MIT-WPU

।। विश्वशान्तिर्ध्रुवं ध्रुवा ।।

Dr. Vishwanath Karad
MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

Data Structures-II

S. Y. B. Tech CSBS **Semester – III**

SCHOOL OF COMPUTER ENGINEERING AND TECHNOLOGY
B. Tech CSE - Computer Science & Business Systems

Data Structures-II

Pre-requisites: Data Structures-I

Course Objectives:

Knowledge:	(i) To understand different types of trees as data structure.
	(ii) To understand graph structure and various operations on graphs
Skills:	(i) To apply non-linear data structures to solve complex problems in various domains
	(ii) To apply different file organization methods and hashing
Attitude:	(i) To use appropriate non-linear data structures for solving real time problems

Course Outcomes: **After completion of the course the students will be able to :-**

1. To implement different types of trees and apply them to problem solutions
2. To implement various operations on graphs and their applicability
3. To apply and compare file organization methods and design different hashing functions
4. To apply the concept of nonlinear data structures for problem solving and its applications

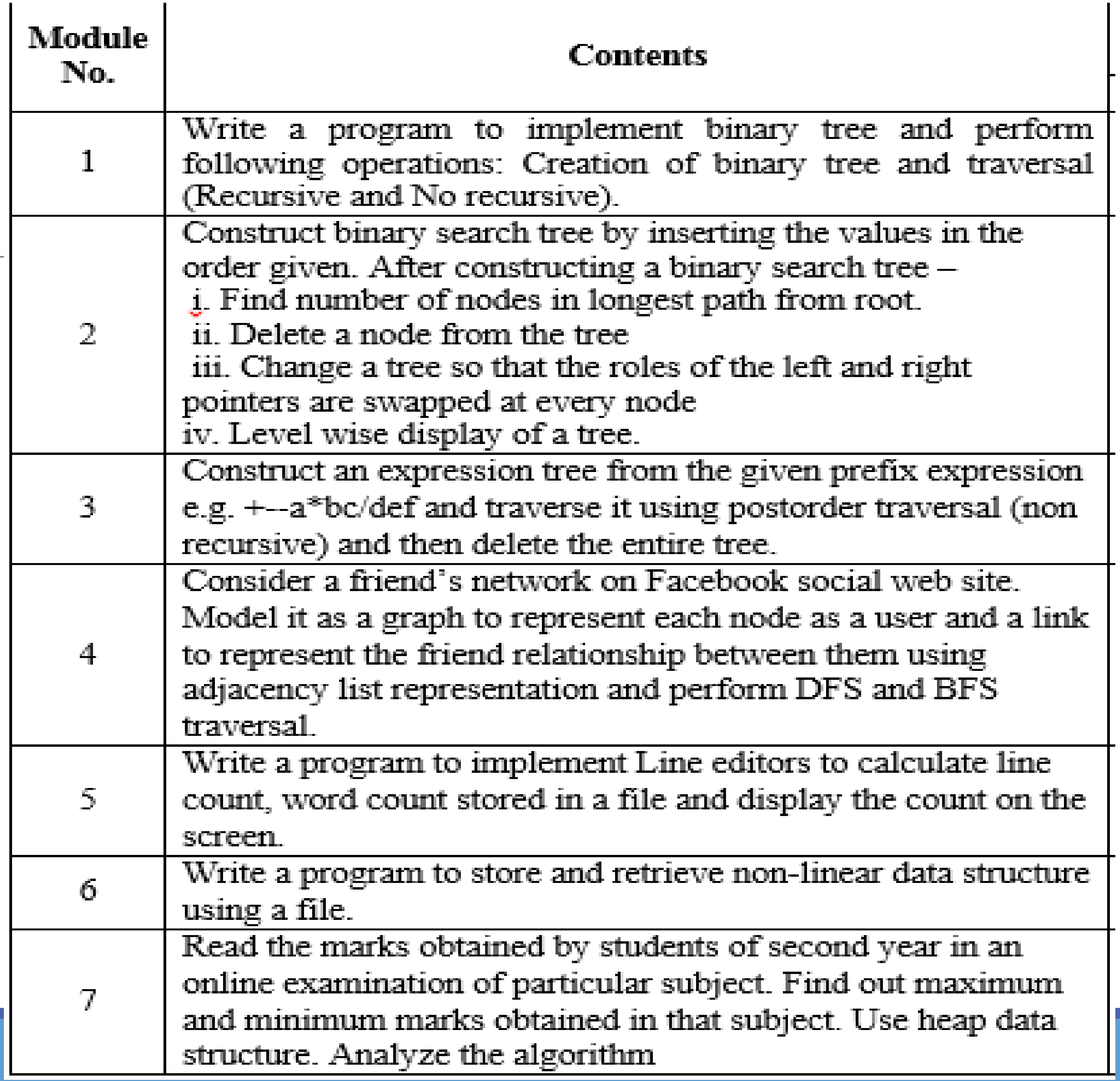
Syllabus

1. **Tree:** Basic Terminology, Binary Tree- Properties, Converting Tree to Binary Tree, Representation using Sequential and Linked organization, Binary tree creation and Traversals, Operations on binary tree. Binary Search Tree (BST) and its operations, Threaded binary tree- Creation and Traversal of In-order Threaded Binary tree. Comparison Trees. Case Study- Expression tree.
2. **Graph:** Basic Terminology, Graphs (Directed, Undirected), Various Representations, Traversals & Applications of graph- Prim's and Kruskal's Algorithms, Dijsktra's Single source shortest path, Analysis complexity of algorithm.
3. **Heap:** Heap as a priority queue, Heap sort, Height Balanced Tree: AVL tree, Splay Tree, Multiway search trees: B & B+ Tree.
4. **Hashing:** Concepts-hash table, hash function, basic operations, bucket, collision, probe, synonym, overflow, open hashing, closed hashing, perfect hash function, load density, full table, load factor, rehashing, issues in hashing, hash functions- properties of good hash function, division, multiplication, extraction, mid-square, folding and universal, Collision resolution strategies- open addressing and chaining, Hash table overflow- open addressing and chaining
5. **File Organization:** Sequential file organization- concept and primitive operations, Direct Access File- Concepts and Primitive operations, Indexed sequential file organization-concept, types of indices, structure of index sequential file

