Visvesvaraya Technological University
Belagavi-590 018, Karnataka

A Mini Project Report on

"Indexing For Movies Data Set"

Mini Project Report submitted in partial fulfilment of the requirement for the File Structure Lab [17ISL68]

Bachelor of Engineering
In
Information Science and Engineering

Submitted by
POORNIMA P[1JT17IS026]

Under the guidance of
Mr.Vadiraja.A
Assistant Professor, Department of ISE

Department of Information Science and Engineering
Jyothy Institute of Technology
Tataguni, Bengaluru-560082
2020-2021

Jyothy Institute of Technology

Tataguni, Bengaluru-560082

Department of Information Science and Engineering

CERTIFICATE

Certified that the mini project work entitled **“INDEXING”** carried out by **POORNIMA P [1JT17IS026]** bonfire student of Jyothy Institute of Technology, in partial fulfilment for the award of **Bachelor of Engineering** in **Information Science and Engineering** department of the **Visvesvaraya Technological University, Belagavi** during the year **2020-2021**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

Mr . Vadiraja A	**Dr. Harshvardhan Tiwari**
Guide, Asst. Professor	**Associate Professor and HOD**
Dept. Of ISE	**Dept. Of ISE**

External Viva Examiner

Signature with Date :

1.
2.

ACKNOWLEDGEMENT

Firstly, we are very grateful to this esteemed institution **“Jyothy Institute of Technology**” for providing us an opportunity to complete our project.

We express our sincere thanks to our Principal **Dr. Gopalakrishna K for** providing us with adequate facilities to undertake this project.

We would like to thank **Dr. Harshawardhan Tiwari, Associate Professor and Head** of Information Science and Engineering Department for providing for his valuable support.

We would like to thank our guide **Mr.Vadiraja A, Asst. Prof.** for his keen interest and guidance in preparing this work.

Finally, we would thank all our friends who have helped us directly or indirectly in this project.

POORNIMA P

[1JT17IS026]

ABSTRACT

Record management carries significant importance in the organization. Record management is concerned with keeping the records safely and providing as per the requirement. Indexing is such a instrument of record management which aims to maintain the storage space and improve the efficiency of the system. Indexing is a data structure technique to efficiently retrieve records from the files based on some attributes on which the indexing has been done. An index in which the entries are a key ordered linear list. The index file could be sorted or organised using a tree structure, thereby imposing a logical order on the records without physically rearranging them. Each record of a database normally has a unique identifier ,called the primary key. A particular key value might be duplicated in multiple records, is called a secondary key. The secondary key index will associate a secondary key value with the primary key of each record having that secondary key value . The full database might be searched directly for the record with that primary key, or there might be a primary key index that relates each primary key value with a pointer to the actual record on the disk. In this case, the primary index provides the location of the actual record on disk, while the secondary disk indices refer to the primary index. Indexing is an very important technique for organising large databases.

Table of Contents

CHAPTER 1
INTRODUCTION

CHAPTER 1
INTRODUCTION

1.1 Introduction To File Structures

File Structures is the Organization of Data in Secondary Storage Device in such a way that minimizes the access time and the storage space. A file Structure is a combination off representations for data in files and operations for accessing the data. A file structure typically allows applications to read, write and modify data. It might also support finding the data that matches some search criteria or reading through the data in some particular order.

The key design problem that shapes file structure design is the relatively large amount of time that is required to get information from disk. All file structure designs focus on minimizing disk accesses and maximizing the likelihood that the information the user will want is already in RAM.

The fundamental operations of file systems: OPEN( ), CREATE( ), CLOSE( ), READ( ), WRITE( ), and SEEK( ). Each of these operations involves the creation or use of a link between a physical file stored on a secondary device and a logical file that represents a program's more abstract view of the same file. When the program describes an operation using the logical file name, the equivalent physical operation gets performed on the corresponding physical file.

It is relatively easy to come up with file structure designs that meet these goals when we have files that never change. Designing file structures that' maintain these qualities as files change, growing and shrinking as information is added and deleted, is much more difficult.

The choice of a file structure is often influenced by factors such as:

- Frequency of update
- File activity
- File Access Method
- Nature of the System

1.2 Introduction To Python

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991. Python provides constructs that enable clear programming on both small and large scales. Van Rossum led the language community until July 2018.

Python is dynamically typed and garbage collected. It supports multiple programming paradigms, including procedural, object-oriented and functional programming. Python features a comprehensive standard library and is referred to as 'batteries included'.

Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open-source software and has a community based development model. Python and CPython are managed by the non-profit Python Software Foundation.

1.3 Introduction To Indexing

Indexing is a way to optimize the performance of a database by minimizing the number of disk accesses required when a query is processed. It is a data structure technique which is used to quickly locate and access the data in a database.

Indexes are created using a few database columns.

The first column is the **Search key** that contains a copy of the primary key or candidate key of the table. These values are stored in sorted order so that the corresponding data can be accessed quickly. The second column is the **Data Reference** or **Pointer** which contains a set of pointers holding the address of the disk block where that particular key value can be found.

1.3.1 Introduction To Primary Indexing And Secondary Indexing

Primary Indexing

If the index is created on the primary key then it is called as primary indexing. Since these primary keys are unique to each record and it has 1:1 relation between the records, it is much easier to fetch the record using primary indexing. There are two types of primary indexing – Dense index and Sparse index.
Dense index- indexing is created based on primary as well as on columns which means user
The dense index addresses quick search on any search , the space used for index and address
can fire query based on even columns. Hence index on all search key columns are stored.
becomes overhead in the memory . Therefore more space is consumed to store the indexes
the record size increases.
Sparse index- in this method of indexing, range of index columns store the same data block address and when the data is to be retrieved, the block address will be fetched linearly till we get the requested data.

