UNIT 8 DATA STRUCTURES AND FILE ORGANIZATION

Structure

8.0 OBJECTIVES

The database concept is an important aspect of the storage and retrieval of information. Data structure and file organization are concerned with methods of organization of data in a database dealing with physical organization of data. After reading this Unit, you will be able to:

- understand the basic concepts related to data structures and file organization.
- comprehend physical storage structures of data and file organization techniques.
- gain an insight into the role which the data structures and file organization play in the overall performance and access efficiency in a database.

8.1 INTRODUCTION

Data structures and file organization refer to the methods of organizing the data in a database. They primarily deal with physical storage of data which assumes significance in retrieving, storing and reorganizing data in a database.

The performance of a database greatly depends on the data storage structures and file organization. Data structures define the physical design of a database.

8.2 DEFINITIONS AND BASIC CONCEPTS

Some of the basic concepts which are essential to provide the necessary background for the understanding related to data structures and file organization have been given in the following paragraphs:

8.2.1 Memory Hierarchy

Computer storage media form a memory hierarchy that includes two main categories of storage:

- **Primary storage:** Pertains to storage media used by the Central Processing Unit (CPU) i.e. the main memory and also the cache memory. The primary storage memory also called RAM (Random Access Memory) provides fast access to data and is volatile, i.e., loses its content in case of a power outage.
- **Secondary storage:** Includes magnetic disks, optical disks and tapes. Secondary storage memory provides slower access to data than RAM.

The memory hierarchy is represented in Fig. 2.1. As one moves down the hierarchy from cache memory, access speed and cost decrease.

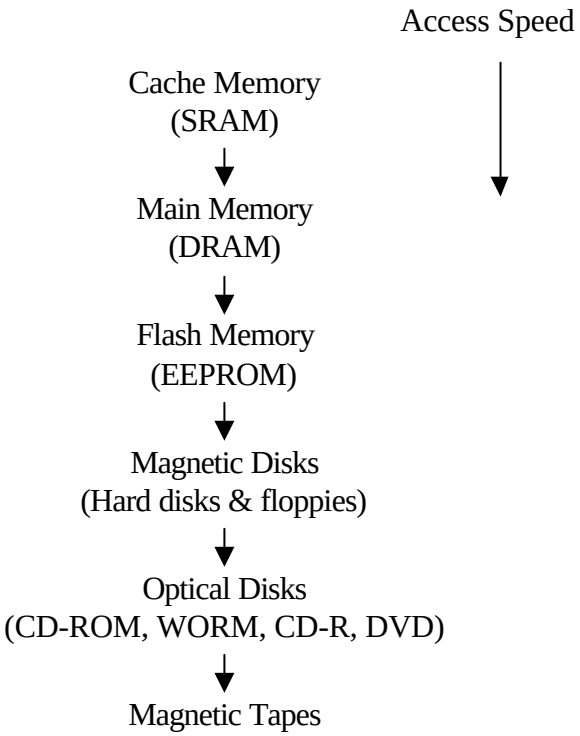

Fig. 2.1 : Memory hierarchy

Static RAM (SRAM) which is cache memory is used by CPU to speed up execution of programmes while Dynamic RAM (DRAM) provides the main work area for CPU. Flash memory which is non-volatile and called EEPROM (Electrically Erasable Programmable Read-Only-Memory) has access speed and performance between DRAM and magnetic disks.

CD-ROM (Compact Disk Read-Only-Memory) disks store data optically and are read by a laser. WORM (Write-Once-Read-Memory) disks are used for archiving

data and allow data to be written once and read any number of times. DVD (Digital Video Disk) – a type of optical disk allows storage of four to fifteen gigabytes of data per disk. Magnetic tapes are used for archiving and back-up storage and are becoming popular as tertiary storage to hold terabytes of data. Juke boxes (optical and tape) are employed to use arrays of CD-ROMs and tapes.

8.2.2 RAID Technology

A major advance in secondary storage technology is represented by RAID (Redundant Array of Inexpensive/Independent Disks) technology. The RAID idea has been developed into an elaborate set of alternative RAID architectures (RAID levels 0 through 6).

