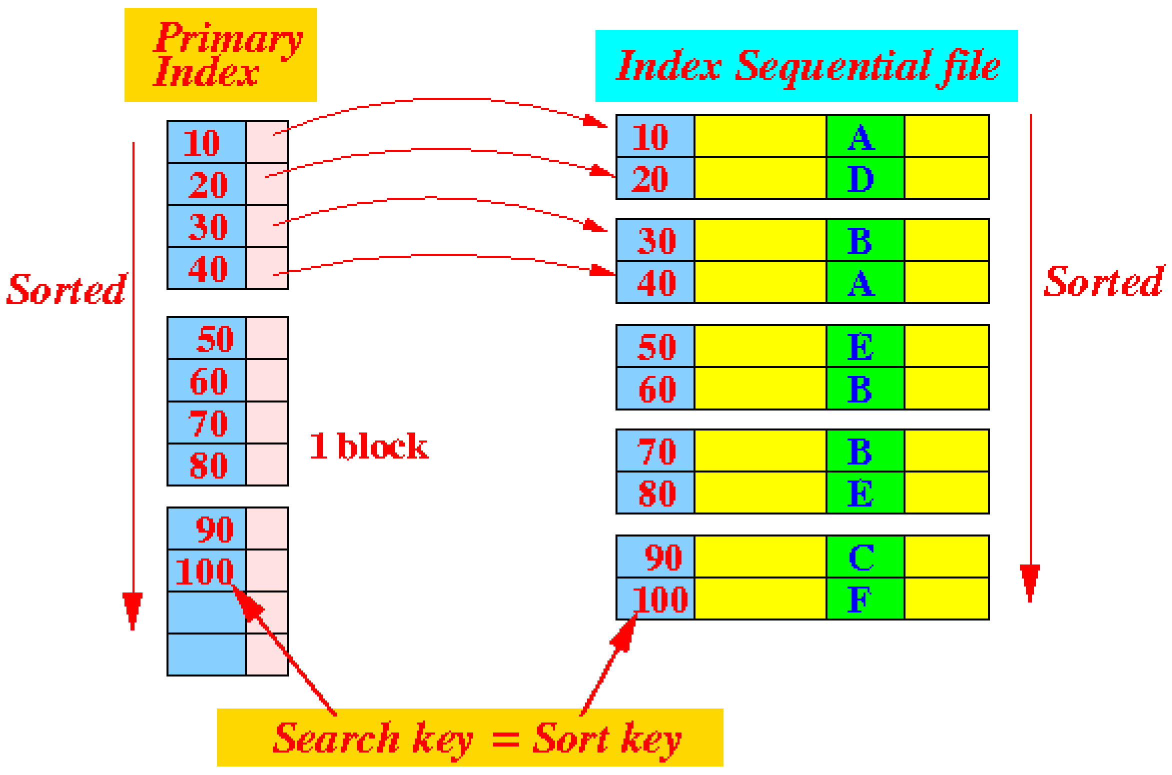
Indexed sequential access files organization

