**File Organization**  
  
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**  
  
1. Sequential access file organization  
2. Direct access file organization  
3. Indexed sequential access file organization

**1. Sequential access file organization**

* Storing and sorting in contiguous block within files on tape or disk is called as **sequential access 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 file.

**Advantages of sequential file**

* It is simple to program and easy to design.
* Sequential file is best use if storage space.

**Disadvantages of sequential file**

* Sequential file is time consuming process.
* It has high data redundancy.
* Random searching is not possible.

**2. Direct access file 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 does 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.
* It is also called as hashing.

**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 back up facility.
* It is expensive.
* It has less storage space as compared to sequential file.

**3. Indexed sequential access file organization**

* 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 have 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.

**Hashing**

Hashing is an important data structure designed to solve the problem of efficiently finding and storing data in an array. For example, if you have a list of 20000 numbers, and you have given a number to search in that list- you will scan each number in the list until you find a match.

The hash function in the data structure verifies the file which has been imported from another source. A hash key for an item can be used to accelerate the process. It increases the efficiency of retrieval and optimises the search. This is how we can simply give hashing definition in data structure.

It requires a significant amount of your time to search in the entire list and locate that specific number. This manual process of scanning is not only time-consuming but inefficient too. With hashing in the data structure, you can narrow down the search and find the number within seconds.

## What is Hashing in Data Structure?

**Hashing in the** **data structure** is a technique of mapping a large chunk of data into small tables using a hashing function. It is also known as the message digest function. It is a technique that uniquely identifies a specific item from a collection of similar items.

It uses hash tables to store the data in an array format. Each value in the array has been assigned a unique index number. Hash tables use a technique to generate these unique index numbers for each value stored in an array format. This technique is called the hash technique.

You only need to find the index of the desired item, rather than finding the data. With indexing, you can quickly scan the entire list and retrieve the item you wish. Indexing also helps in inserting operations when you need to insert data at a specific location. No matter how big or small the table is, you can update and retrieve data within seconds.

The hash table is basically the array of elements and the hash techniques of search are performed on a part of the item i.e. key. Each key has been mapped to a number, the range remains from 0 to table size 1

Types of hashing in data structure is a two-step process.

1. The hash function converts the item into a small integer or hash value. This integer is used as an index to store the original data.
2. It stores the data in a hash table. You can use a hash key to locate data quickly.

### Examples of Hashing in Data Structure

The following are real-life examples of **hashing in the data structure**–

* In schools, the teacher assigns a unique roll number to each student. Later, the teacher uses that roll number to retrieve information about that student.
* A library has an infinite number of books. The librarian assigns a unique number to each book. This unique number helps in identifying the position of the books on the bookshelf.

## Hash Function

The hash function in a data structure maps the arbitrary size of data to fixed-sized data. It returns the following values:  a small integer value (also known as hash value), hash codes, and hash sums. The hashing techniques in the data structure are very interesting, such as:

**hash = hashfunc(key)**

**index = hash % array\_size**

The hash function must satisfy the following requirements:

* A good hash function is easy to compute.
* A good hash function never gets stuck in clustering and distributes keys evenly across the hash table.
* A good hash function avoids collision when two elements or items get assigned to the same hash value.

One of the hashing techniques of using a hash function is used for data integrity. If using a hash function one change in a message will create a different hash.

The three characteristics of the hash function in the data structure are:

1. Collision free
2. Property to be hidden
3. Puzzle friendly

## Hash Table

**Hashing in data structure** uses hash tables to store the key-value pairs. The hash table then uses the hash function to generate an index. Hashing uses this unique index to perform insert, update, and search operations.

It can be defined as a bucket where the data are stored in an array format. These data have their own index value. If the index values are known then the process of accessing the data is quicker.

SAAS Business – What is So Different?

## How does Hashing in Data Structure Works?

In hashing, the hashing function maps strings or numbers to a small integer value. Hash tables retrieve the item from the list using a hashing function. The objective of hashing technique is to distribute the data evenly across an array. Hashing assigns all the elements a unique key. The hash table uses this key to access the data in the list.

Hash table stores the data in a key-value pair. The key acts as an input to the hashing function. Hashing function then generates a unique index number for each value stored. The index number keeps the value that corresponds to that key. The hash function returns a small integer value as an output. The output of the hashing function is called the hash value.

Let us understand **hashing in a data structure** with an example. Imagine you need to store some items (arranged in a key-value pair) inside a hash table with 30 cells.

The values are: (3,21) (1,72) (40,36) (5,30) (11,44) (15,33) (18,12) (16,80) (38,99)

The hash table will look like the following:

| **Serial Number** | **Key** | **Hash** | **Array Index** |
| --- | --- | --- | --- |
| 1 | 3 | 3%30 = 3 | 3 |
| 2 | 1 | 1%30 = 1 | 1 |
| 3 | 40 | 40%30 = 10 | 10 |
| 4 | 5 | 5%30 = 5 | 5 |
| 5 | 11 | 11%30 = 11 | 11 |
| 6 | 15 | 15%30 = 15 | 15 |
| 7 | 18 | 18%30 = 18 | 18 |
| 8 | 16 | 16%30 = 16 | 16 |
| 9 | 38 | 38%30 = 8 | 8 |

The process of taking any size of data and then converting that into smaller data value which can be named as hash value. This hash alue can be used in an index accessible in hash table. This process define hashing in data structure.

## Collision Resolution Techniques

**Hashing in data structure** falls into a collision if two keys are assigned the same index number in the hash table. The collision creates a problem because each index in a hash table is supposed to store only one value. **Hashing in data structure** uses several collision resolution techniques to manage the performance of a hash table.