The main goal of RAID is to even out the widely different rates of performance improvement of disks against those in memory and microprocessors. While RAM capacities have quadrupled every two to three years, disk access times are improving at less than 10 per cent per year, and disk transfer rates are improving at roughly 20 per cent per year. Though disk capacities are improving at a fast rate, the speed and access time improvements are of much smaller magnitude.

The problem is overcome by using a large array of small independent disks acting as a single high-performance logical disk. A concept called data striping is used. It utilizes parallelism to improve disk performance. Data striping distributes data transparently over multiple disks to make them appear as a single large, fast disk. Striping improves overall I/O performance by allowing multiple I/Os to be serviced in parallel, thus providing high overall transfer rates. Data striping also accomplishes load balancing among disks.

It should be noted that data can be read or written only one block at a time, so a typical transfer contains 512 bytes (block size = 512 bytes). Data striping can be applied at a finer granularity by breaking up a byte of data into bits and spreading the bits to multiple disks. Using bit-level data striping with 8-bit bytes, eight physical disks may be considered as one logical disk with an eight fold increase in data transfer rate. Each disk participates in each I/O request and the total data read per request is eight times. Data striping may also be done at block level which distributes blocks of a file across disks.

In addition to improving performance, RAID is also used to improve reliability by storing redundant information on disks. One technique for introducing redundancy is called mirroring. Data is written redundantly on two identical physical disks that are treated as one logical disk. If a disk fails, the other is used until the first is repaired.

Thus RAID technology has contributed significantly to improving the performance and reliability of data storage on disks.

8.2.3 Indexes

An index is a file in which each entry (record) consists of a data value together with one or more pointers (physical storage addresses). The data value is a value for some field of the indexed file (the indexed field) and pointers identify records in the indexed file having that value for that field.

An index can be used in two ways. First, it can be used for sequential access to the indexed file i.e. access according to the values of the indexed field by imposing an ordering of the indexed file. Second, it can also be used for direct access to individual

records in the indexed file on the basis of a given value for that same field. In general, indexing speeds up retrieval but may slow down update.