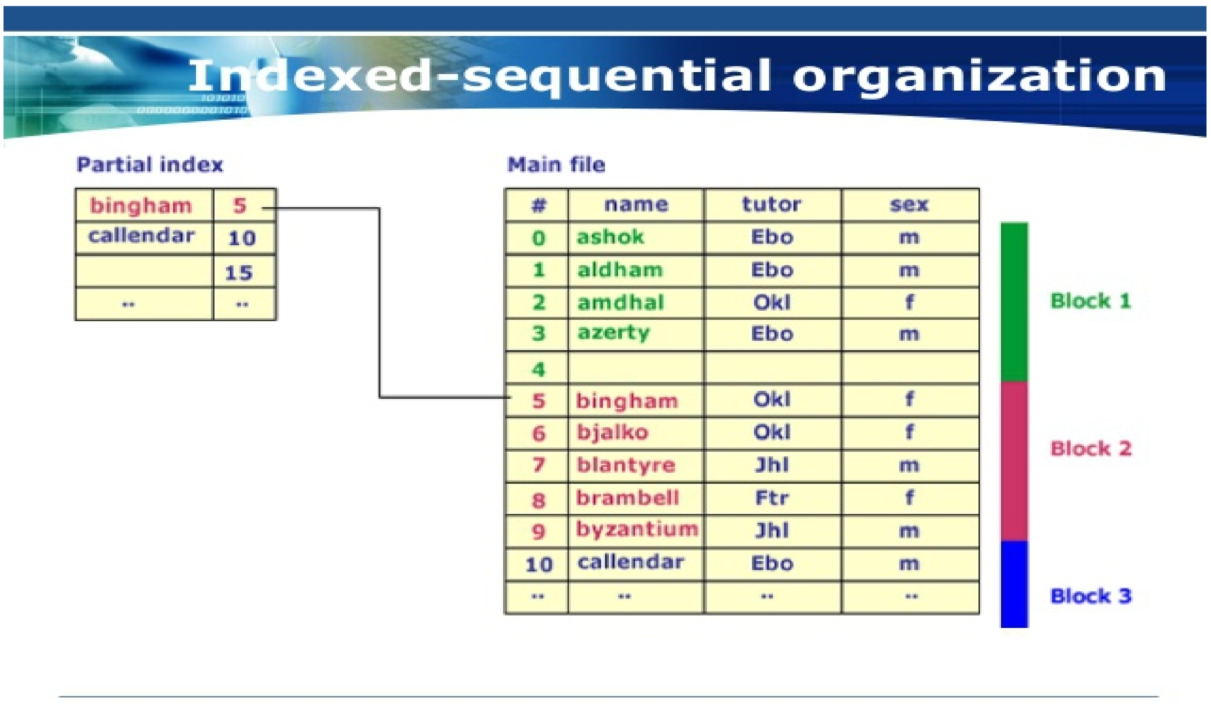
When there is need to access records sequentially by some key value and also to access records directly by the same key value, the collection of records may be organized in an effective manned called Indexes Sequential Organization.

This is an advanced sequential file organization method. Here records are stored in order of primary key in the file. Using the primary key, the records are sorted. For each primary key, an index value is generated and mapped with the record. This index is nothing but the address of record in the file.



In this method, if any record has to be retrieved, based on its index value, the data block address is fetched and the record is retrieved from memory.





Indexed-sequential organization

Partial index

bingham	5
callendar	10
	15
..	..

Main file

#	name	tutor	sex
0	ashok	Ebo	m
1	aldham	Ebo	m
2	amdhal	Okl	f
3	azerty	Ebo	m
4			
5	bingham	Okl	f
6	bjalko	Okl	f
7	blantyre	Jhl	m
8	brambell	Ftr	f
9	byzantium	Jhl	m
10	callendar	Ebo	m
..	..	..	..

Indexed sequential access file combines both sequential file and direct access file organization.

In indexed sequential access file, records are stored randomly on a direct access device such as magnetic disk by a primary key.

This file has multiple keys. These keys can be alphanumeric in which the records are ordered is called primary key.

The data can be access either sequentially or randomly using the index. The index is stored in a file and read into memory when the file is opened.

Advantages of Indexed sequential access file organization

In indexed sequential access file, sequential file and random file access is possible.

It accesses the records very fast if the index table is properly organized.

The records can be inserted in the middle of the file.

It provides quick access for sequential and direct processing.

It reduces the degree of the sequential search.

Disadvantages of Indexed sequential access file organization

Indexed sequential access file requires unique keys and periodic reorganization.

Indexed sequential access file takes longer time to search the index for the data access or retrieval.

It requires more storage space.

It is expensive because it requires special software.

It is less efficient in the use of storage space as compared to other file organizations.

3. Direct access files organization

Direct access file is also known as random access or relative file organization.