MIT-WPU

॥ विश्वशान्तिर्ध्रुवं ध्रुवा ॥

List of Assignments



Module No.	Contents
1	Write a program to implement binary tree and perform following operations: Creation of binary tree and traversal (Recursive and No recursive).
2	Construct binary search tree by inserting the values in the order given. After constructing a binary search tree – i. Find number of nodes in longest path from root. ii. Delete a node from the tree iii. Change a tree so that the roles of the left and right pointers are swapped at every node iv. Level wise display of a tree.
3	Construct an expression tree from the given prefix expression e.g. +--a*bc/def and traverse it using postorder traversal (non recursive) and then delete the entire tree.
4	Consider a friend's network on Facebook social web site. Model it as a graph to represent each node as a user and a link to represent the friend relationship between them using adjacency list representation and perform DFS and BFS traversal.
5	Write a program to implement Line editors to calculate line count, word count stored in a file and display the count on the screen.
6	Write a program to store and retrieve non-linear data structure using a file.
7	Read the marks obtained by students of second year in an online examination of particular subject. Find out maximum and minimum marks obtained in that subject. Use heap data structure. Analyze the algorithm

Learning Resources

Text Books:

1. Fundamentals of Data Structures, E. Horowitz, S. Sahni, S. A-Freed, Universities Press.
2. Data Structures and Algorithms, A. V. Aho, J. E. Hopperoft, J. D. UIlman, Pearson.

Reference Books:

1. The Art of Computer Programming: Volume 1: Fundamental Algorithms, Donald E. Knuth.
2. Introduction to Algorithms, Thomas, H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, The MIT Press.
3. Open Data Structures: An Introduction (Open Paths to Enriched Learning), (Thirty First Edition), Pat Morin, UBC Press.

Web links: https://www.tutorialspoint.com/data_structures_algorithms/

MOOCs: http://nptel.ac.in/courses/106102064/1

https://nptel.ac.in/courses/106103069/

File Organization

- Sequential file organization- concept and primitive operations,

- Direct Access File- Concepts and Primitive operations,

- Indexed sequential file organization-concept, types of indices, structure of index sequential file

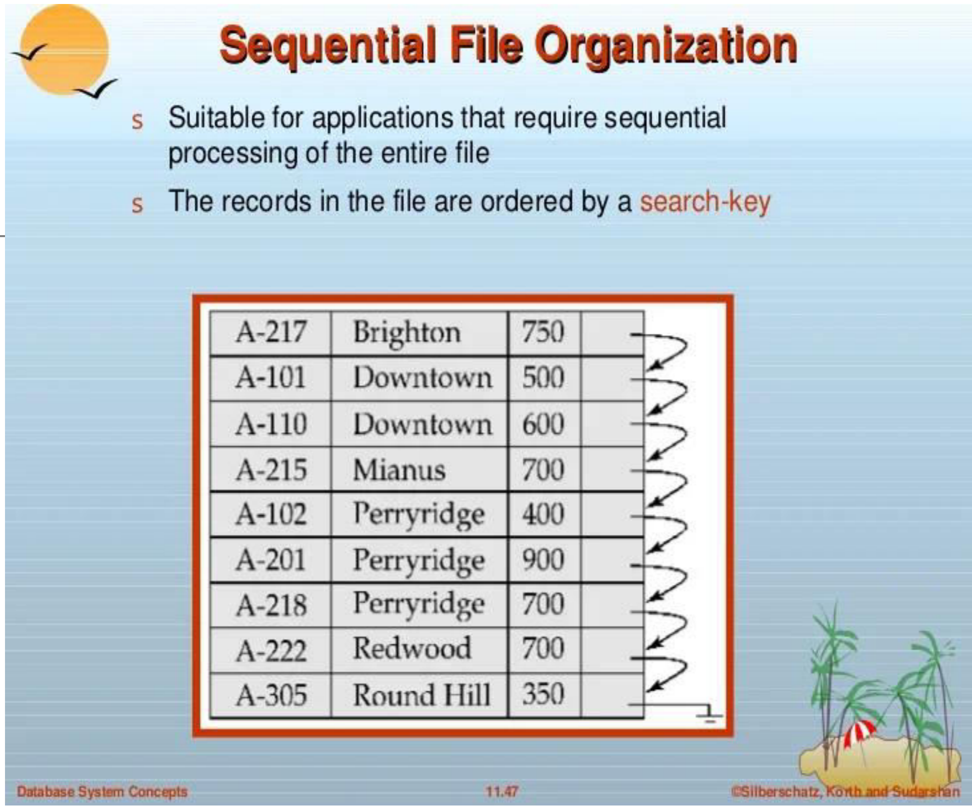
Introduction

- A file is a collection of records, each record having one or more fields
- The fields used to distinguish among the records are known as keys
- File organization describes the way where the records are stored in a file
- File organization is concerned with representing data records on an external storage media
- **Mode of Retrieval:**
  - Real time : response time for any query should be minimized.
  - Batched: Response time is not very significant.

FILE ORGANIZATION

- Number of keys: Files having only one key & files with more than one key.
- One key-records may be stored on this key and stored sequentially either on tape or disk. (batch retrieval)
- For more than one key, sequential organization is not adequate.
- Several indices have to be maintained

Sequential File Organization



Sequential File Organization

- Suitable for applications that require sequential processing of the entire file
- The records in the file are ordered by a search-key

A-217	Brighton	750
A-101	Downtown	500
A-110	Downtown	600
A-215	Mianus	700
A-102	Perryridge	400
A-201	Perryridge	900
A-218	Perryridge	700
A-222	Redwood	700
A-305	Round Hill	350

Database System Concepts 11.47 ©Silberschatz, Korth and Sudarshan

A sequential file stores records in the order they are entered.

The records are stored and sorted in physical contiguous blocks.

Within each block records are in sequence.