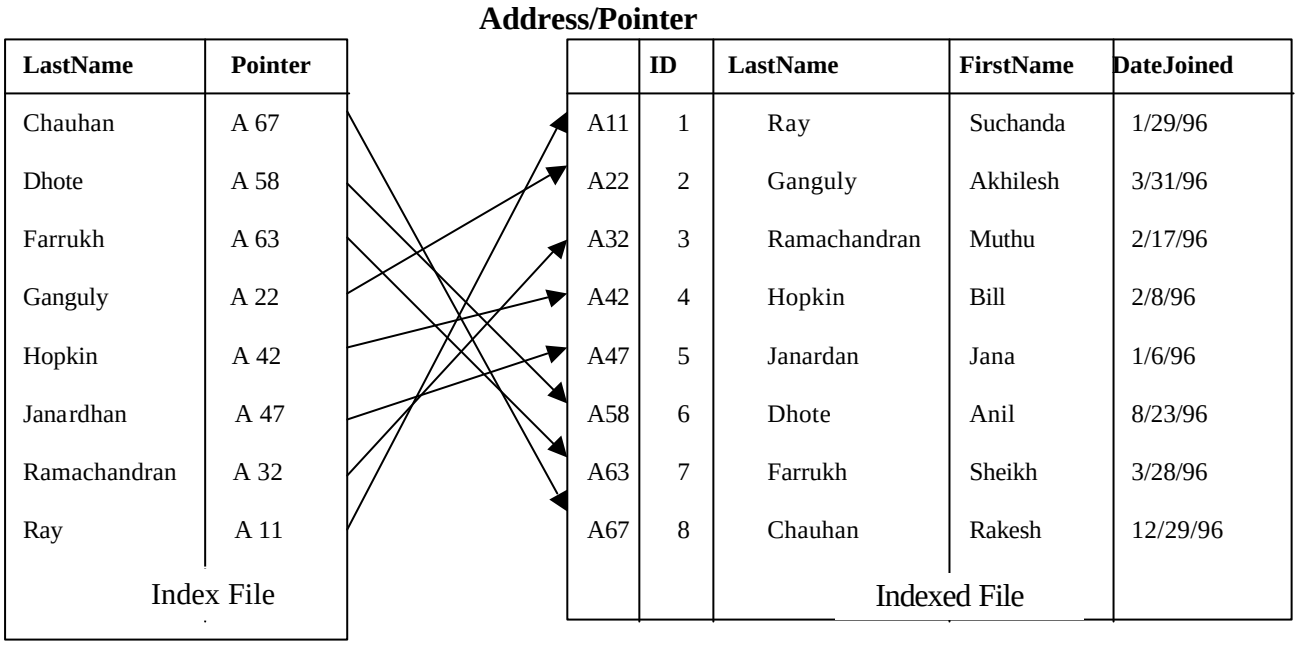

Fig. 2.2: An illustration of indexes

8.2.4 Binary Search

Binary searching is a technique used to substantially lessen the time required to search the indexes of lengthy inverted lists (see 2.3.2). In this technique, the value sought is first compared to the value in the middle of the list. This indicates whether the value sought is in the top or bottom half of the list. The value sought is then compared with the middle entry of the appropriate half. This indicates which fourth the value is in. Then the value is compared to the middle of the fourth, and so on until the desired value is found. Thus the binary search keeps splitting the data set in half until it finds the desired value.

An example of binary search is shown in Fig. 2.3. To find the entry for Janardan find the middle of the list (Gautam). Janardan is post Gautam so split the second half in half (Kamla). Keep splitting the reminder in half until Janardan is found.

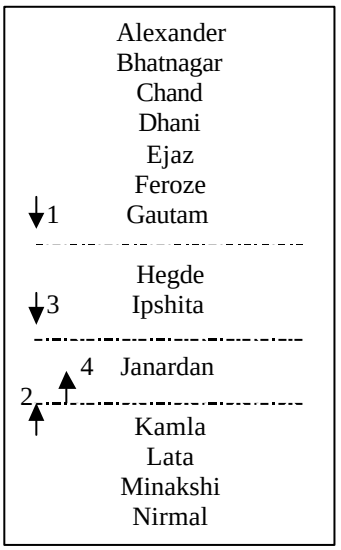

Fig. 2.3 : An example of binary search

Self Check Exercise

1) What is the role of indexes in database search ?

.....

.....

.....

.....

8.3 DATA STRUCTURES

The term data structure refers to the manner in which relationships between data elements are represented in the computer system. Organization of indexes, representation of stored fields, physical sequence of stored records etc. are included in the purview of data structures. Thus, an understanding of data structures is important in gaining an understanding of database management systems.

There are three major types of data structures : linked lists, inverted lists and B-Trees. These data structures have been explained in the following paragraphs :

8.3.1 Linked Lists

A simple linked list is a chain of pointers embedded in records. It indicates either a record sequence for an attribute other than the primary key or all the records with a common property. With a linked list, any data element can be stored separately. A pointer is then used to link it to the next data item.

Fig. 2.4 illustrates the basic concept of linked lists. In this example each row of data is stored separately. Then an index is created on the field (key) LastName. However, each element of the index is stored separately. An index element consists of three parts : the key value, a pointer to the rest of the data of that row, and a pointer to the next index element. To retrieve data sequentially, start at the first element (Chauhan) and follow the link (pointer) to the next element (Dhote). Each element of the index is found by following the link to the next element. The data pointer in each index element provides the entire datarow for that key value. The strength of a linked list lies in its ability to easily and rapidly insert and delete data.