Secondary Indexing

In sparse indexing as the table size grows, the(index, address) mapping file size also grows which results in slower fetching of address. To overcome this problem secondary indexing is introduced.

In secondary indexing method, another level of indexing is introduced to reduce the mapping size. That means initially huge range for columns are selected so that the first level of mapping is small. Then each range is divided into smaller ranges. First level of mapping is stored in the primary memory so that address can be fetched faster. Secondary level of mapp ing and the actual data are stored in the secondary memory.

1.4 Importance of Indexing

- Easy location – Indexing points out the required records or file and facilitates easy location.
- Saves time and effort – Indexing gives the ready reference to the records and saves time and efforts of the office.
- Efficiency – Indexing helps to find out the records easily and quickly which enhances the efficiency.
- Reduce costs – Indexing helps to reduce the cost by saving time and efforts.
- Cross Reference – A particular record can be maintained through two ways. Indexing facilitates to find out such records through cross reference.

Index is not only necessary to large office but also necessary for small office. When a   large number of files are maintained, the necessity of maintaining index is increased. Indexing increases the utility of filing by providing an easy reference to files. The very purpose of maintaining index is that it is easy and location of filing is faster.

CHAPTER 2
DESIGN AND IMPLEMENTATION

CHAPTER 2
DESIGN AND IMPLEMENTATION

2.1 Algorithm Of Indexing

Step 1: Creation of a file containing a particular dataset.

Step 2: Input -Takes a search key as input.

Step 3: Output - Efficiently returns a collection of matching records.

Step 4: The first column is the search key that contains a copy of primary key or candidate key of the table.

Step 5: The second column is the pointer which contains a set of pointers holding the address of the disk block where that particular key value is found.

Step 6: Record addition - This consists of appending the data file and inserting a new record. The rearrangement of the index consists of sliding down the records with keys larger than the inserted key and then placing new record in the opened space.

Step 7: Record deletion- This should use the techniques for reclaiming space in files when deleting from the data file. We must delete the corresponding entry from the index. Shift all records with keys larger than key of the deleted record to the previous position in memory or make the index entry as deleted.

Step 8: In our record file we built an index for b_id which is primary key and there is author name as secondary key.

Step 9: Record addition in secondary indexing :When adding a record entry must also be added to the secondary key index. There may be duplicates in secondary key, keep duplicates in sorted order of primary key.

Step 10: Record deletion in secondary key : Deleting a record implies removing all the references to the record in primary index and in all secondary indexes. When accessing the file through secondary key ,the primary indexed file will be checked and a deleted record can be identified.

Step 11: It allows binary search to obtain a keyed access to a record in variable length record file.

Step 12: Time taken for dataset has been calculated for each functionality.

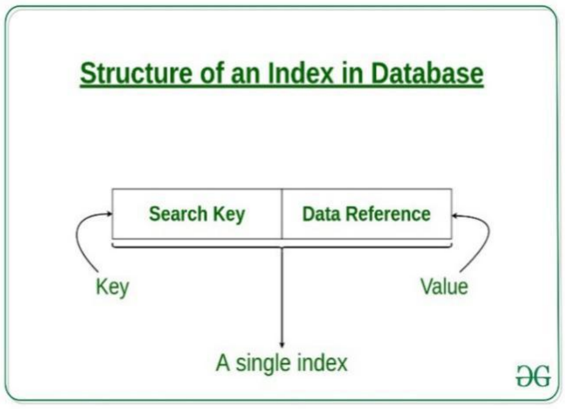


Fig 2.1.1

2.2 Basic Operations On Indexing

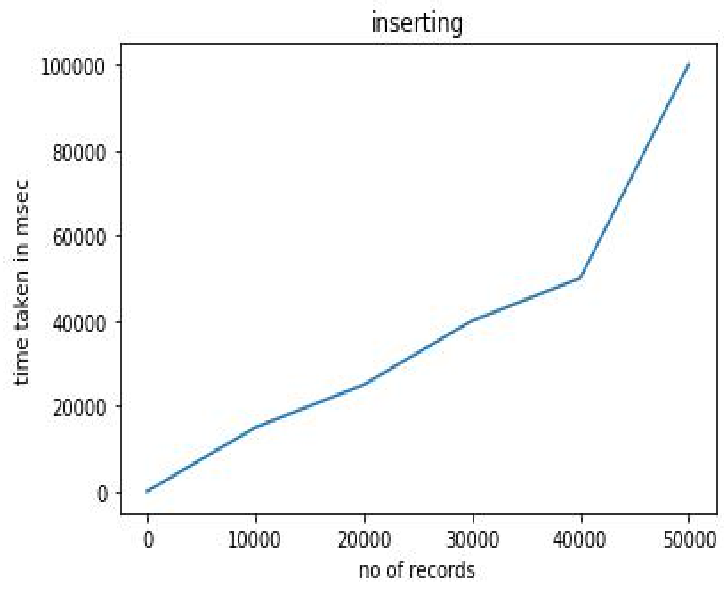
The following are the basic operations performed on indexing:

- Entering the details.
- Searching.
- Deleting.
- Build Index

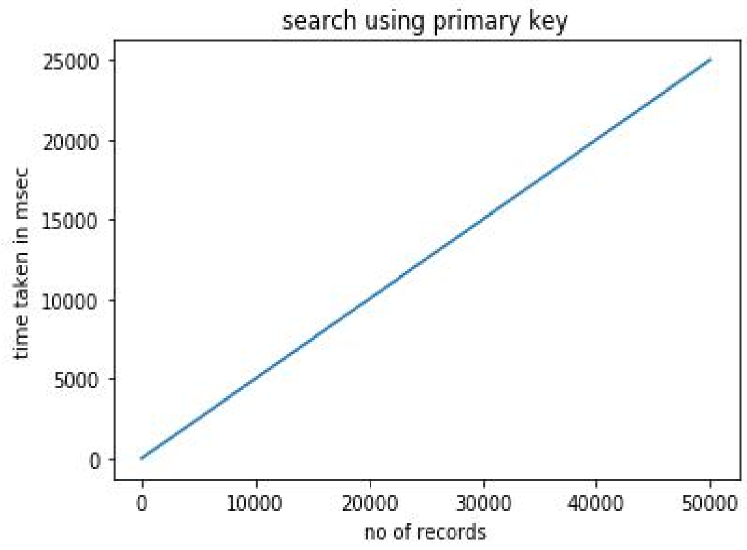
CHAPTER 3
TIME ANALYSIS OF INDEXING

CHAPTER 3
TIME ANALYSIS OF INDEXING

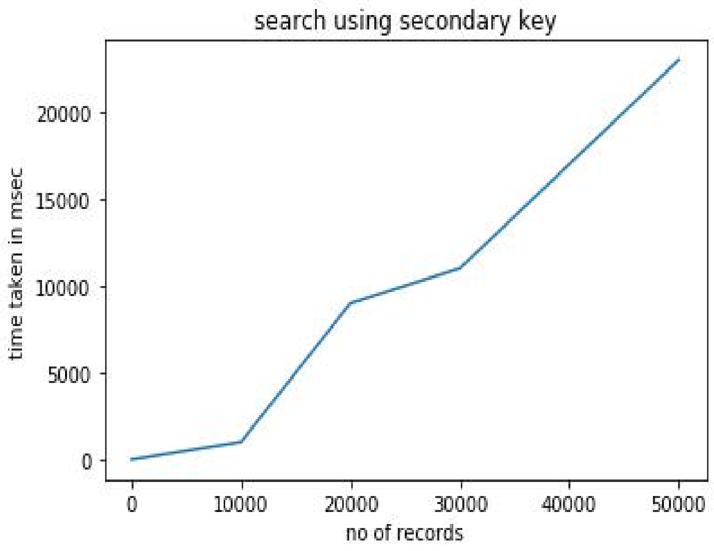
3.1 Time Analysis



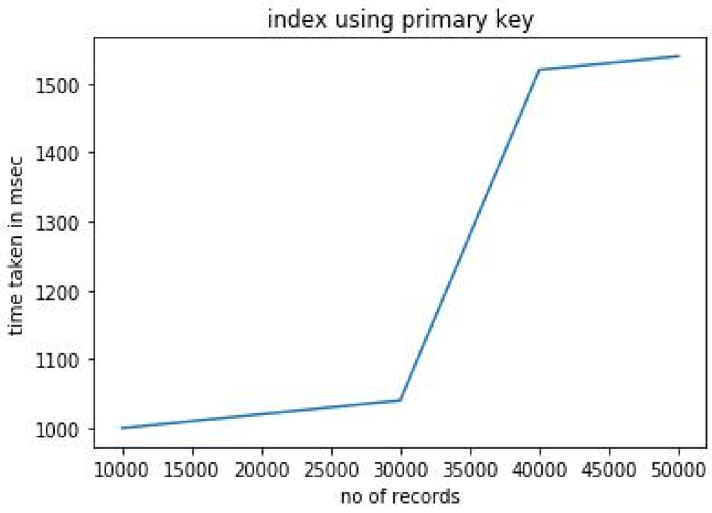
3.1.1 : The time taken to insert a record into the file, as the number of records increase the time also increases.



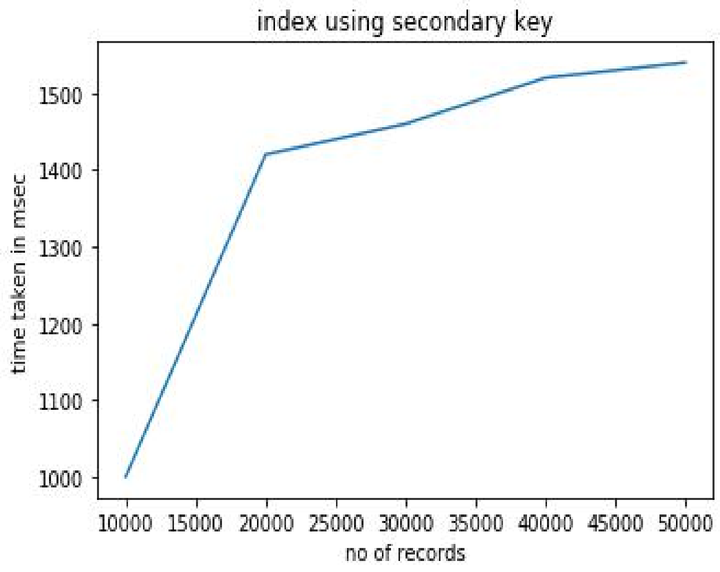
3.1.2 : Time analysis for searching a record in the file by using primarykey, as the number of records increase the time will be delayed to search record.



3.1.3 : Time analysis for searching a record in the file by using Secondary Key, as the number of records increase the time will be delayed to search record.



3.1.4 : Building index using Primary key.