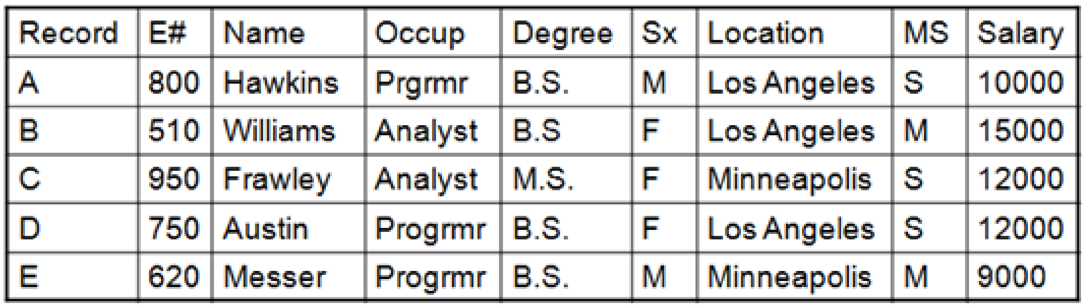
In direct access file, all records are stored in direct access storage device (DASD), such as hard disk. The records are randomly placed throughout the file.

The records do not need to be in sequence because they are updated directly and rewritten back in the same location.

This file organization is useful for immediate access to large amount of information. It is used in accessing large databases.

Example:

Sample Employee File



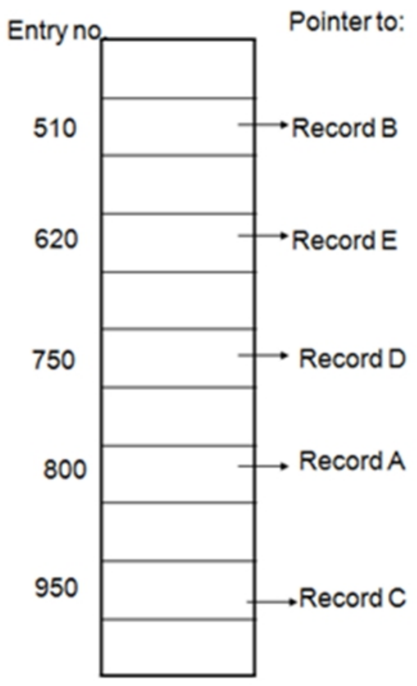
Record	E#	Name	Occup	Degree	Sx	Location	MS	Salary
A	800	Hawkins	Prgrmr	B.S.	M	Los Angeles	S	10000
B	510	Williams	Analyst	B.S	F	Los Angeles	M	15000
C	950	Frawley	Analyst	M.S.	F	Minneapolis	S	12000
D	750	Austin	Progrmr	B.S.	F	Los Angeles	S	12000
E	620	Messer	Progrmr	B.S.	M	Minneapolis	M	9000

Records are stored at random locations on the disk. This randomization could be achieved by any of several techniques:

1. Direct addressing,
2. Directory lookup,
3. Hashing.

Direct addressing: In direct addressing with equal size records, available disk space is divided out into nodes large enough to hold a record. Numeric value of primary key is used to determine the node into which a particular record is to be stored.

Directory lookup: the index is not direct access type but is a dense index (**there is an index record for every search key value in the database**)maintained using a structure suitable for index operations. Retrieving a record involves searching the index for the record address and then accessing the record itself. The storage management scheme will depend on whether fixed size or variable size nodes are being used. It requires more accesses for retrieval and update, since index searching will generally require more than one access. In both direct addressing and directory lookup, some provision must be made to handle collisions.



Advantages of direct access file organization

Direct access file helps in online transaction processing system (OLTP) like online railway reservation system.

In direct access file, sorting of the records are not required.

It accesses the desired records immediately.

It updates several files quickly.

It has better control over record allocation.

Disadvantages of direct access file organization

Direct access file does not provide backup facility.

It is expensive.

It has less storage space as compared to sequential file.

Linked organization

Linked organizations differ from sequential organizations essentially in that the logical sequence of records is generally different from the physical sequence.

In sequential i^{th} record is placed at location l_i, then the $i+1^{th}$ record is placed at $l_i + c$ where c is the length of i^{th} record or some fixed constant.

In linked organization the next logical record is obtained by following link value from present record. Linking in order of increasing primary key eases insertion deletion.

Searching for a particular record is difficult since no index is available, so only sequential search possible.

