Visvesvaraya Technological University
Belagavi-590 018, Karnataka

A Mini Project Report on

"INDEXING ON PRODUCT REVIEWS 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

SNEHAPRIYA B [1JT17IS042]
SHIJI K SHRIDHAR [1JIT17IS039]

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

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 SNEHAPRIYA B [1JT17IS042]** & **SHIJI K SHRIDHAR [1JT17IS039]** 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 **2018-2019**. 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
Guide, Asst. Professor
Dept. Of ISE

Dr. Harshvardhan Tiwari
Associate Professor and HOD
Dept. Of ISE

External Viva Examiner

1.
2.

Signature with Date :

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.

SNEHAPRIYA B [1JT17IS042]
SHIJI K SHRIDHAR [1JIT17IS039]

ABSTRACT

In this mini project we have created one application which is easy to access and user friendly. For this application is developed using simple java code. The project title is “Indexing” which is used in the application and for distinguishing the type of file structure to be used.

The work provides a simple and attractive and it creates the simple index and simplify the work as well as it reduces the time taken and t takes the same time to search the record in the first and also the record in the last.

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 liner list. Simple indexing can be useful when the entire index can be held in memory.

Changes (additions and searching) require both the index and the data file to be changes. Updates affect the index if the key field is changed, or if the record is moved. An update which moves a record order is determined by the order in which they are entered. The physical order of records in the file may not be the dame as order of entry, because of record deletions and space reuse.

The index should be read into memory when the data file is opened.

Table of Contents

CHAPTER 1
INTRODUCTION

INTRODUCTION

Introduction to File Structure

A database is simply on organized collection of related data, typically stored on disk, and accessible by possibly many concurrent users. Databases are generally separated into application areas. For example, one database may contain Human Resource (employee and payroll) data; another may contain sales data; another may contain accounting data; and so on. Databases are managed by a DBMS.

In computing, a file system or filesystem controls how data is stored and retrieved. Without a file system, information placed in a storage medium would be one large body of data with noway to tell where one piece of information stops and the next begins. By separating the data into pieces and giving each piece a name, the information is easily isolated and identified. Taking its name from the way paper-based information systems are named, each groups of data is called a “file”. The structure and logic rules used to manage the groups of information and their names is called a “file system”.

There are many different kinds of file systems. Each one has different structure and logic, properties of speed, flexibility, security, size and more. Some file systems have been designed to be used for specific applications. For example, the ISO 9660 file system is designed specifically for optical discs.

File systems can be used on numerous different types of storage devices that use different kinds of media. The most common storage device in use today is a hard disk drive. Other kinds of media that are used include flash memory, magnetic tapes, and optical discs. In some cases, such as with tmpfs, the computers’s main memory (random-access memory, RAM) is used to create a temporary file system for short-term use.

Some file systems are used on local data storage devices; others provide file access

via a network protocol(for example, NFS, SMB, or 9P clients). Some file systems are "virtual". Meaning that the supplied "files" (called virtual files) are computed on request (e.g. procfs) or are merely a mapping into a different file system used as a blacking store. The file system manages access to both the content of files and the metadata about those files. It is responsible for arranging storage space; reliability, efficiency, and tuning with regard to the physical storage medium are important design considerations.

Introduction to JAVA

- Like any programming language, Java language has its own structure, syntax rules, and programming paradigm. The Java language's programming paradigm is based on concept of OOP, which the language's features support.
- The Java language is a C-language derivative, so its syntax rules look much like C's. For example, code blocks are modularized into methods and delimited by braces and variables are declared before they are used.
- Structurally, Java language starts with packages. A packages is the Java language's namespace mechanism. Within packages are classes, and within classes are methods variables, constants, and more. You learn about the parts of the java language in this tutorial.

Indexing

What is an Index?

Index: A structure containing a set of entries, each consisting of a key field and a reference field, which is used to locate records in a data file.

