CONTENTS

CHAPTER 30

CHAPTER – 30

HASHING

OVERVIEW

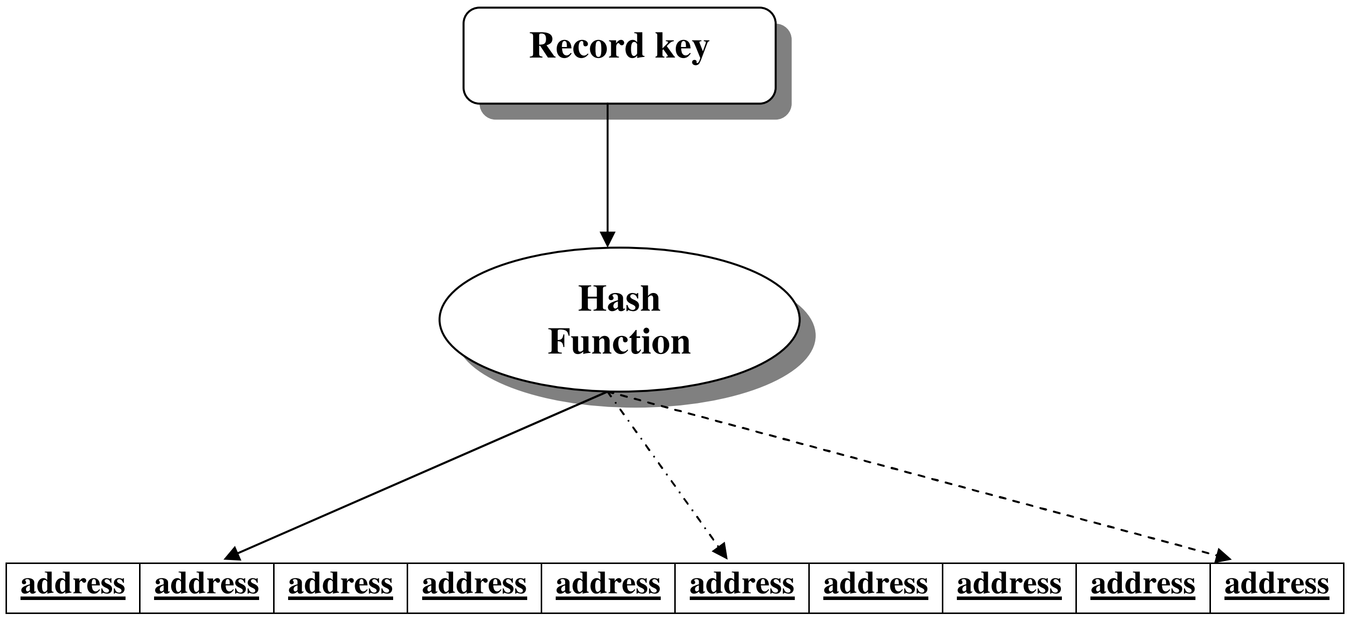
A record being sought is assumed to be stored in some table and that it is necessary to pass through some number of keys before finding the desired one. The organization of the file and the order in which the keys are inserted affect the number of keys that must be inspected before obtaining the desired one. Obviously, the efficient **search techniques** are those that minimize the number of these comparisons. Optimally, a preferred way is to have a table organization and **search technique** in which there are no unnecessary comparisons.

The **linear search techniques** discussed so far require several tests before the desired data is found. In an **ideal search**, if the location of the data is known, the search can be made directly at that location. In **sequential searches**, the search with a group of tests is done in a loop until the list ends or a successful search in finding the intended data. With **binary search** the number of searches made can be reduced compared to any similar linear list. The search continues in **binary search** in a manner to break the list into two halves, and test is done to which half further search will be done. Still the there are sizable number of comparisons will be made before the data is found or the list is exhausted in an unsuccessful search. The pursuit is to access the data with one single test.

A **hash search** is a seach in which the key, through an algorithmic function, determines the location of the data. The key is transformed by the algorithm into an index that contains the data. If each record is to be retrieved in a single access, the location of the record within the table can depend only on the key; it may not depend on the locations of other keys. The most efficient way to organize such a table is an array. If the record keys are integers, the keys themselves can serve as indices to the array.