Example: We can facilitate indexes by maintaining indexes corresponding to ranges of employee numbers

Example: E# Index

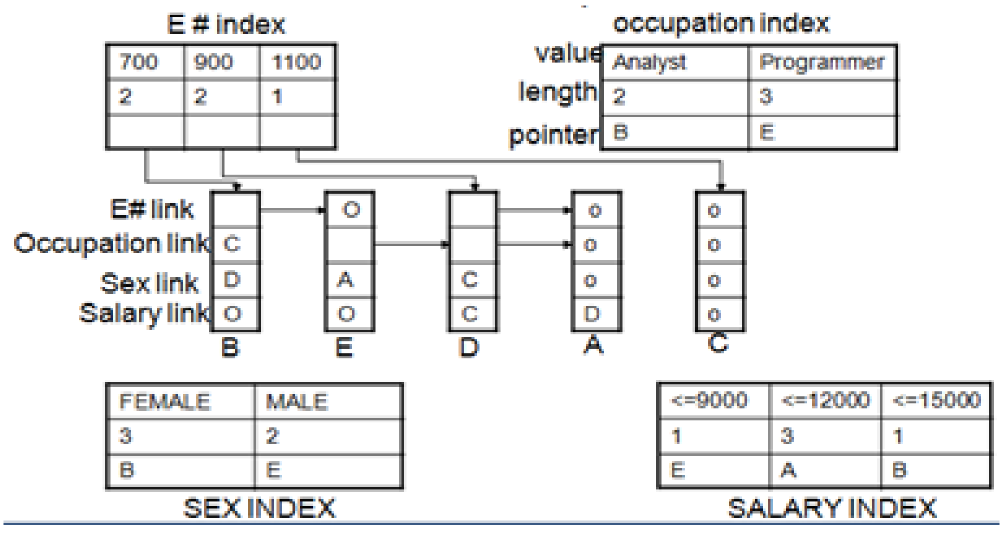
E# Range

501-700 -> First Record->Second Record

701-900 -> First Record->Second Record->Third Record

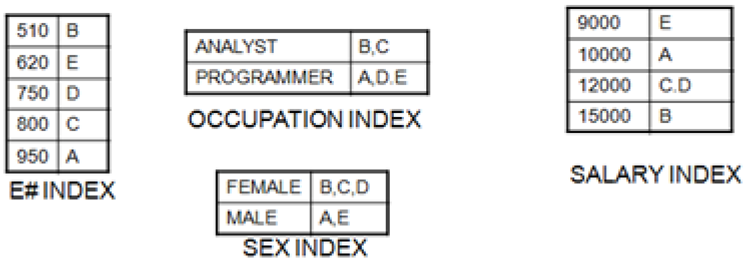
All records with same range will be linked together.

We can generalize this idea for secondary key level also. We just set up indexes for each key and allow records to be in more than one list. This leads to the **multi list** structure for file representation.



Inverted files

Inverted files are similar to multi lists. Multi lists records with the same key value are linked together with link information being kept in individual record. In case of inverted files the link information is kept in index itself.



510	B
620	E
750	D
800	C
950	A

E# INDEX

ANALYST	B,C
PROGRAMMER	A,D,E

OCCUPATION INDEX

FEMALE	B,C,D
MALE	A,E

SEX INDEX

9000	E
10000	A
12000	C,D
15000	B

SALARY INDEX

The retrieval works in two steps. In the first step, the indexes are processed to obtain a list of records satisfying the query and in the second, these records are retrieved using the list. The no. of disk accesses needed is equal to the no. of records being retrieved + the no. to process the indexes.

Inverted files represent one extreme of file organization in which only the index structures are important. The records themselves can be stored in any way.

Inverted files may also result in space saving compared with other file structures when record retrieval doesn't require retrieval of key fields. In this case key fields may be deleted from the records unlike multi list structures.

Cellular partitions

To reduce the file search times, the storage media may be divided into cells. A cell may be an entire disk pack or it may simply be a cylinder. Lists are localized to lie within a cell.

Thus if we had a multi list organization in which the list for key1=prog list included records on several different cylinders then we could break the list into several smaller lists where each prog list included only those records in the same cylinder. The index entry for prog will now contain several entries of the type (addr, length) where addr is a pointer to start of a list of records with key1=prog and length is the no. of records on the list. By doing this all records of the same cell may be accessed without moving the read/write heads.

Hashing

Hashing is another approach in which time required to search an element doesn't depend on the total number of elements. Using hashing data structure, a given element is searched with constant time complexity. Hashing is an effective way to reduce the number of comparisons to search an element in a data structure.