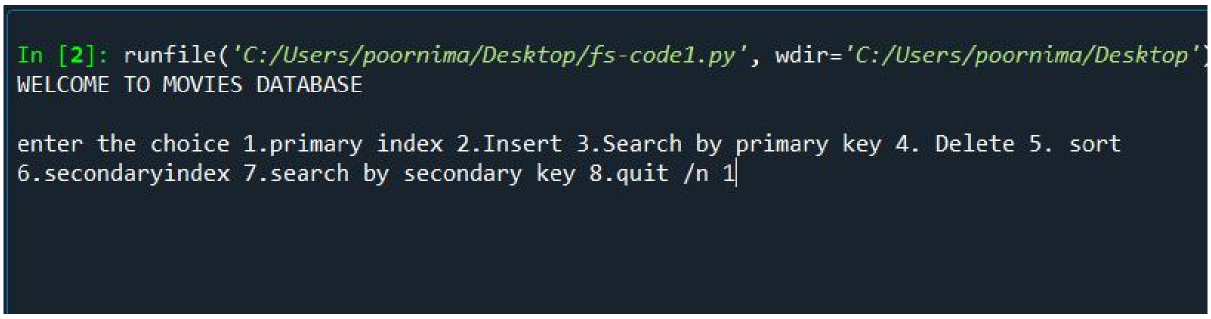


3.1.5 :Building index using secondary key.

CHAPTER 4
RESULTS AND SCREENSHOTS

CHAPTER 4
RESULTS AND SCREENSHOTS

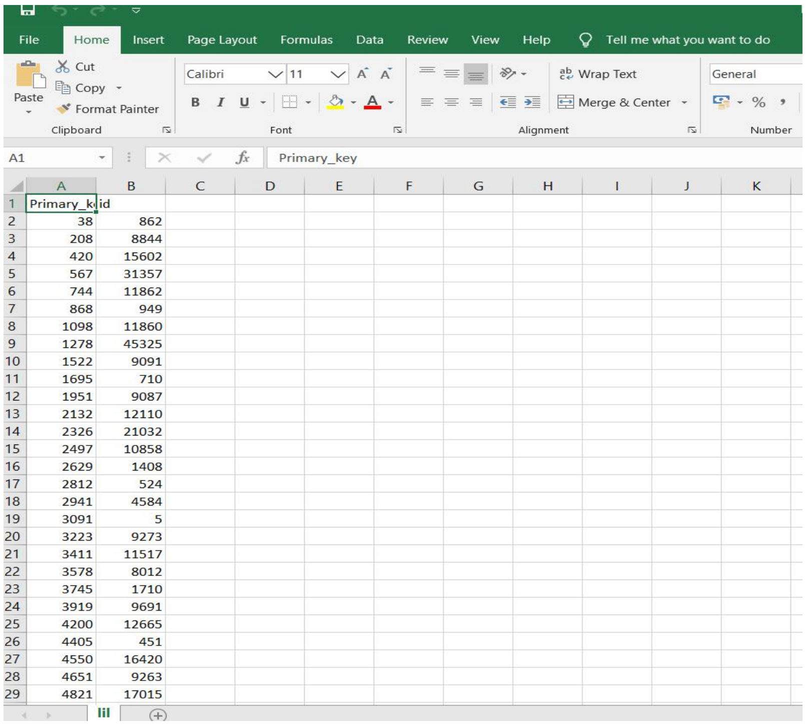
4.1 Primary Indexing



```
In [2]: runfile('C:/Users/poornima/Desktop/fs-code1.py', wdir='C:/Users/poornima/Desktop')
WELCOME TO MOVIES DATABASE

enter the choice 1.primary index 2.Insert 3.Search by primary key 4. Delete 5. sort
6.secondaryindex 7.search by secondary key 8.quit /n 1
```

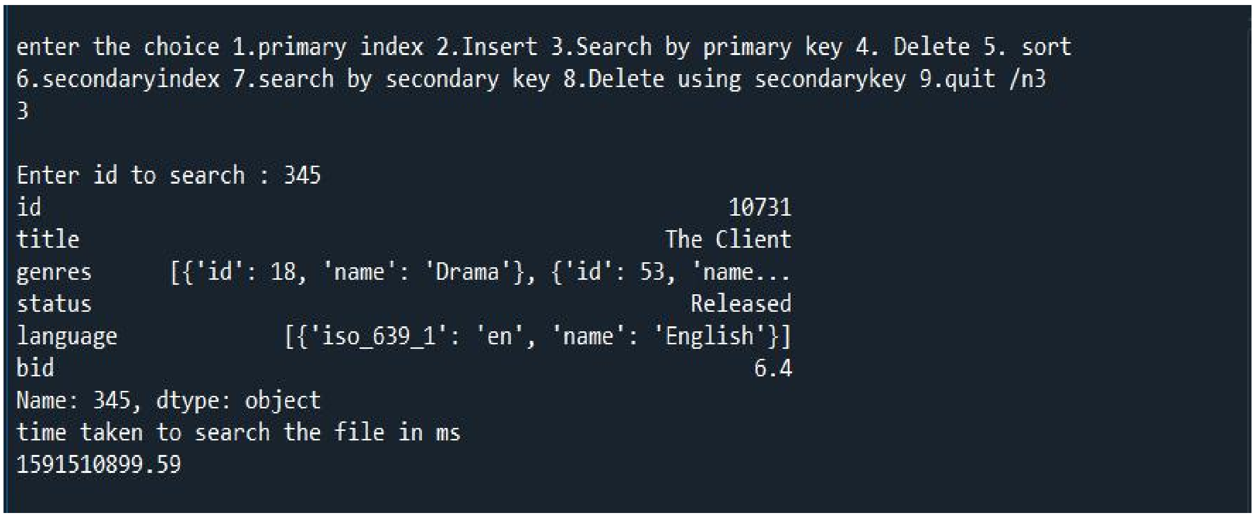
By selecting option 1 you can perform primary indexing