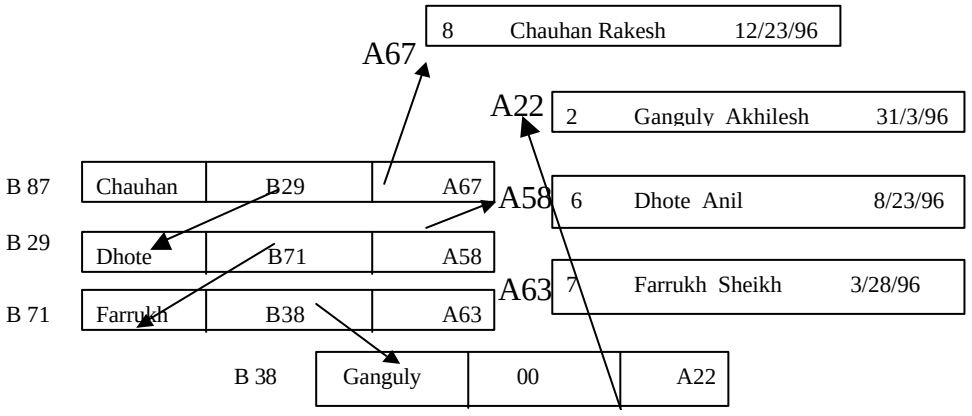

Fig. 2.4 : An illustration of linked list

8.3.2 Inverted Lists

Inverted lists may be viewed simply as index tables of pointers stored separately from the data records rather than embedded in pointer fields in the stored records themselves.

Distinction should be made between nondense and dense lists. In case of a nondense list only a few of the records in the file are part of the list while a dense list is one with a pointer for most or all of the records in the file.

Processing for unique secondary keys (those having 1:1 association with primary keys) is somewhat different than those with 1:M associations with primary keys. In the former case, dense indexes are generated while the later gives nondense indexes.

Examples of inverted lists are given below :

List 1

Company	Area	Primary Key
Digital	Computer	1245
Ford	Auto	1175
GM	Auto	1323
Intel	Computer	1231
Lockheed	Aerospace	1152

List 2

Company Symbol	Primary Key
DEC	1245
F	1175
GM	1323
INTL	1231
L	1152

Fig. 2.5 : Dense inverted lists

The above lists are dense since there is one-to-one relationship between both company name and primary key and company symbol and primary key.

Fig. 2.6 gives an example of non-dense inverted list for area (relationship between area and primary key is one-to-many).

Area	**Primary Key**
Aerospace	1152
Auto	1175, 1323
Computer	1231, 1245

Fig. 2.6: Nondense inverted list

The lists are said to be inverted because company names (or area names) have been alphabetized and the corresponding primary keys have been "inverted" or rearranged accordingly.

8.3.3 B-Trees

B-trees (also called B^+-trees) are a form of data structure based on hierarchies. Some author claim that the letter "B" stands for Bayer, the originator while others say it stands for "balanced". B-Trees are balanced in the sense that all the terminal (bottom) nodes have the same path length to the root (top). Algorithms have been developed for efficiently searching and maintaining B-Tree indexes, which have

become quite popular for representing both primary and secondary indexes. B-Trees provide both sequential and indexed access and are quite flexible.

The height of a B-Tree is the number of levels in the hierarchy. Each node on the tree contains an index element which has a key value, a pointer to the rest of the data and two link pointers (see Fig.2.7) One link (to the left) points to the elements (nodes) that have lower values while the other link (to the right) points to elements that have a value greater than or equal to the value in the node. The root is the highest node on the tree. The bottom nodes are called leaves because they are at the end of the tree branches.