Hashing is the process of indexing and retrieving element (data) in a data structure to provide a faster way of finding the element using a hash key.

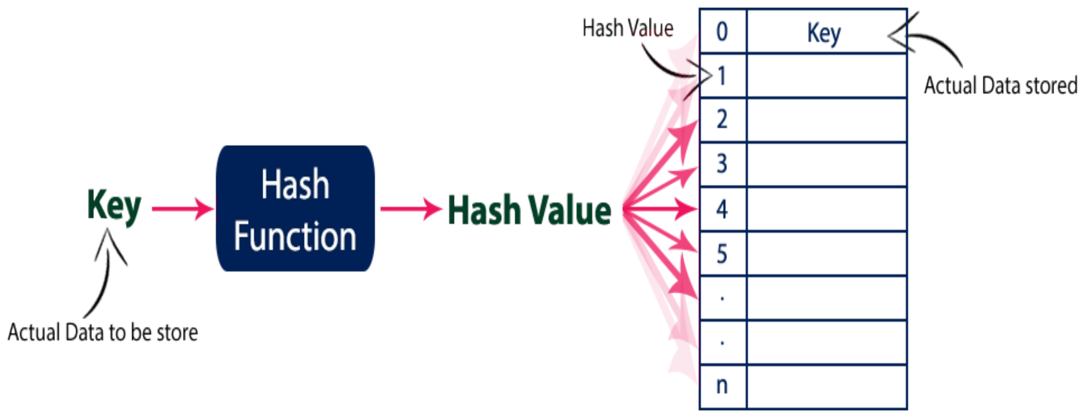
Here, the hash key is a value which provides the index value where the actual data is likely to be stored in the data structure.

In this data structure, we use a concept called Hash table to store data. All the data values are inserted into the hash table based on the hash key value. The hash key value is used to map the data with an index in the hash table. And the hash key is generated for every data using a hash function. **That means every entry in the hash table is based on the hash key value generated using the hash function.**

Hash table is just an array which maps a key (data) into the data structure with the help of hash function such that insertion, deletion and search operations are performed with constant time complexity

Hash tables are used to perform insertion, deletion and search operations very quickly in a data structure. Using hash table concept, insertion, deletion, and search operations are accomplished in constant time complexity. Generally, every hash table makes use of a function called hash function to map the data into the hash table.

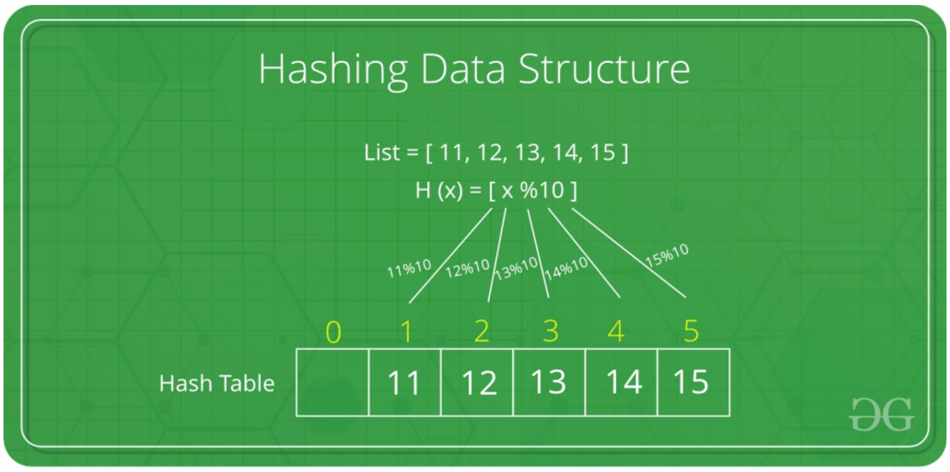
Hash function is a function which takes a piece of data (i.e. key) as input and produces an integer (i.e. hash value) as output which maps the data to a particular index in the hash table.