It is a process of finding an alternate location. The collision resolution techniques can be named as-

1. Open Hashing (Separate Chaining)
2. Closed Hashing (Open Addressing)

* Linear Probing
* Quadratic Probing
* Double Hashing

## Linear Probing

**Hashing in data structure** results in an array index that is already occupied to store a value. In such a case, hashing performs a search operation and probes linearly for the next empty cell.

Linear probing in hash techniques is known to be the easiest way to resolve any collisions in hash tables. A sequential search can be performed to find any collision that occurred.

### Linear Probing Example

Imagine you have been asked to store some items inside a hash table of size 30. The items are already sorted in a key-value pair format. The values given are: (3,21) (1,72) (63,36) (5,30) (11,44) (15,33) (18,12) (16,80) (46,99).

The hash(n) is the index computed using a hash function and T is the table size. If slot index = ( hash(n) % T) is full, then we look for the next slot index by adding 1 ((hash(n) + 1) % T). If (hash(n) + 1) % T is also full, then we try (hash(n) + 2) % T. If (hash(n) + 2) % T is also full, then we try (hash(n) + 3) % T.

The hash table will look like the following:

| **Serial Number** | **Key** | **Hash** | **Array Index** | **Array Index after Linear Probing** |
| --- | --- | --- | --- | --- |
| 1 | 3 | 3%30 = 3 | 3 | 3 |
| 2 | 1 | 1%30 = 1 | 1 | 1 |
| 3 | 63 | 63%30 = 3 | 3 | 4 |
| 4 | 5 | 5%30 = 5 | 5 | 5 |
| 5 | 11 | 11%30 = 11 | 11 | 11 |
| 6 | 15 | 15%30 = 15 | 15 | 15 |
| 7 | 18 | 18%30 = 18 | 18 | 18 |
| 8 | 16 | 16%30 = 16 | 16 | 16 |
| 9 | 46 | 46%30 = 8 | 16 | 17 |

## Double Hashing

The double hashing technique uses two hash functions. The second hash function comes into use when the first function causes a collision. It provides an offset index to store the value.

The formula for the double hashing technique is as follows:

**(firstHash(key) + i \* secondHash(key)) % sizeOfTable**

Where i is the offset value. This offset value keeps incremented until it finds an empty slot.

For example, you have two hash functions: h1 and h2. You must perform the following steps to find an empty slot:

1. Verify if hash1(key) is empty. If yes, then store the value on this slot.
2. If hash1(key) is not empty, then find another slot using hash2(key).
3. Verify if hash1(key) + hash2(key) is empty. If yes, then store the value on this slot.
4. Keep incrementing the counter and repeat with hash1(key)+2hash2(key), hash1(key)+3hash2(key), and so on, until it finds an empty slot.

### Double Hashing Example

Imagine you need to store some items inside a hash table of size 20. The values given are: (16, 8, 63, 9, 27, 37, 48, 5, 69, 34, 1).

**h1(n)=n%20**

**h2(n)=n%13**

**n h(n, i) = (h1 (n) + ih2(n)) mod 20**

| **n** | **h(n,i) = (h’(n) + i2) %20** |
| --- | --- |
| 16 | I = 0, h(n,0) = 16 |
| 8 | I = 0, h(n,0) = 8 |
| 63 | I = 0, h(n,0) = 3 |
| 9 | I = 0, h(n,0) = 9 |
| 27 | I = 0, h(n,0) = 7 |
| 37 | I = 0, h(n,0) = 17 |
| 48 | I = 0, h(n,0) = 8  I = 0, h(n,1) = 9  I = 0, h(n,2) = 12 |
| 5 | I = 0, h(n,0) = 5 |
| 69 | I = 0, h(n,0) = 9  I = 0, h(n,1) = 10 |
| 34 | I = 0, h(n,0) = 14 |
| 1 | I = 0, h(n,0) = 1 |

## What is a hash table?

A hash table is an implementation of an associative array, a structure used in computer programming to implement an abstract data type (ADT). In an abstract data type, the programmer does not need to know about the implementation details of the data type (such as how the data is stored in memory), only the operations that can be performed on the data type. A hash table uses a hash function to compute an index into an array of buckets or slots, from which the desired value can be found. Hash Tables are used to implement map-like data structures. Hash tables are very much in use in modern computers for implementing things like dictionaries (as in python), associative arrays (as in PHP), java hashtables, etc. Hash tables are usually implemented in languages as an array of values sorted by their keys. This makes look up and insert/delete operations very fast, as the data is stored systematically in memory.

## What are the applications of hashing functions?

Hashing functions are used for several applications in computer science, for example, cryptography and document fingerprinting. The main purpose of a hashing function is to map large amounts of input to a fixed length output. In cryptography, hashing is used to ensure that a message or document has not been tampered with. If the document or message is altered in any way (even a single character), the hash value is also altered. It is therefore almost impossible to create a document or message with a given hash value.

## What are the collision resolution techniques in hashing?

Collision resolution techniques in hashing are used to resolve collisions in hashing. Collision resolution techniques are either chaining or open addressing. In chaining, we retain the old element in place and insert the new element in the next available space. It is a simple method of collision resolution but has a drawback of poor performance. In open addressing, we replace the old element with new element and mark the old element as a collision.

## What is key in hashing?

These are types of hashing in a data structure, the static hashing gives access to the user to lookup at the finalised dictionary, the finalised dictionary cannot be changed. Whereas, the dynamic dictionary allows the data to be moved based on the demand, as the data can be added or removed. The final result also changes the basis of the records.