New records always appear at the end of file

Records can only be read or written sequentially

Records may be either fixed or variable in length

Search time associated with sequential files is more because records are accessed sequentially from beginning of the file.

Sequential files are compatible to the magnetic tape storage as shown in following figure

Record1	Record2	Record3	Record4	……	…	…	End

Primitive Operations

Open- Opens the file and sets file pointer to the first record

Read-next- returns next record to the user. If no record is present then EOF condition will be set.

Close- Closes the file and terminates access to the file

Write-next – File pointers are set to next of last record and this record is written to the file

EOF- if EOF occurs this operation returns true otherwise it returns false

Search- Searches for the record with a given key

Update- The Current record is written at the same position with updated values

Add-

It is one operation algorithm

The new record is appended at the end of the file

One physical write is required for appending a record in a file.

Also many records can be collected in the buffer and a block of records can be written at a time in the file

The following steps involved in addition

1. Open a file in append mode
2. Read a record from user
3. Write a record to the file
4. Close the file

Search-

A particular record is searched through the file using key sequentially by comparing with each record key.

The Search starts from first record and continues till the EOF

The following steps are involved in searching

1. Open a file in read mode
2. Read the value of the record key of the record to be searched
3. Read the next record from file
4. If record key=value, display record and go to 7
5. If not EOF, then go to 3
6. Display, ‘Record not found’
7. Close the file

Delete-

Deletion is done in two ways

- Logical deletion
- Physical deletion

Logical Deletion-

<u>Method1</u>

- When disk files are used, records may be logically deleted by just flagging them as having been deleted.
- This can be done by assigning a specific value to one of the attributes of the record
- This method needs one extra field to be maintained with each record
- The algorithm needs to modify and check the flag field during operations

<u>Method2</u>

- Keep a record of active and deleted records in a bit map file
- A bit is a one dimensional array in which each bit represents a record in a file
- The first bit refers to the first record & so on
- Bit value ‘1’ indicates record is active and ‘0’ indicates record is deleted.

Delete-

Method2-

```
1. Open a file in read + write mode
2. Read the record key of the record to be deleted
3. Read the next record from the file
4. If record key=value
       Change status or deleted flag as 1
       Write record back to the same position
       Go to Step7
5. If not EOF, then go to 3
6. Display ‘Record not found’
7. Close the file
```

Delete-

Physical Deletion(pack or reorganize)

Copy records to another file skipping the deleted records and rename the file

When number of logically deleted records is high, then it is advisable to delete them physically which is known as reorganization of file.

The steps involved are

1. Open a file in read mode
2. Open ‘temporary’ file in write mode
3. Read the record key of the record to be deleted
4. Read the next record from file
5. If record key! Value, write the record to temporary file
6. If not EOF, then go to 4
7. Close both the files
8. Delete the original file
9. Rename temporary file as original file
10. Close the file

Updation-

A record is updated when one or more fields is changed by modifying the information. The following steps are involved in updation:

1. Open a file in write mode
2. Read the record key of the record to be modified
3. Read the new attributes of the record to be modified
4. If record key=value ,modify record and go to 7
5. If not EOF, then go to 4
6. Display ‘Record not found’
7. Close the file

Advantages of Sequential File Organization

Owing to its simplicity, it can be used with a variety of media including magnetic tapes and disks.

It is compatible with variable length records, while most other file organizations are not.

Security is ensured with ease

For a run in which a high proportion of a block is hit, sequential file is efficient specially when processed in batches.

Drawbacks of Sequential File Organization

Insertion and deletion of records in between positions cause huge data movements.

Accessing any record requires a pass through all the preceding records which is time consuming. So, searching a record also takes more time

Needs reorganization of the file from time to time. If too many records are deleted logically then the file must be reorganized to free the space occupied by unwanted records.

Direct Access File

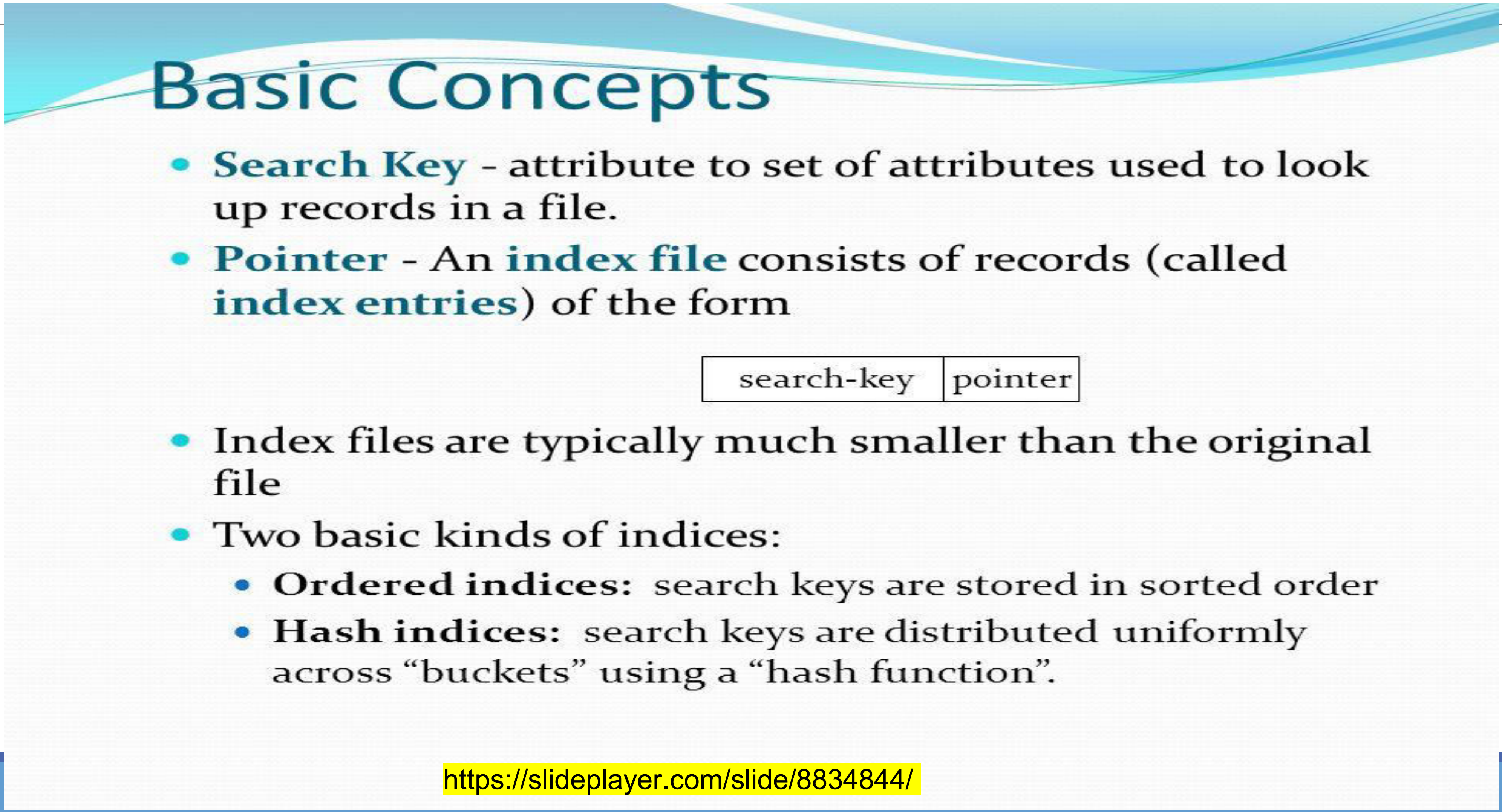
- Files that have been designed to make direct record retrieval as easy & efficient as possible are known as directly organized files.
- This is achieved by retrieving a record with a key by getting address of a record using the key.
- To achieve this, a suitable algorithm called as hashing is used to convert keys to addresses.
- They are often used in accessing large databases.