Output :



	A	B
1	Primary_k	id
2	38	862
3	208	8844
4	420	15602
5	567	31357
6	744	11862
7	868	949
8	1098	11860
9	1278	45325
10	1522	9091
11	1695	710
12	1951	9087
13	2132	12110
14	2326	21032
15	2497	10858
16	2629	1408
17	2812	524
18	2941	4584
19	3091	5
20	3223	9273
21	3411	11517
22	3578	8012
23	3745	1710
24	3919	9691
25	4200	12665
26	4405	451
27	4550	16420
28	4651	9263
29	4821	17015

4.1.1 Searching Using Primary Key

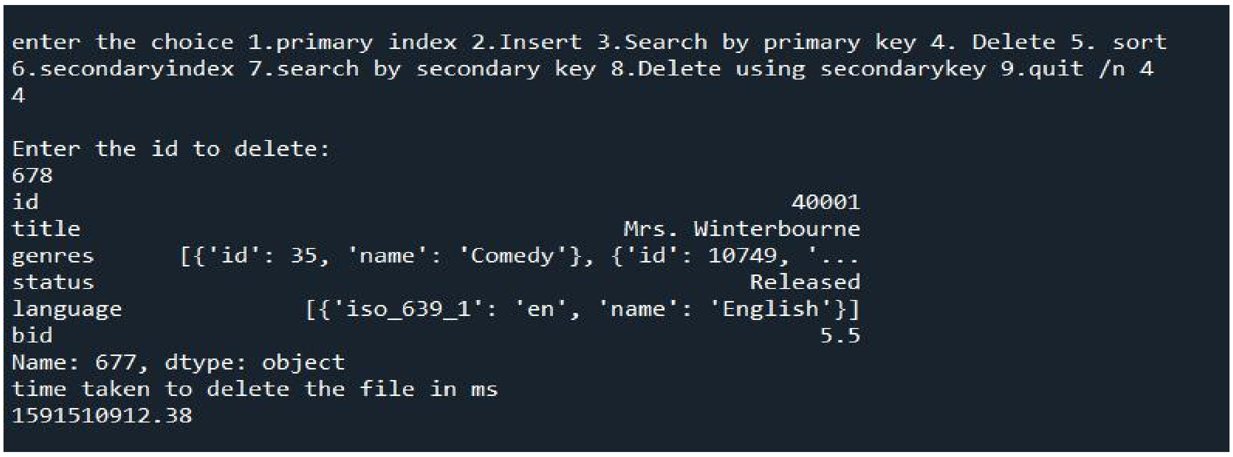


```
enter the choice 1.primary index 2.Insert 3.Search by primary key 4. Delete 5. sort
6.secondaryindex 7.search by secondary key 8.Delete using secondarykey 9.quit /n3
3

Enter id to search : 345
id                                                      10731
title                                              The Client
genres      [{'id': 18, 'name': 'Drama'}, {'id': 53, 'name...
status                                               Released
language             [{'iso_639_1': 'en', 'name': 'English'}]
bid                                                       6.4
Name: 345, dtype: object
time taken to search the file in ms
1591510899.59
```

By selecting 3 you can perform searching.

From the above fig, the 345 is searched using its primary index value.

4.1.2 Deleting A Record Using Primary key

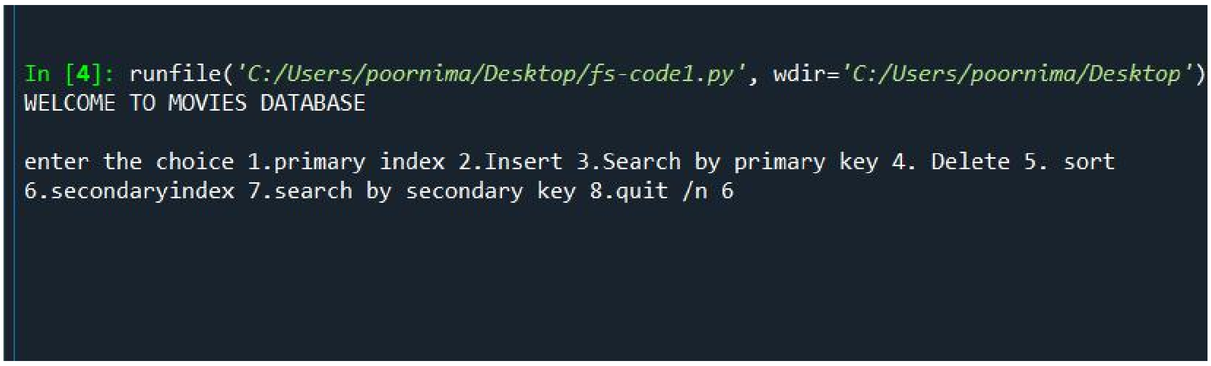


```
enter the choice 1.primary index 2.Insert 3.Search by primary key 4. Delete 5. sort
6.secondaryindex 7.search by secondary key 8.Delete using secondarykey 9.quit /n 4
4

Enter the id to delete:
678
id                                                      40001
title                                      Mrs. Winterbourne
genres      [{'id': 35, 'name': 'Comedy'}, {'id': 10749, '...
status                                               Released
language             [{'iso_639_1': 'en', 'name': 'English'}]
bid                                                       5.5
Name: 677, dtype: object
time taken to delete the file in ms
1591510912.38
```

By selecting option 4 you can perform deletion operation.

From the above fig, the id 678 is deleted in the file by referring to its index values entered by the user.