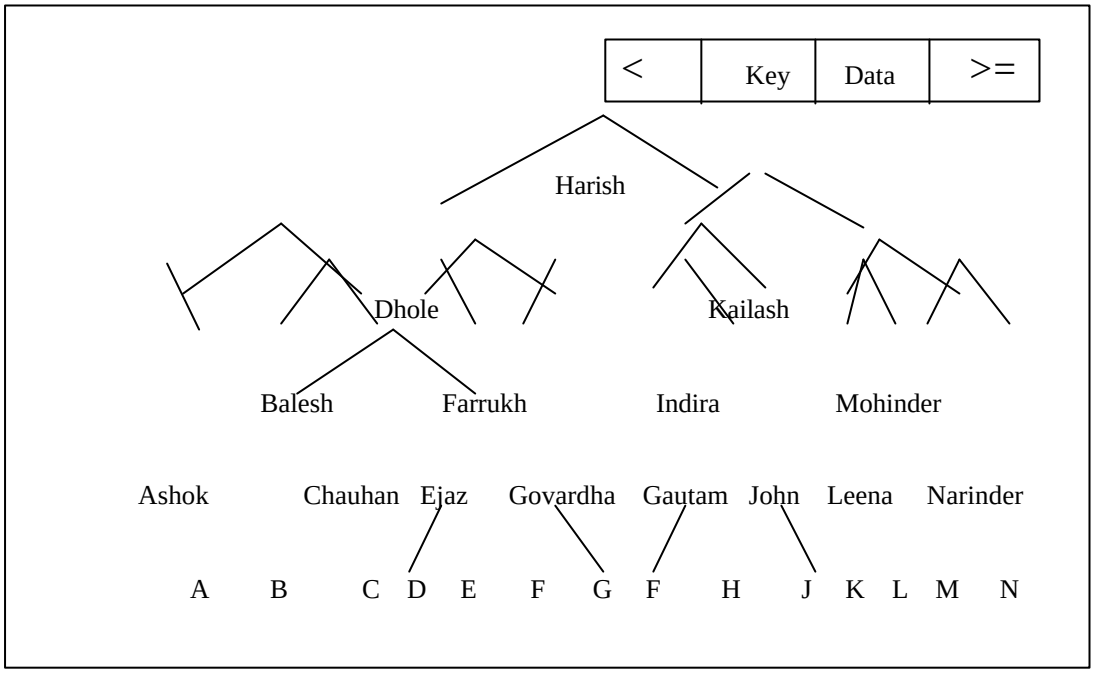

Fig. 2.7: An illustration of a B-tree

A B-tree is called unbalanced if the terminal nodes (leaves) are not all at the same level i.e. if different terminal nodes are at different depths below the top node.

Self Check Exercise

2) Give examples of commercially available database management systems which use B-trees ?

8.4 FILES AND THEIR ORGANIZATIONS

8.4.1 File Storage Concepts

A file is a sequence of records. File organization refers to physical layout or a structure of record occurrences in a file. File organization determines the way records are stored and accessed.

In many cases, all records in a file are of the same record type. If every record in the file has exactly the same size (in bytes), the file is said to be made of fixed-length records. If different records in the file have different sizes, the file is said to be made up of variable-length records. A file may have variable-length records for several reasons :

i) The file records are of the same record type, but one or more fields are of varying sizes (variable-length fields).

ii) The file records are of the same record type but one or more fields may have multiple values for individual records. Such a field is called repeating field and a group of values for the field is often called a repeating group.

iii) The file records are of the same record type, but one or more fields are optional.

iv) The file has records of different record types and hence of varying size (mixed file). This would occur if related records of different types were clustered (placed together) on disk blocks.

The records of a file must be allocated to disk blocks because a block is a unit of data transfer between disk and memory. The division of a track (on storage medium) into equal sized disk blocks is set by the operating system during disk formatting. The hardware address of a block comprises a surface number, track number and block number. Buffer – a contiguous reserved area in main storage that holds one block-has also an address. For a read command, the block from disk is copied into the buffer, whereas for a write command the contents of the buffer are copied into the disk block. Sometimes several contiguous blocks, called a cluster, may be transferred as a unit. In such cases buffer size is adjusted to cluster size.

When the block size is larger than the record size, each block will contain numerous records, while there can be files with large records that cannot fit in one block. In the latter case the records can span more than one block.

Here it is worthwhile to note the difference between the terms file organization and access method. A file organization refers to the organization of the data of a file into records, blocks and access structures; this includes the way the records and blocks are placed on the storage medium and interlinked. An access method on the other hand, provides a group of operations – such as find, read, modify, delete etc., — that can be applied to a file. In general, it is possible to apply several access methods to a file organization. Some access methods, though, can be applied only to files organised in certain ways. For example, we cannot apply an indexed access method to a file without an index.

8.4.2 Sequential Access Method (SAM)

In sequential files, the records are stored in a predefined order. Records which occur in a sequential file are usually sorted on the primary key and physically arranged on the storage medium in order by primary key. If only sequential access is required (which is rarely the case), sequential media (magnetic tapes) are suitable and probably the most cost-effective way of processing such files. Direct access devices such as disks may be but are not necessarily, referenced sequentially. Some types of processing are best done through sequential access, even when direct access devices are used.