Operations on Direct Access File

- **Open-**Opens the file and sets the file pointer to the first record
- **Read-next-**It returns next record to user. If no record present then EOF will be set.
- **Read-direct-** sets file pointer to specific position and gets the record for the user. If slot is empty or out of range, then it gives error.
- **Write-direct-** it sets file pointer to specific position and write record to file at that position. user. If slot is out of range,then it gives error.
- **Update-** current record is written at the same position with updated values
- **Close-** this will terminate the access to the file
- **EOF-** if EOF occurs, it returns true else returns false

Indexing

- An index, whether it is a book or a data file index (in computer memory), is based on the basic concepts such as keys and reference fields
- The index to a book provides a way to find a topic quickly



Basic Concepts

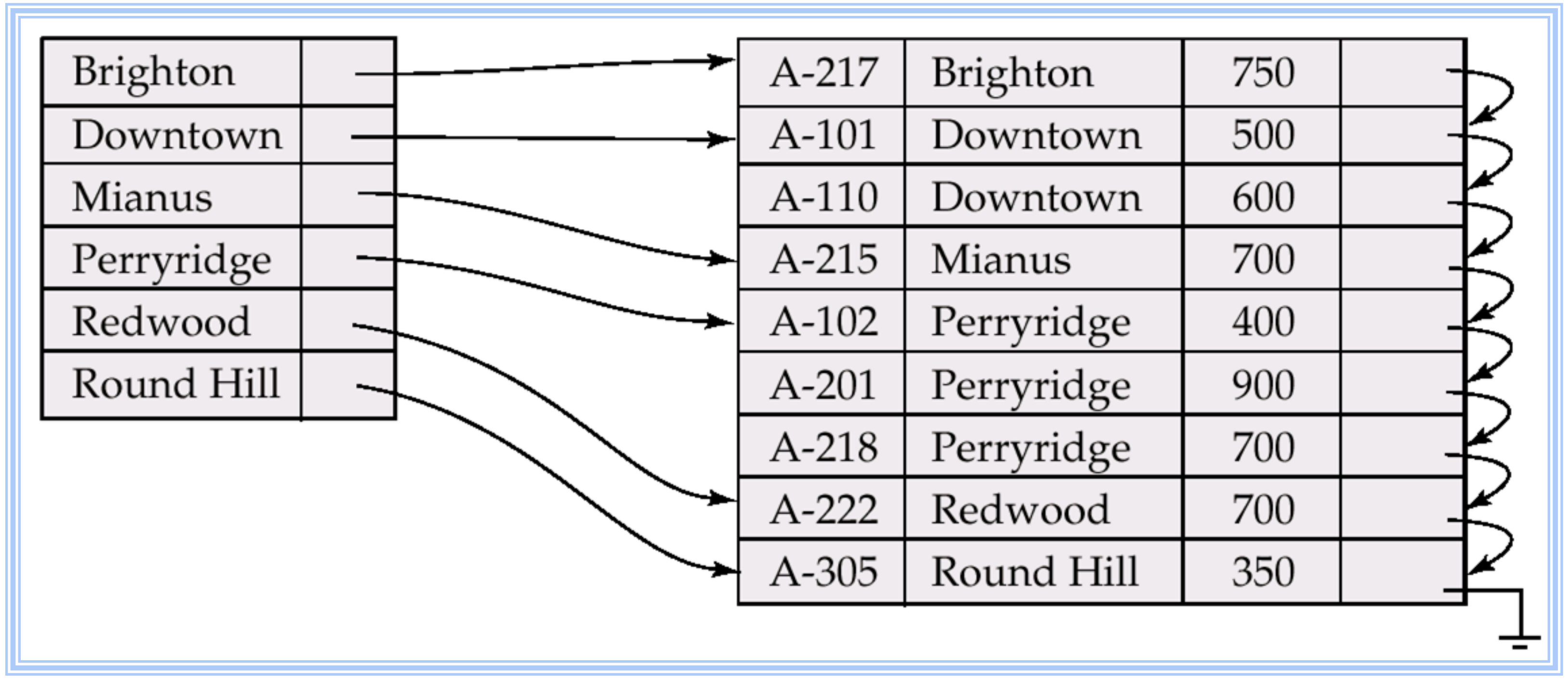
- **Search Key** - attribute to set of attributes used to look up records in a file.
- **Pointer** - An **index file** consists of records (called **index entries**) of the form

search-key	pointer

- Index files are typically much smaller than the original file
- Two basic kinds of indices:
  - **Ordered indices:** search keys are stored in sorted order
  - **Hash indices:** search keys are distributed uniformly across "buckets" using a "hash function".