Key Field: The part of an index which contains keys.

Reference Field: The part of an index which contains information to locate records.

- An index imposes order on a file without rearranging the file.
- Indexing works by indirection.

A Simple Index for Entry-Sequenced Files

An index in which the entries are a key ordered linear list.

- Simple indexing can be useful when the entire index can be held in memory.
- Changes (additions and deletions) require both the index and the data file to be changed.
- Updates affect the index if the key field is changed, or if the record is moved.

- An update which moves a record can be handled as a deletion followed by an addition.

Object Oriented Support for Indexed, Entry Sequenced Files

A file in which the record order is determined by the order in which they are entered

- The physical order of records in the file may not be the same as order of entry, because of record deletions and space reuse.
- The index should be read into memory when the data file is opened.
  Indexes that are too large to hold in memory
- Searching of a simple index on disk takes too much time.
- Maintaining a simple index on disk in sorted order takes too much time.
- Tree structured indexes such as B-tress are a scalable alternative to simple indexes.
- Hashed organization is an alternative to indexing when only a primary index is needed.
  Indexing to Provide Access by Multiple Keys
  A search key other than the primary key.
  An index built on a secondary key.
- Secondary indexes can be built on any field of the data file, or on combinations of fields.
- Secondary indexes will typically have multiple locations for a single key.
- Changes to the data may now affect multiple indexes.
- The reference field of a secondary index can be a direct reference to the location of the entry in the data file.
- The references fields of a secondary index can also be an indirect reference to the location of the entry in the data file, through the primary key.
- Indirect secondary key references simplify updating of the file set.
- Indirect secondary key references increases access time.
  Retrieval Using Combinations of Secondary Keys
- The search for records by multiple keys can be done on multiple index, with the combination of index entries defining the records matching the key combination.
- If two keys are to be combined, a list of entries from each key index is retrieved.
- For an “or” combination of keys, the lists are merged.
- I.e., only entries found in entire list matches the search.

CHAPTER 2
DESIGN

DESIGN

Algorithm to build a Primary Index and Secondary Index:

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

Indexes are created using some columns in a file:

- 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 block where that particular key value can be forced.

Clustering index is defined on an ordered data file. The data file is ordered on a non-key field. In some cases, the index is created on non-primary key columns which may not be unique for each record. In such cases, in order to identify the records faster, we will group two or more columns together to get the unique values and create index out of them. This method is known as clustering index. Basically, records with similar characteristics are grouped together and indexes are created for these groups.

The data is sorted according to the search key. It includes sequential file organization. The primary key in data frame is used to create the index. As primary keys are unique and are stored in sorted manner, the performance of searching operation is quite efficient.

Algorithm:

```
Int insert Sorted (int arr [], int n, int key, int capacity)
If (n >= capacity)
Return n;
Int I;
For (1=n-1; (I >=0 && arr[i] >key); i--)
arr [i+1] = arr[i]
arr [i+1] = key;
Return (n+1);
```

Algorithm to Sort the Index based on Primary key:

The Insertion sort, although still o (n2) o (n2), works in a slightly different way. It always maintains a sorted a sub_ list in the lower positions of list. Each new item is then "inserted" Back into the pervious sub -list such that the sorted sub-list is one item larger. We begin by assuming that a list with one item (position 00) is already sorted. On each pass, one for each item 1 through n-1n-1, the current item is checked against those in the already sorted sub-list. As we look back into the already sorted sub -list, we shift those items that are greater to the right when we reach a smaller item or the end of the sub -list, the current item can be inserted.