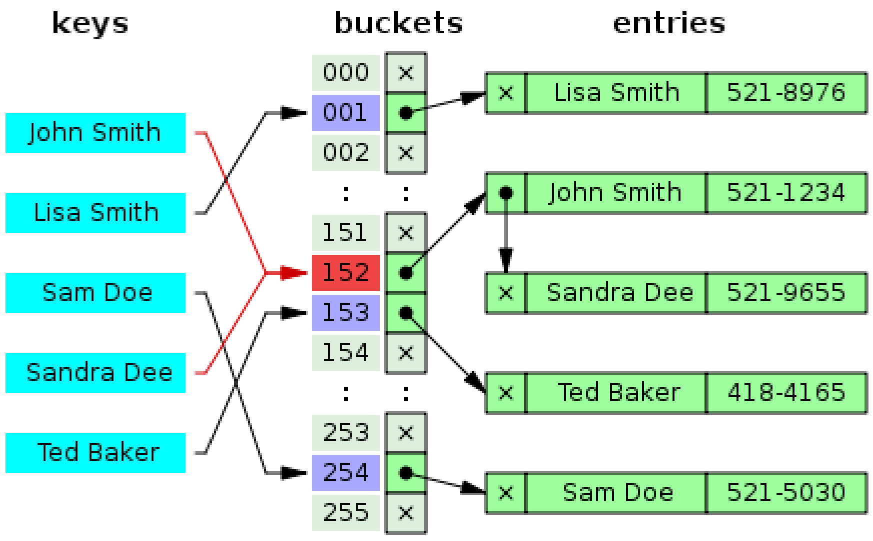
Example:

Let a hash function H(x) maps the value x at the index x%10 in an Array.

If the list of values is [11,12,13,14,15] it will be stored at positions {1,2,3,4,5} in the array or Hash table respectively.



Hashing Data Structure
List = [11, 12, 13, 14, 15]
H (x) = [x %10]
11%10
12%10
13%10
14%10
15%10
0
1
2
3
4
5
Hash Table
11
12
13
14
15



keys
buckets
entries
John Smith
Lisa Smith
Sam Doe
Sandra Dee
Ted Baker
000
001
002
151
152
153
154
253
254
255
Lisa Smith
521-8976
John Smith
521-1234
Sandra Dee
521-9655
Ted Baker
418-4165
Sam Doe
521-5030

Hashing Functions

There are various types of hash function which are used to place the data in a hash table,

1. Division method

In this the hash function is dependent upon the remainder of a division. For example:-if the record 52,68,99,84 is to be placed in a hash table and let us take the table size is 10.

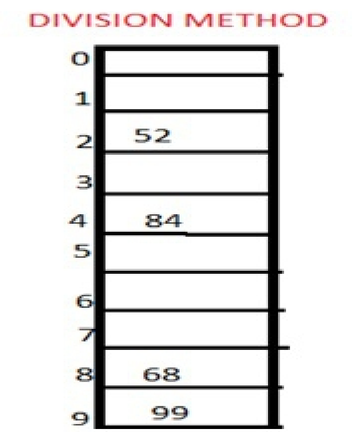
Then: h(key)=(record key value)% table size.

2=52%10

8=68%10

9=99%10

4=84%10



2. Mid square method

In this method firstly key is squared and then mid part of the result is taken as the index. For example: consider that if we want to place a record of 3101 and the size of table is 1000. So 3101*3101=9616201 i.e. h (3101) = 162 (middle 3 digits)

3. Digit folding method

In this method the key is divided into separate parts and by using some simple operations these parts are combined to produce a hash key. For example: consider a record of

12465512 then it will be divided into parts i.e. 124, 655, 12. After dividing the parts combine these parts by adding it.

H(key)=124+655+12

=791

Characteristics of good hashing function

1. The hash function should generate different hash values for the similar string.
2. The hash function is easy to understand and simple to compute.
3. The hash function should produce the keys which will get distributed, uniformly over an array.
4. A number of collisions should be less while placing the data in the hash table.
5. The hash function is a perfect hash function when it uses all the input data.

Collision

It is a situation in which the hash function returns the same hash key for more than one record, it is called as collision. Sometimes when we are going to resolve the collision it may lead to an overflow condition and this overflow and collision condition makes the poor hash function.