HASH TABLE

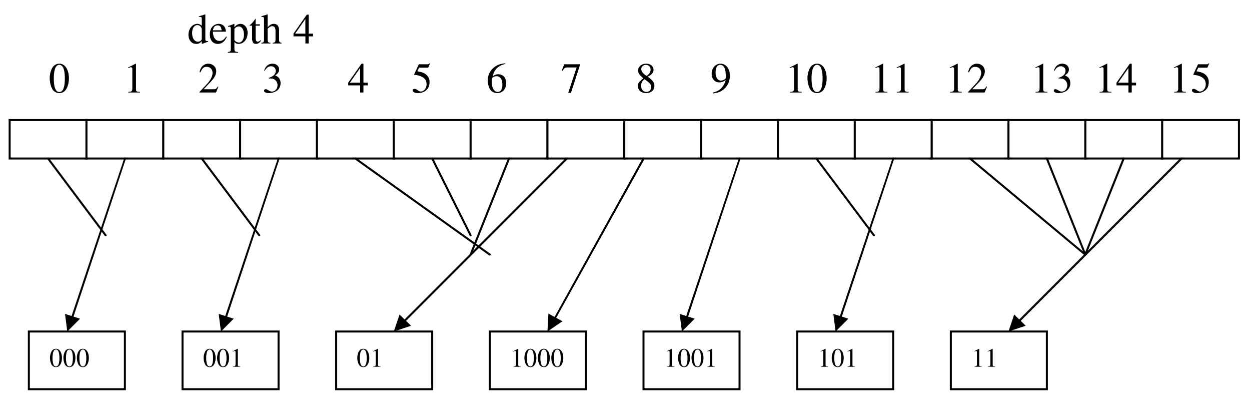
The idea of a ***hash table*** is to allow many of the different possible keys that might occur to be mapped to the same location in an array under the action of the index function. There will be a possibility that two records want to be in the same place, but if the number of records that actually occur is small relative to the size of the array, then this possibility will cause little loss of time. Even when most entries in the array are occupied, hash methods can be an effective means of information retrieval. An array of indexes is declared through which the record location can be found. The size of the array is the hash table size. The hash table will have ***0*** through table ***size – 1*** number of possible unique values those can be mapped to the corresponding record location.



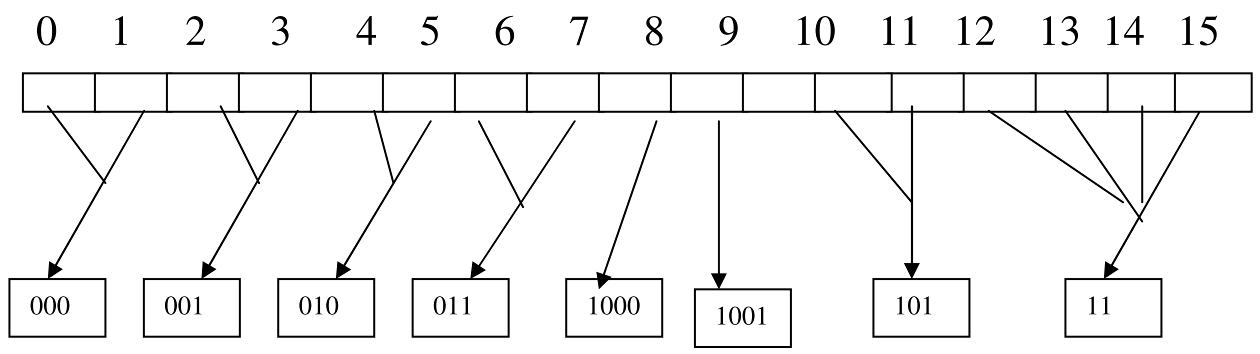
Hash Table

data	**data**	**data**	**data**	**data**	**data**	**data**	**data**	**data**	**data**

EXTENDIBLE HASHING



Extendible hashing configuration



Bucket 3 overflows

DRAWBACKS OF DYNAMIC AND EXTENDIBLE HASHING

One drawback of dynamic and extendible hashing is the need for index. Although the index may be kept in internal storage once the file is opened, this is not always possible if the index becomes very large. Also, the index does require external storage when the file is not in use. In addition, the external copy of the index may have to be constantly updated to guard against power failure or other interruption that would prevent rewriting the index when the file is closed.