The implementation of insertion Sort shows that there we are again n-1n-1 passes to sort n items. The iteration starts at position 1 and moves through position n-1n-1, as these are the items the need to be inserted back into sorted sub-lists. Line 8 performs the shift operation that moves a value up one position in the list, making room behind it for the insertion. Remember that this not a complete exchange as was performed in the previous algorithms.
The, maximum number of comparisons for an insertions sort is the sum of the first n-1n-1 integers. Again this is o (n2) o (n2). However, in the best case, only one comparison needs to be done on each pass. This would be the case for an already sorted list .One note about shifting versus exchanging is also important. In general, a shift operations requires approximately a third of the processing work of an exchange since only one assignment is performed. In benchmark studies, insert ion sort will allow very good performance.

Algorithm for searching

When data items are sorted in a collection such as a list, we say that they are linear or sequential relationship. Each data items is stored in a position relative to the others. In java lists, these relative positions are the index values of the individual items. Since these index values are ordered, it os possible for us to visit them in sequence. This process gives rise to our first searching technique, the sequential technique, the SEQUENTIAL SEARCH.
The java implantation for this algorithm is function the needs the list and the items we are looking for and returns a Boolean values as to whether it is present. The Boolean variable found is initialized to False and is assigned the values= True f we discover he items in the list.

Algorithm:

```
SET Lo to 0
SET Hi to array length – 1
WHILE Lo < = Hi
        SET Mid to { Lo + Hi } / 2
        IF X < array [Mid] THEN
             SET Hi to Mid -1
        ELSE IF X > array[Mid] THEN
              SET Lo to  Mid +1
        ELSE
              RETURN Mid
        ENDIF
ENDWHILE
RETURN -1
```

CHAPTER 3
IMPLEMENTATION

IMPLMENTATION

Indexing Orientations

- If n entries have v possible orientations

```
t=0
for i=1 to n
t = t * v
t = t + or[i]
endfor
return t
```

- To extract the individual orientations again from t < vn-1, use the following code:

```
for i = n to 1
 or[i] = t mod v
 t = t / v
endfor
```

- Usually there is the constraint that the total twist is 0 modulo v, in other words the orientation of the last entry is dependent on the other n-1. In this case just do not encode the orientation of the last piece, which gives a number between 0 and vn-1-1:
- To extract the individual orientations again from t<vn-1-1, use the following code:

```
s = 0
for I = n-1 to 1
or[i] = t mod v
s = s – or[i]
if s < 0 then s = s + v
t = t / v
endfor
or[n] = s
```

- If n entries can be permuted amongst themselves then their permutation can be encoded in a number between 0 and n!-1. Use a fixed numbering for both the entries and the positions: the n entries positions are numbered from 1 to n, and the entries are also from 1 to n

```
t = 0;
for I = 1 to n-1
t = t * (n-i+1)
for j=i+1 to n
if pm[i]>pm[j] then t=t+1
end for
end for
retrun t
```

- Note that if all entries are in position then it is encoded as 0. To extract the permutation again from a number t<n! use this:

```
for i=n-1 to 1
pm[i]=1 +(tmod(n-i+1))
t=t/(n-i+1)
for j= i+1 to n
if pm[j]>=pm[i] then pm[j]=pm[i]+1
endfor
endfor
```

- Often there is the constraint that the permutation must have even parity. This means that the position of the last two entries is dependent on the other n-1. In this case just do not encode the position of the last two entries, which gives a number between 0 andn!2-1:

```
t=0;
for i=1 to n-2
t=t*(n-i+)
for j=i+1 to n
if pm[i] > pm[j] then t=t+
endfor
endfor
```

- To extract the even permutation again from a number t>n!/ use this

```
pm[n-1] =1
s=0
for i=n-2 to 1
pm[i]=1 + { t mod (n-i+1)}
s=s+pm[i]-1
t=t/{n-i+1}
for j=i+1 to n
if pm[j] >=pm[i] then pm[j] = pm[j] +1
end for
end for
```

```
if s mod 2=1 then swap pm[n],pm[n-1]
```

