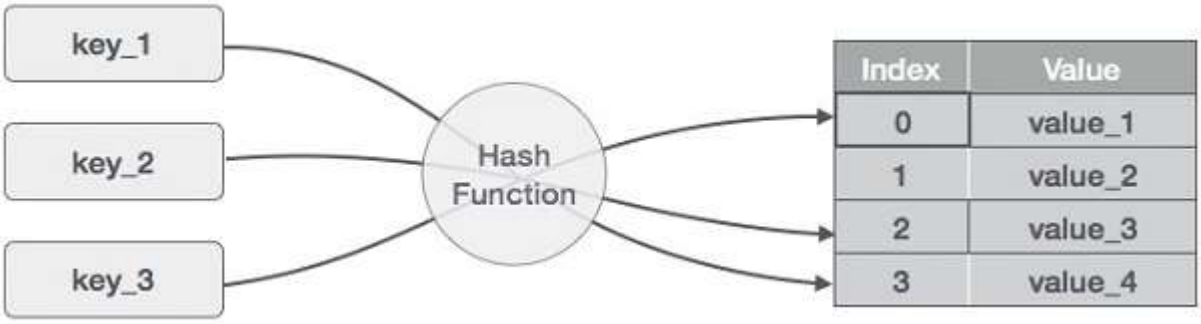
HASHGING in DATA STRUCTURE

Hash Table is a data structure which stores data in an associative manner. In a hash table, data is stored in an array format, where each data value has its own unique index value. Access of data becomes very fast if we know the index of the desired data.

Thus, it becomes a data structure in which insertion and search operations are very fast irrespective of the size of the data. Hash Table uses an array as a storage medium and uses hash technique to generate an index where an element is to be inserted or is to be located from.

Hashing

Hashing is a technique to convert a range of key values into a range of indexes of an array. We're going to use modulo operator to get a range of key values. Consider an example of hash table of size 20, and the following items are to be stored. Item are in the (key,value) format.



- (1,20)
- (2,70)
- (42,80)
- (4,25)
- (12,44)
- (14,32)
- (17,11)
- (13,78)
- (37,98)

Sr.No.	Key	Hash	Array Index
1	1	1 % 20 = 1	1
2	2	2 % 20 = 2	2

3	42	42 % 20 = 2	2
4	4	4 % 20 = 4	4
5	12	12 % 20 = 12	12
6	14	14 % 20 = 14	14
7	17	17 % 20 = 17	17
8	13	13 % 20 = 13	13
9	37	37 % 20 = 17	17

Linear Probing

As we can see, it may happen that the hashing technique is used to create an already used index of the array. In such a case, we can search the next empty location in the array by looking into the next cell until we find an empty cell. This technique is called linear probing.

Sr.No.	Key	Hash	Array Index	After Linear Probing, Array Index
1	1	1 % 20 = 1	1	1
2	2	2 % 20 = 2	2	2
3	42	42 % 20 = 2	2	3
4	4	4 % 20 = 4	4	4
5	12	12 % 20 = 12	12	12
6	14	14 % 20 = 14	14	14

7	17	17 % 20 = 17	17	17
8	13	13 % 20 = 13	13	13
9	37	37 % 20 = 17	17	18

Hashed Indices

In the last chapter, we discussed hashing in detail. The same concept of hashing can be used tocreate hashed indices. So far, we have studied that hashing is used to compute the address of a record by using a hash function on the search key value. If at any point of time, the hashed values map to the same address, then collision occurs and schemes to resolve these collisions are applied to generate a new address.

Choosing a good hash function is critical to the success of this technique. By a good hash function, we mean two things. First, a good hash function, irrespective of the number of search keys, gives an average-case lookup that is a small constant. Second, the function distributes records uniformly and randomly among the buckets, where a bucket is defined as a unit of one or morerecords (typically a disk block). Correspondingly, the worst hash function is one that maps all the keys to the same bucket.

However, the drawback of using hashed indices includes:

1. Though the number of buckets is fixed, the number of files may grow with time.
2. If the number of buckets is too large, storage space is wasted.
3. If the number of buckets is too small, there may be too many collisions.

It is recommended to set the number of buckets to twice the number of the search key values in the file.

This gives a good space–performance trade off. A hashed file organization uses hashed indices. Hashing is used to calculate the address of disk block where the desired record is stored. If K is the

set of all search key values and B is the set of all bucket addresses, then a hash function H maps K to B.

We can perform the following operations
in a hashed file organization.

Insertion

To insert a record that has ki as its search value, use the hash function h(ki) to compute the address of the bucket for that record. If the bucket is free, store the record else use chaining to store the record.

Search

To search a record having the key value
ki, use h(ki)
to compute the address of the bucket wherethe record is stored. The bucket may contain one or several records, so check for every record in the bucket (by comparing ki with the key of every record) to finally retrieve the desired record with the given key value.

Deletion

To delete a record with key value ki, use h(ki) to compute the address of the bucket where the record is stored. The bucket may contain one or several records so check for every record in the bucket
(by comparing Ki with the key of every record). Then delete the record as we delete a node from a linear linked list.

Note that in a hashed File organization, the secondary indices need to be organized using hashing.