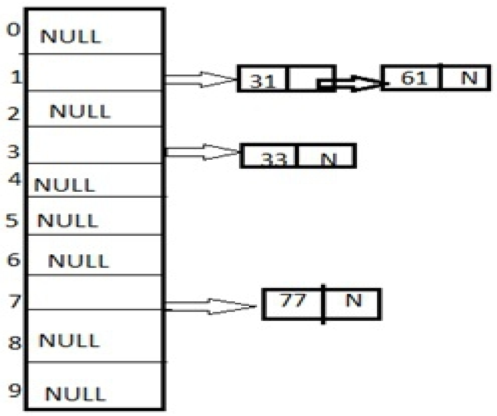
Collision resolution technique

If there is a problem of collision occurs then it can be handled by apply some technique. These techniques are called as collision resolution techniques. There are generally four techniques which are described below.

1) Chaining

It is a method in which additional field with data i.e. chain is introduced. A chain is maintained at the home bucket. In this when a collision occurs then a linked list is maintained for colliding data.

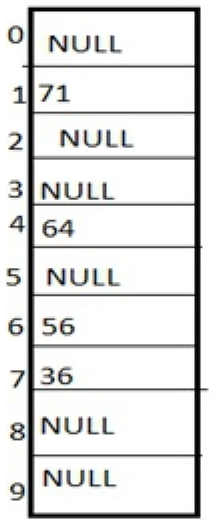
Example: Let us consider a hash table of size 10 and we apply a hash function of H(key)=key % size of table. Let us take the keys to be inserted are 31,33,77,61. In the diagram below we can see at same bucket 1 there are two records which are maintained by linked list or we can say by chaining method.



2) Linear probing

It is very easy and simple method to resolve or to handle the collision. In this collision can be solved by placing the second record linearly down, whenever the empty place is found. In this method there is a problem of clustering which means at some place block of a data is formed in a hash table.

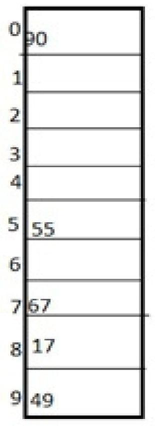
Example: Let us consider a hash table of size 10 and hash function is defined as H(key)=key % table size. Consider that following keys are to be inserted that are 56,64,36,71.



In this diagram we can see that 56 and 36 need to be placed at same bucket but by linear probing technique the records linearly placed downward if place is empty i.e. it can be seen 36 is placed at index 7.

3) Quadratic probing

This is a method in which solving of clustering problem is done. In this method the hash function is defined by the H(key)=(H(key)+x*x)%table size. Let us consider we have to insert following elements that are:-67, 90,55,17,49.



In this we can see if we insert 67, 90, and 55 it can be inserted easily but at case of 17 hash function is used in such a manner that :-(17+0*0)%10=17 (when x=0 it provide the index value 7 only) by making the increment in value of x. let x =1 so (17+1*1)%10=8.in this case bucket 8 is empty hence we will place 17 at index 8.

4) Double hashing

It is techniques in which two hash function are used when there is an occurrence of collision. In this method first hash function is simple as same as division method. But for the second hash function there are two important rules which are

1. It must never evaluate to zero.
2. Must sure about the buckets, that they are probed.

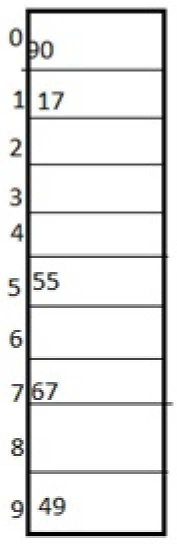
The hash functions for this technique are:

H1(key)=key % table size

H2(key)=P-(key mod P)

Where, P is a prime number which should be taken smaller than the size of a hash table.

Example: Let us consider we have to insert 67, 90,55,17,49.



In this we can see 67, 90 and 55 can be inserted in a hash table by using first hash function but in case of 17 again the bucket is full and in this case we have to use the second hash function which is H2(key)=P-(key mode P) here p is a prime number which should be taken smaller than the hash table so value of p will be the 7.

i.e. H2(17)=7-(17%7)=7-3=4 that means we have to take 4 jumps for placing the 17. Therefore 17 will be placed at index 1.