Sequential access is fast and efficient while dealing with large volumes of data that need to be processed periodically. However, it is require that all new transactions be sorted into a proper sequence for sequential access processing. Also, most of the database or file may have to be searched to locate, store, or modify even a small number of data records. Thus, this method is too slow to handle applications requiring immediate updating or responses.

Sequential files are generally used for backup or transporting data to a different system. A sequential ASCII file is a popular export/import format that most database systems support.

8.4.3 Indexed Sequential Access Method (ISAM)

In indexed sequential files, record occurrences are sorted and stored in order by primary key on a direct access storage device. In addition, a separate table (or file) called an index is maintained on primary key values to give the physical address of each record occurrence. This approach gives (almost) direct access to record occurrences via the index table and sequential access via the way in which the records are laid out on the storage medium.

The physical address of a record given by the index file is also called a pointer. The pointer or address can take many forms depending on the operating system and the database one is using.

Today systems use virtual addresses instead of physical addresses. A virtual address could be based on imaginary disk drive layout. The database refers to a base set of tracks and cylinders. The computer then maps these values into actual storage locations. This arrangement is the basis for an approach known as the virtual sequential access method (VSAM). Another common approach is to define a location in terms of its distance from the start of a file (relative address). Virtual or relative addresses are always better than the physical address because of their portability.

In case a few records need to be processed quickly, the index is used to directly access the records needed. However, when large numbers of records must be processed periodically, the sequential organization provided by this method is used.

An illustration of access using index file is given in Fig. 2.8.

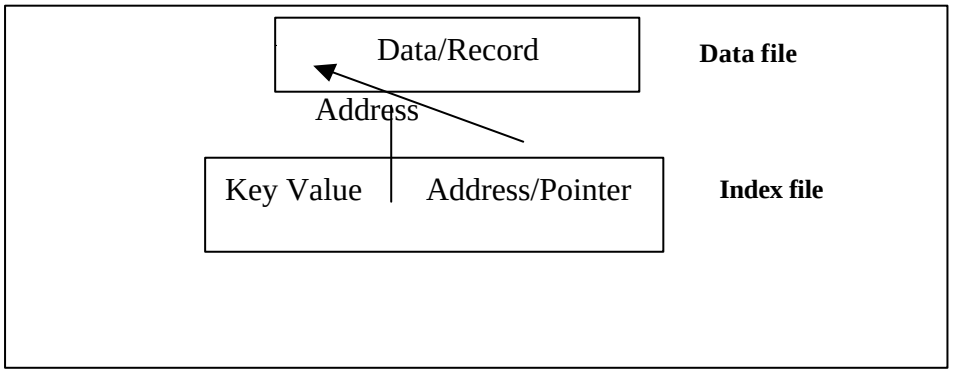

Fig. 2.8: Data access using index file

8.4.4 Direct Access Method (DAM)

When using the direct access method, the record occurrences in a file do not have to be arranged in any particular sequence on storage media. However, the computer must keep track of the storage location of each record using a variety of direct organization methods so that data is retrieved when needed. New transactions data do not have to be sorted, and processing that requires immediate responses or updating is easily handled.

In the direct access method, an alogrithm is used to compute the address of a record. The primary key value is the input to the algorithm and the block address of the record is the output.

To implement the approach, a portion of the storage space is reserved for the file. This space should be large enough to hold the file plus some allowance for growth. Then the algorithm that generates the appropriate address for a given primary key is devised. The algorithm is commonly called hashing algorithm. The process of converting primary key values into addresses is called key-to-address transformation.

More than one logical record usually fits into a block, so we may think of the reserved storage area as being broken into record slots sequentially numbered from 1 to n. These sequential numbers are called relative pointers or relative addresses, because they indicate the position of the record relative to the beginning of the file.

The objective of the hashing algorithm is to generate relative addresses that disperse the records throughout the reserved storage space in a random but uniform manner. The records can be retrieved very rapidly because the address is computed rather than found through table look-up via indexes stored on a disk file.

A collision is said to occur if more than one record maps to the same block. Since one block usually holds several records, collisions are only a problem when the number of records mapping to a block exceeds the block's capacity. To account for this event, most direct access methods support an overflow area for collisions which is searched sequentially.