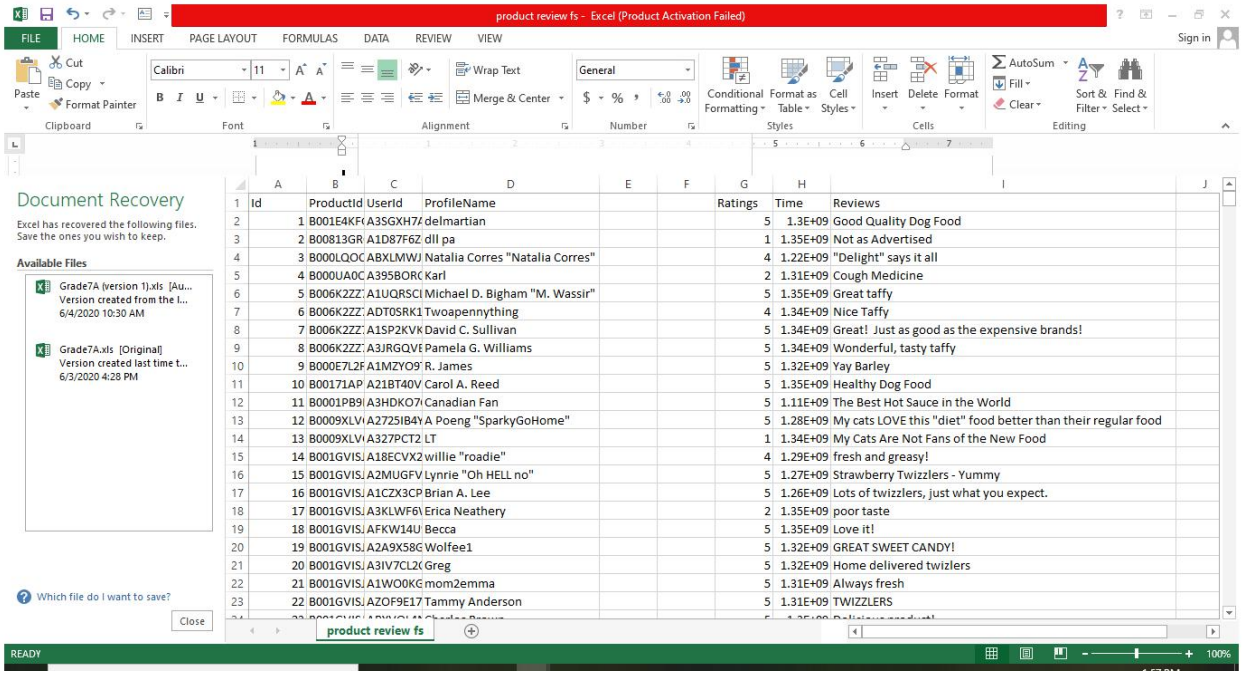
Indexing combinations

```
t=0,r=m
t=0
r=m
for i=n-1 to 0
if cm[i+1] =p then
t=t+ comb(i,r)
r=r-1
endif
next
return t
r=m
```