https://slideplayer.com/slide/8834844/

Index Files Example



Brighton	
Downtown	
Mianus	
Perryridge	
Redwood	
Round Hill	

A-217	Brighton	750	
A-101	Downtown	500	
A-110	Downtown	600	
A-215	Mianus	700	
A-102	Perryridge	400	
A-201	Perryridge	900	
A-218	Perryridge	700	
A-222	Redwood	700	
A-305	Round Hill	350	

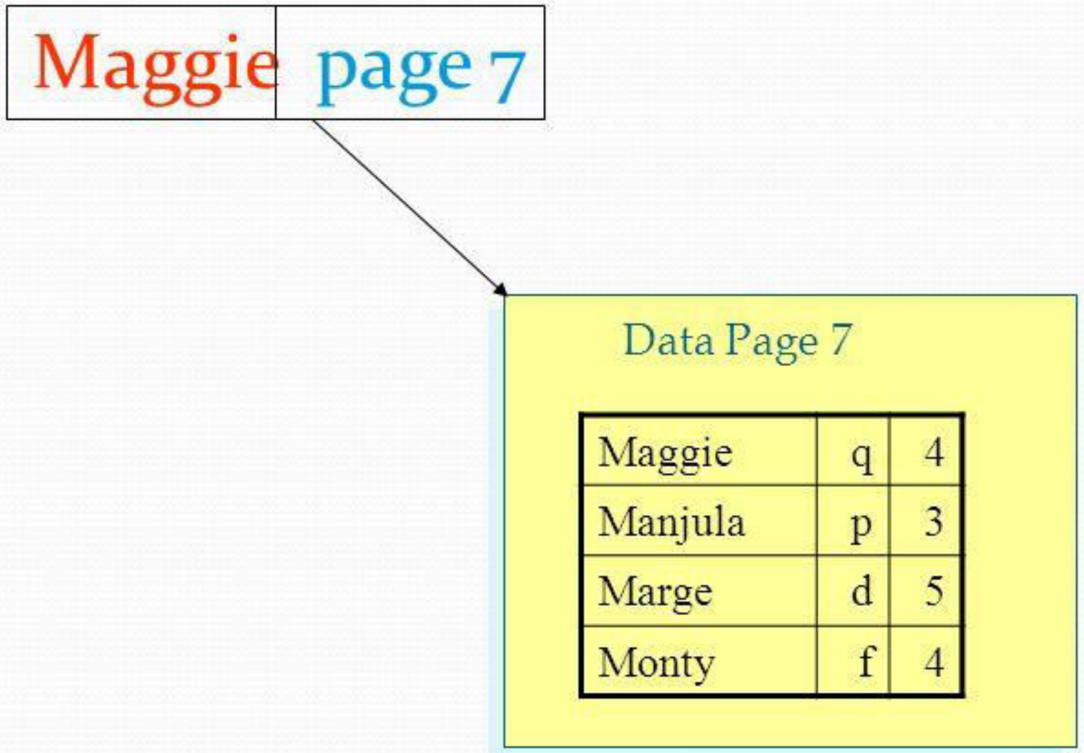
Indexed Sequential File Organization

- An index file contains records ordered by record key
- Each record contains a field that contains record key
- Record key uniquely identifies the record and determines the sequence in which it is accessed with respect to other records.
- An index file can use alternate indices that is record keys that let you the file using a different logical arrangement of the records.
- For example: file can be accessed through employee dept instead of emp number
- Index file read & written sequentially. Index is a data structure that allows particular record in a file to be located more quickly.

Indexed Sequential Access Method

- An **index entry** is a <key,pointer> pair, where key is the value of the first key on the page, and pointer, points to the page.