The hashed key approach is extremely fast since the key's value is immediately converted into a storage location, and data can be retrieved in one pass to the disk.

An illustration of direct access method using hashed key is given in Fig. 2.9.

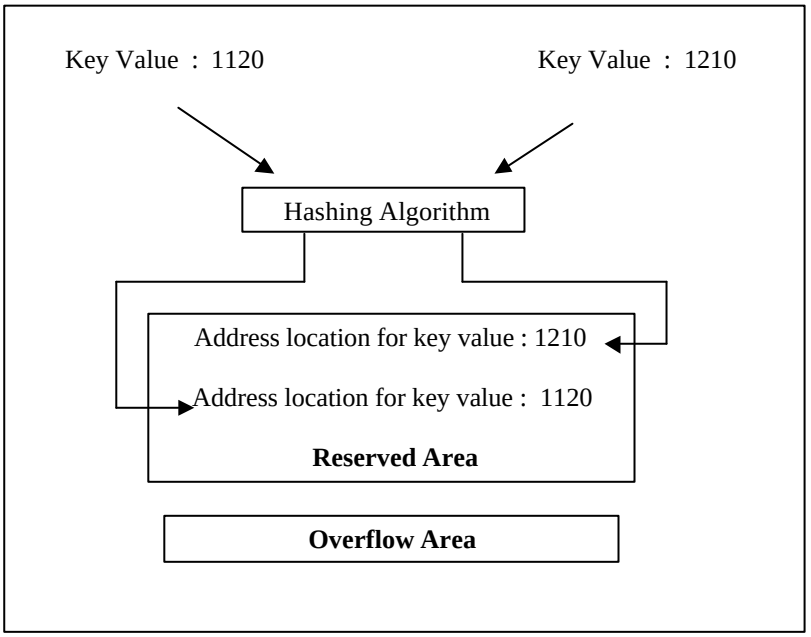

Fig. 2.9 : Direct access using hashed-key approach

8.5 SUMMARY

This Unit covers some of the key issues related to data structures and file organization and provides the essential background to facilitate their understanding. The role of data structures and file organization on the performance of access methods has been explained. RAID technology, binary search and indexes have been discussed

to elucidate how access speed can be improved. Typical data structures (linked lists, inverted lists and B-Trees) and file organization techniques (SAM, ISAM and DAM) have been dealt with.

This Unit lays the foundation for understanding the concepts involved in physical database design.

8.6 ANSWERS TO SELF CHECK EXERCISES

1) The existence of appropriate indexes is critical in data retrieval. In systems where query response time is a major consideration, indexes are used to speed up access. However, the presence of indexes tends to slow down the updating process.

2) A number of commercially available DBMS have B-Tree index creation features built into the system, i.e., the system automatically generates B-Tree indexes for speeding up and optimising queries. ORACLE and SYBASE are examples of relational database management systems, which support B-Tree indexes.

8.7 KEYWORDS

Cache Memory : A high speed temporary storage in the CPU for storing parts of a program or data during processing.

Binary Search : A search technique for sorted data.

B-Tree : An indexed data storage method that is efficient for a wide range of data access tasks.

Index : A sorted list of key values from the original table along with a pointer to the rest of the data in each row.

Pointer : A logical or physical address of a piece of data.

RAID : Redundant Array of Independent Disks (RAID) is a disk drive system that consists of multiple drives with independent controllers. The goal is to split the data to provide faster access and automatic duplication for error recovery.

8.8 REFERENCES AND FURTHER READING

Date, C.J. (1989). Introduction to Database Systems. New Delhi : Narosa Publishing House.

Gerald V. Post (2000). Database Management Systems. New Delhi: Tata McGraw-Hill Publishing Company Limited.

James, F. Courtney, David, B. Paradice. (1988). Database Systems for Management. Toronto: Times Mirror/Mosby College Publishing.

James A. O'Brien (1997). Introduction to Information Systems. Irwin: The McGraw-Hill Company.

Ramaz Elmasri, Shaukan B. Navathe (2000). Fundamentals of Database Systems: Pearson: Education Asia.