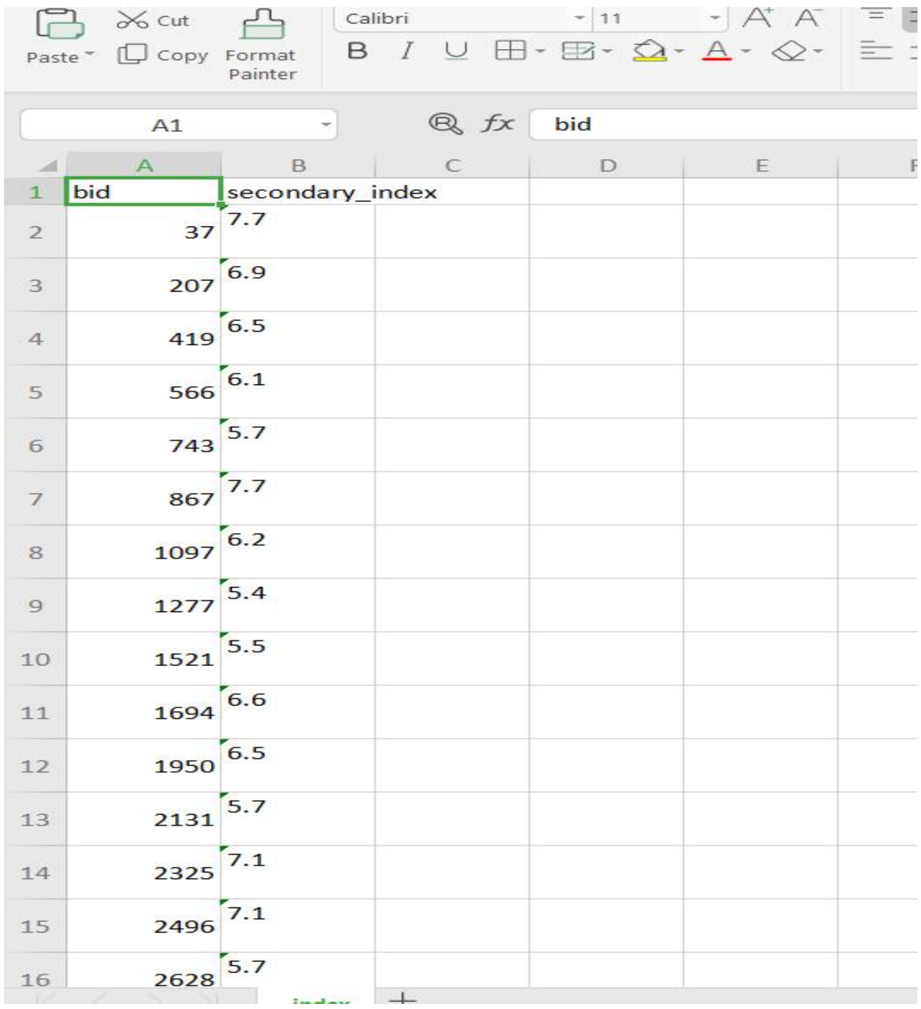
4.2 Secondary Indexing



```
In [4]: runfile('C:/Users/poornima/Desktop/fs-code1.py', wdir='C:/Users/poornima/Desktop')
WELCOME TO MOVIES DATABASE

enter the choice 1.primary index 2.Insert 3.Search by primary key 4. Delete 5. sort
6.secondaryindex 7.search by secondary key 8.quit /n 6
```

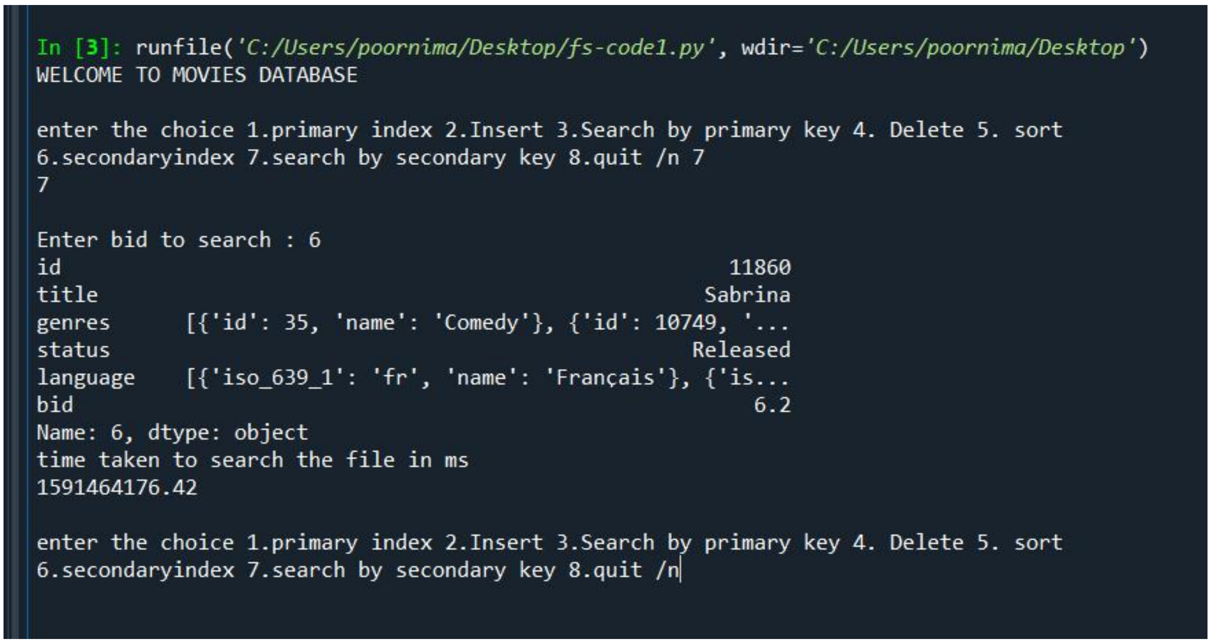
By selecting option 6 you can perform secondary indexing.