- Example



Maggie	q	4
Manjula	p	3
Marge	d	5
Monty	f	4

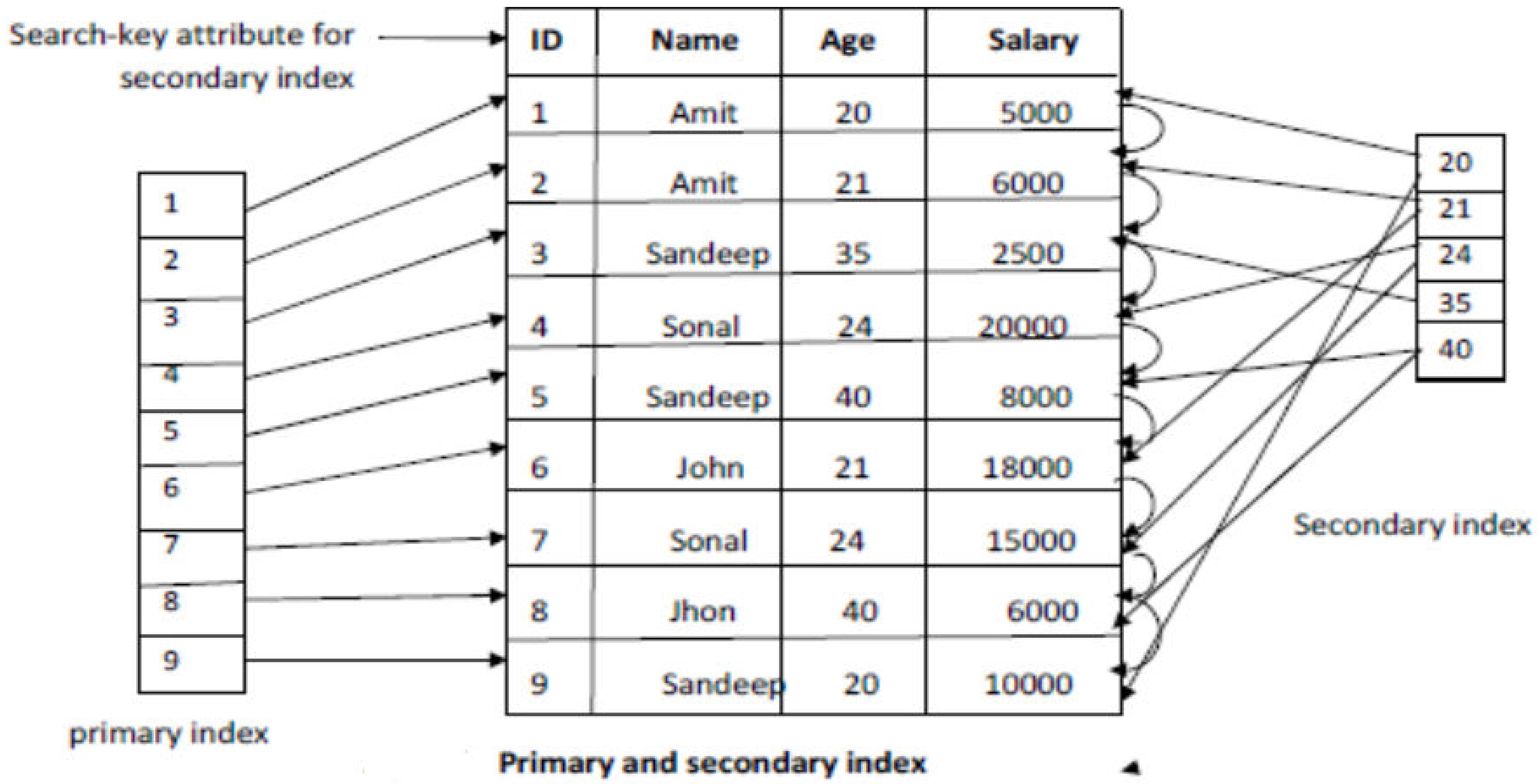
Types of Indices

- **Primary Index-** It is an index ordered in the same way as data file which is sequentially ordered according to a key. The indexing field is equal to this key

- **Secondary Index-** This is an index that is defined on a non ordering field of data file. In this case indexing field need not contain unique values

- **Clustering Index-** A data file can associate with utmost one primary index and several secondary indices. In this organization, key searches are improved. The single level indexing structure is simplest one where file, whose records are pairs, contains a key & pointer. This pointer is the position in the data file of the record with given key.

Types of Indices



ID	Name	Age	Salary
1	Amit	20	5000
2	Amit	21	6000
3	Sandeep	35	2500
4	Sonal	24	20000
5	Sandeep	40	8000
6	John	21	18000
7	Sonal	24	15000
8	Jhon	40	6000
9	Sandeep	20	10000

Primary and secondary index

Types of Indices…contd

- **Key search is performed as below**
- Search key is compared with index keys to find highest index key coming in front of search key while a linear search is performed from the record that the index key points to until the search key is matched or until the record pointed to by the next index entry is reached
- Hardware for indexed sequential file is usually disk based rather than tape.
- Records are physically ordered by primary key and index gives the physical location of the record.
- Records can be accessed sequentially or directly via index.
- Index is stored in file and read into memory at the point when the file is opened. The indices must also be maintained.

Structure of Indexed Sequential File

- In primary area, actual data records are stored. Data records are stored as sequential file
- Second area is an index area in which the index is stored and is automatically generated. An index file consist of three areas.
- **Primary Storage Area-** It includes unused space to allow for additions made in data
- **Separate Index or Indices-** Each query will reference this index first;it will redirect query to part of data file in which the target record is saved.
- **Overflow Area-** This is optional separate overflow area.

Structure of Indexed Sequential File…

- A number of index levels may be involved in index sequential file
- The lowest level is track index which is written at track 0, i.e first track of cylinder
- The track index contains two entries for each prime track of the cylinders for index sequential file
- The normal entry is composed of address of prime track to which each entry is associated and highest value of the keys for the records is stored on that track.
- The index indicates how records are distributed over number of cylinders
- In index sequential file, records are organized in sequence of key field known as primary key
- For fast searching, it is supported by index
- Index is a pair of key & address where that record is stored in main file
- Number of records are same as number of blocks of main file

Characteristics of Indexed Sequential File

Records are stored sequentially and a separate index file is maintained for accessing the record directly

Records can be accessed randomly in constant time

Magnetic tape is not suitable for indexed sequential storage

Index is the address of physical storage of a record

When a very few records are to be accesses, then indexed sequential file is better

This is a faster access method

Additional overhead is that the index is to be maintained

Indexed sequential files are popularly used in many applications such as digital library.

Hash Indices

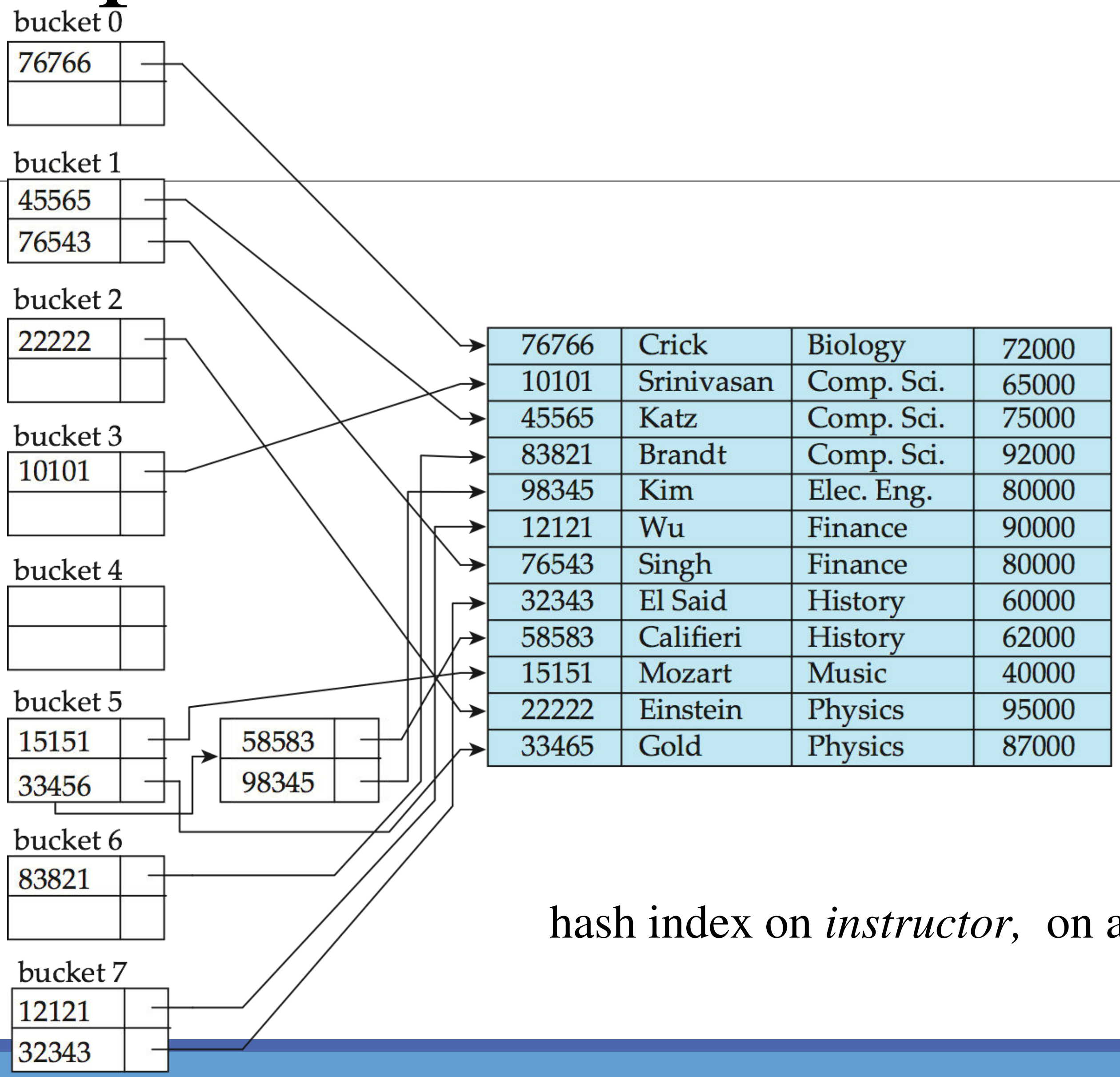
Hashing can be used not only for file organization, but also for index-structure creation.

A **hash index** organizes the search keys, with their associated record pointers, into a hash file structure.

Strictly speaking, hash indices are always secondary indices

- if the file itself is organized using hashing, a separate primary hash index on it using the same search-key is unnecessary.
- However, we use the term hash index to refer to both secondary index structures and hash organized files.

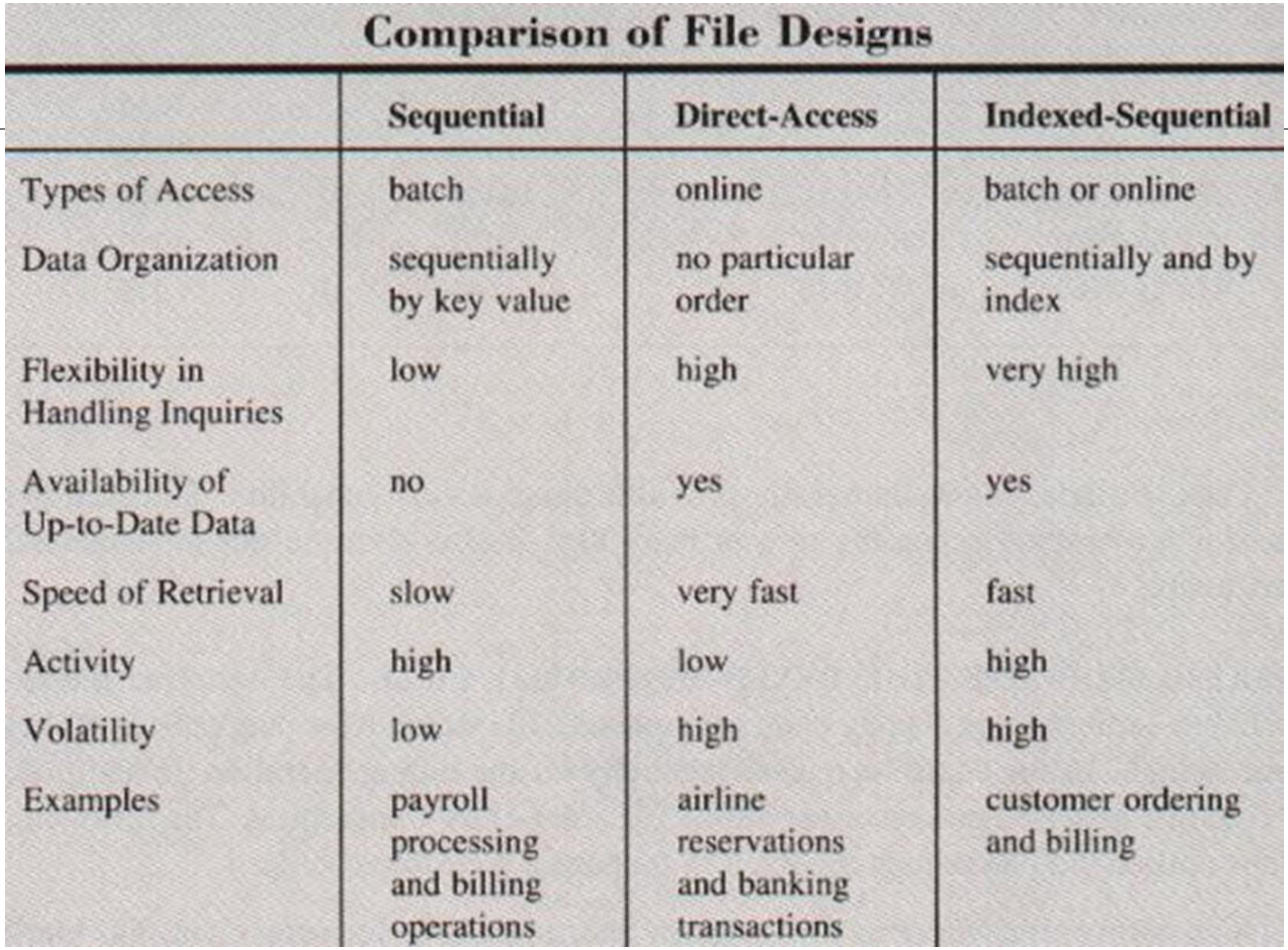
Example of Hash Index



76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
45565	Katz	Comp. Sci.	75000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000
12121	Wu	Finance	90000
76543	Singh	Finance	80000
32343	El Said	History	60000
58583	Califieri	History	62000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
33465	Gold	Physics	87000