Output :



bid	secondary_index
37	7.7
207	6.9
419	6.5
566	6.1
743	5.7
867	7.7
1097	6.2
1277	5.4
1521	5.5
1694	6.6
1950	6.5
2131	5.7
2325	7.1
2496	7.1
2628	5.7

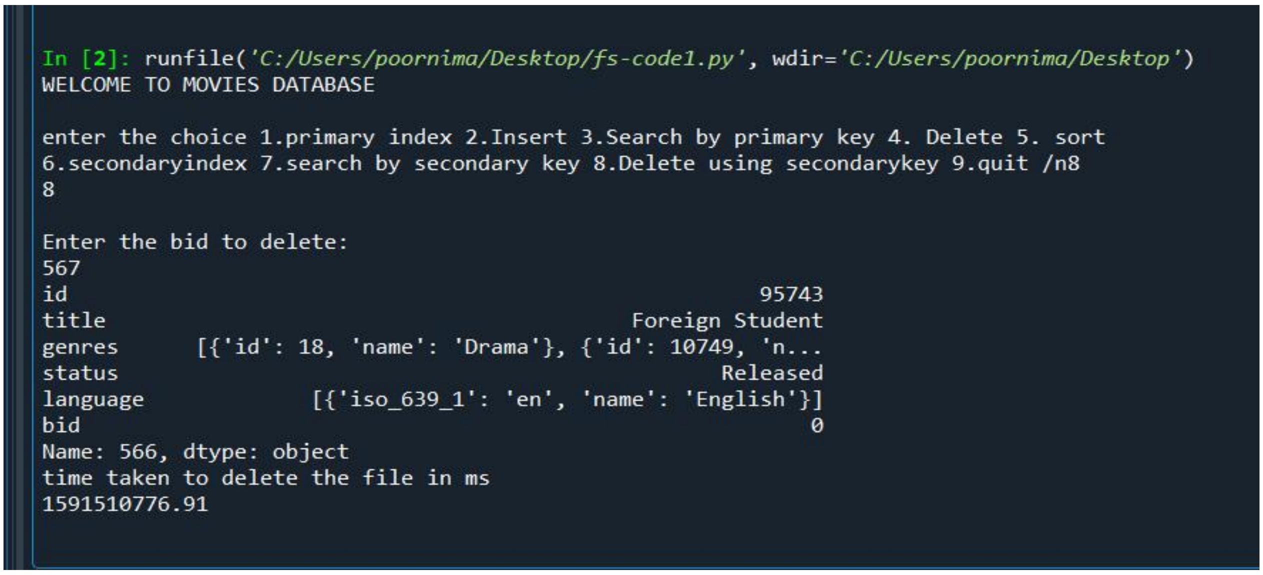
4.2.1 Searching Using Secondary Key



```
In [3]: runfile('C:/Users/poornima/Desktop/fs-code1.py', wdir='C:/Users/poornima/Desktop')
WELCOME TO MOVIES DATABASE

enter the choice 1.primary index 2.Insert 3.Search by primary key 4. Delete 5. sort
6.secondaryindex 7.search by secondary key 8.quit /n 7
7

Enter bid to search : 6
id                                                        11860
title                                                   Sabrina
genres      [{'id': 35, 'name': 'Comedy'}, {'id': 10749, '...
status                                                 Released
language    [{'iso_639_1': 'fr', 'name': 'Français'}, {'is...
bid                                                         6.2
Name: 6, dtype: object
time taken to search the file in ms
1591464176.42

enter the choice 1.primary index 2.Insert 3.Search by primary key 4. Delete 5. sort
6.secondaryindex 7.search by secondary key 8.quit /n
```

By selecting option 7 you can perform searching using secondary key.

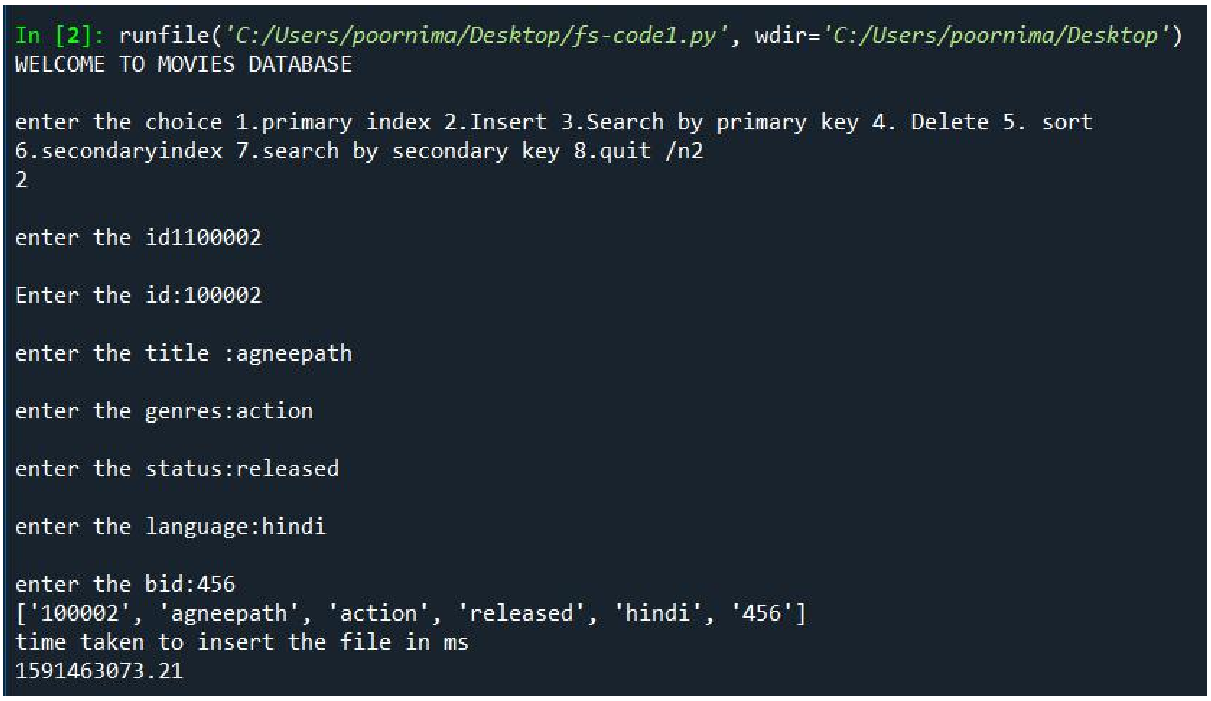
4.2.2 Deleting A Record Using Secondary Key



```
In [2]: runfile('C:/Users/poornima/Desktop/fs-code1.py', wdir='C:/Users/poornima/Desktop')
WELCOME TO MOVIES DATABASE

enter the choice 1.primary index 2.Insert 3.Search by primary key 4. Delete 5. sort
6.secondaryindex 7.search by secondary key 8.Delete using secondarykey 9.quit /n8
8

Enter the bid to delete:
567
id                                                        95743
title                                           Foreign Student
genres      [{'id': 18, 'name': 'Drama'}, {'id': 10749, 'n...
status                                                 Released
language                [{'iso_639_1': 'en', 'name': 'English'}]
bid                                                           0
Name: 566, dtype: object
time taken to delete the file in ms
1591510776.91
```

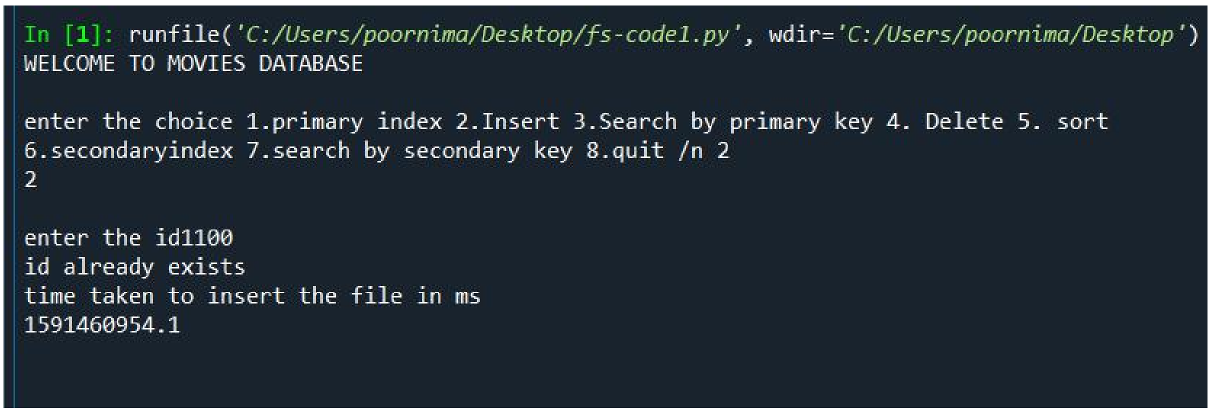
By selecting option 8 you can perform deletion using secondary key.

4.3 Insertion



```
In [2]: runfile('C:/Users/poornima/Desktop/fs-code1.py', wdir='C:/Users/poornima/Desktop')
WELCOME TO MOVIES DATABASE

enter the choice 1.primary index 2.Insert 3.Search by primary key 4. Delete 5. sort
6.secondaryindex 7.search by secondary key 8.quit /n2
2

enter the id1100002

Enter the id:100002

enter the title :agneepath

enter the genres:action

enter the status:released

enter the language:hindi

enter the bid:456
['100002', 'agneepath', 'action', 'released', 'hindi', '456']
time taken to insert the file in ms
1591463073.21
```

Enter 2 to perform insertion using primary key



```
In [1]: runfile('C:/Users/poornima/Desktop/fs-code1.py', wdir='C:/Users/poornima/Desktop')
WELCOME TO MOVIES DATABASE

enter the choice 1.primary index 2.Insert 3.Search by primary key 4. Delete 5. sort
6.secondaryindex 7.search by secondary key 8.quit /n 2
2

enter the id1100
id already exists
time taken to insert the file in ms
1591460954.1
```

From the above fig, if the id already exists in the file then it displays that id already exists does not take that input.

CONCLUSION

We have successfully implemented indexing which helps us in administrating the data used for managing the tasks performed.

View tables are used to display all the components at once so that user can see all the components of a particular type at once. One can just select the component and modify and remove the component.

We have successfully used various functionalities of python and created the File structures.

Features:

1. Clean separation of various components to facilitate easy modification and revision.

2. All the data is maintained in a separate file to facilitate easy modification

3. All the data required for different operations is kept in a separate file.

4. Quick and easy saving and loading of database file.

REFERENCES

The information about Indexing was gathered by referring to the following sites**:**

- Github(github.com)
- Stackoverflow(stackoverflow.com)
- GeeksforGeeks(GeeksforGeeks.com)
- Tutorialspoint(tutorialspoint.com)
- Youtube(Youtube.com)