hash index on *instructor*, on attribute *ID*

Comparison of File Organization



Comparison of File Designs

	Sequential	Direct-Access	Indexed-Sequential
Types of Access	batch	online	batch or online
Data Organization	sequentially by key value	no particular order	sequentially and by index
Flexibility in Handling Inquiries	low	high	very high
Availability of Up-to-Date Data	no	yes	yes
Speed of Retrieval	slow	very fast	fast
Activity	high	low	high
Volatility	low	high	high
Examples	payroll processing and billing operations	airline reservations and banking transactions	customer ordering and billing

File Handling in C

C provides a number of build-in function to perform basic file operations:

- **fopen()** - create a new file or open a existing file
- **fclose()** - close a file
- **getc()** - reads a character from a file
- **putc()** - writes a character to a file
- **fscanf()** - reads a set of data from a file
- **fprintf()** - writes a set of data to a file
- **getw()** - reads a integer from a file
- **putw()** - writes a integer to a file
- **fseek()** - set the position to desire point
- **ftell()** - gives current position in the file
- **rewind()** - set the position to the beginning point

Opening a file

The fopen() function is used to create a file or open an existing file:

fp = fopen(filename,file mode);

There are many modes for opening a file:

- r - open a file in read mode
- w - opens or create a text file in write mode
- a - opens a file in append mode
- r+ - opens a file in both read and write mode
- a+ - opens a file in both read and write mode
- w+ - opens a file in both read and write mode

Ex:

```
#include<stdio.h>
main()
{
FILE *fp;
char ch;
fp = fopen("hello.txt", "w");
printf("Enter data");
while( (ch = getchar()) != EOF) {
  putc(ch,fp);
}
fclose(fp);
fp = fopen("hello.txt", "r");

while( (ch = getc(fp)! = EOF)
{
  printf("%c",ch);
}

fclose(fp);


}
```

Example: Program to Open a File, Read from it using **fgets()**, And Close the File

```
# include <stdio.h>
# include <string.h>
int main( )
{
      FILE *FP ;
        char data[50];
    FP = fopen("A1.c", "r") ;
      if ( FP == NULL )
        {
        printf( "A1.c file not exist”) ;
        }
    else
        {
        printf("Reading Data from File….\n")
        while( fgets (data, 50, FP ) != NULL )
        {
            printf( "%s" , data) ;
        }
        fclose(FP) ;
        printf("Data read from the file.\n");
      }
    return 0;
}
```

```
#include<stdio.h>
int main()
{
    char string[20];
    FILE *fp;
    char str[] = "C programming.";
    fp = fopen( "test.txt" , "w" );
    fwrite(str , 1 , sizeof(str) , fp );
    fclose(fp) ;

        fp=fopen("test.txt","r");
    fgets(string,10,fp);
    printf("The string is: %s",string);
    fclose(fp);
    return 0;
}
```

fseek() is used to move file pointer associated with a given file to a specific position.

Syntax:

int fseek(FILE *pointer, long int offset, int position)

pointer: pointer to a FILE object that identifies the stream.

offset: number of bytes to offset from position

position: position from where offset is added.

returns:
zero if successful, or else it returns a non-zero value

Position defines the point with respect to which the file pointer needs to be moved. It has three values:

SEEK_END : It denotes end of the file.

SEEK_SET : It denotes starting of the file.

SEEK_CUR : It denotes file pointer's current position.

Example:

```
#include <stdio.h>

int main()
{
   FILE *fp;
   fp = fopen("test.txt", "r");

   // Moving pointer to end
   fseek(fp, 0, SEEK_END);

   // Printing position of pointer
   printf("%ld", ftell(fp));

   return 0;
}
```

ftell() in C is used to find out the position of file pointer in the file with respect to starting of the file. Syntax of ftell() is:

long ftell(FILE *pointer)

Below program create a file file.txt writes the contents *This is tutorialspoint.com* in it. Later we had reset the write pointer at 7th position from the beginning and used puts() statement with *C Programming Language* which over-write the file with the following content − *This is C Programming Language*

```
#include <stdio.h>

int main () {
   FILE *fp;

   fp = fopen("file.txt","w+");
   fputs("This is tutorialspoint.com", fp);

   fseek( fp, 7, SEEK_SET );
   fputs(" C Programming Language", fp);
   fclose(fp);

   return(0);
}
```

lseek()

lseek (C System Call): lseek is a system call that is used to change the location of the read/write pointer of a file descriptor. The location can be set either in absolute or relative terms.

Function Definition

off_t lseek(int fildes, off_t offset, int whence);

Field Description

int fildes : The file descriptor of the pointer that is going to be moved

off_t offset : The offset of the pointer (measured in bytes).

int whence : The method in which the offset is to be interpreted (rela, absolute, etc.). Legal value r this variable are provided at the end.

return value : Returns the offset of the pointer (in bytes) from the beginning of the file. If the return value is -1, then there was an error moving the pointer.

Ex: lseek (fp, n, SEEK_CUR);

References

1. Horowitz, Sahani, Dinesh Mehta, “Fundamentals of Data Structures in C++”, Galgotia Publisher, ISBN: 8175152788, 9788175152786.

2. Peter Brass, “Advanced Data Structures”, Cambridge University Press, ISBN: 978-1-107-43982