CHAPTER 4

RESULT AND SNAPSHOTS

RESULTS AND SNAPSHOTS



Id	ProductId	UserId	ProfileName	Ratings	Time	Reviews
1	B001E4KF(	A3SGXH7/	delmartian	5	1.3E+09	Good Quality Dog Food
2	B00813GR	A1D87F6Z	dll pa	1	1.35E+09	Not as Advertised
3	B000LQOC	ABXLMWJ	Natalia Corres "Natalia Corres"	4	1.22E+09	"Delight" says it all
4	B000UA0C	A395BORC	Karl	2	1.31E+09	Cough Medicine
5	B006K2ZZ	A1UQRSCI	Michael D. Bigham "M. Wassir"	5	1.35E+09	Great taffy
6	B006K2ZZ	ADT0SRK1	Twoapennything	4	1.34E+09	Nice Taffy
7	B006K2ZZ	A1SP2KVK	David C. Sullivan	5	1.34E+09	Great! Just as good as the expensive brands!
8	B006K2ZZ	A3JRGQVI	Pamela G. Williams	5	1.34E+09	Wonderful, tasty taffy
9	B000E7L2F	A1MZYO9	R. James	5	1.32E+09	Yay Barley
10	B00171AP	A21BT40V	Carol A. Reed	5	1.35E+09	Healthy Dog Food
11	B0001PB9	A3HDKO7(	Canadian Fan	5	1.11E+09	The Best Hot Sauce in the World
12	B0009XLV	A2725IB4Y	A Poeng "SparkyGoHome"	5	1.28E+09	My cats LOVE this "diet" food better than their regular food
13	B0009XLV	A327PCT2	LT	1	1.34E+09	My Cats Are Not Fans of the New Food
14	B001GVIS	A18ECVX2	willie "roadie"	4	1.29E+09	fresh and greasy!
15	B001GVIS	A2MUGFV	Lynrie "Oh HELL no"	5	1.27E+09	Strawberry Twizzlers - Yummy
16	B001GVIS	A1CZX3CP	Brian A. Lee	5	1.26E+09	Lots of twizzlers, just what you expect.
17	B001GVIS	A3KLWF6\	Erica Neathery	2	1.35E+09	poor taste
18	B001GVIS	AFKW14U	Becca	5	1.35E+09	Love it!
19	B001GVIS	A2A9X58(	Wolfee1	5	1.32E+09	GREAT SWEET CANDY!
20	B001GVIS	A3IV7CL2(	Greg	5	1.32E+09	Home delivered twizlers
21	B001GVIS	A1WO0KG	mom2emma	5	1.31E+09	Always fresh
22	B001GVIS	AZOF9E17	Tammy Anderson	5	1.31E+09	TWIZZLERS

Fig4.1: Dataset The above dataset consists of id, Product Id, User Id, Product Name, Ratings,Time and Reviews. This is the dataset which is further get indexed.

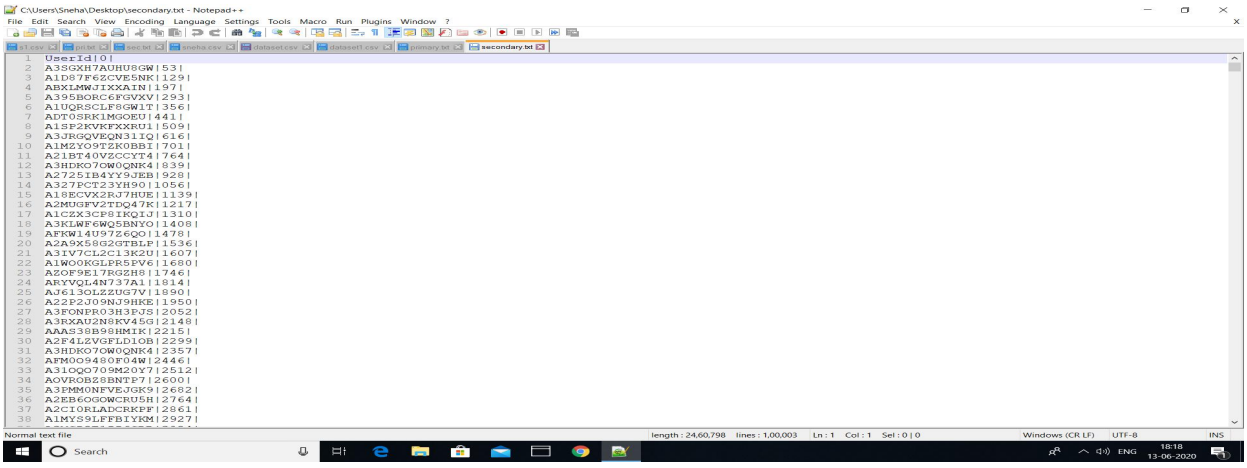


Fig 4.2 index build for product reviews dataset using primary key id

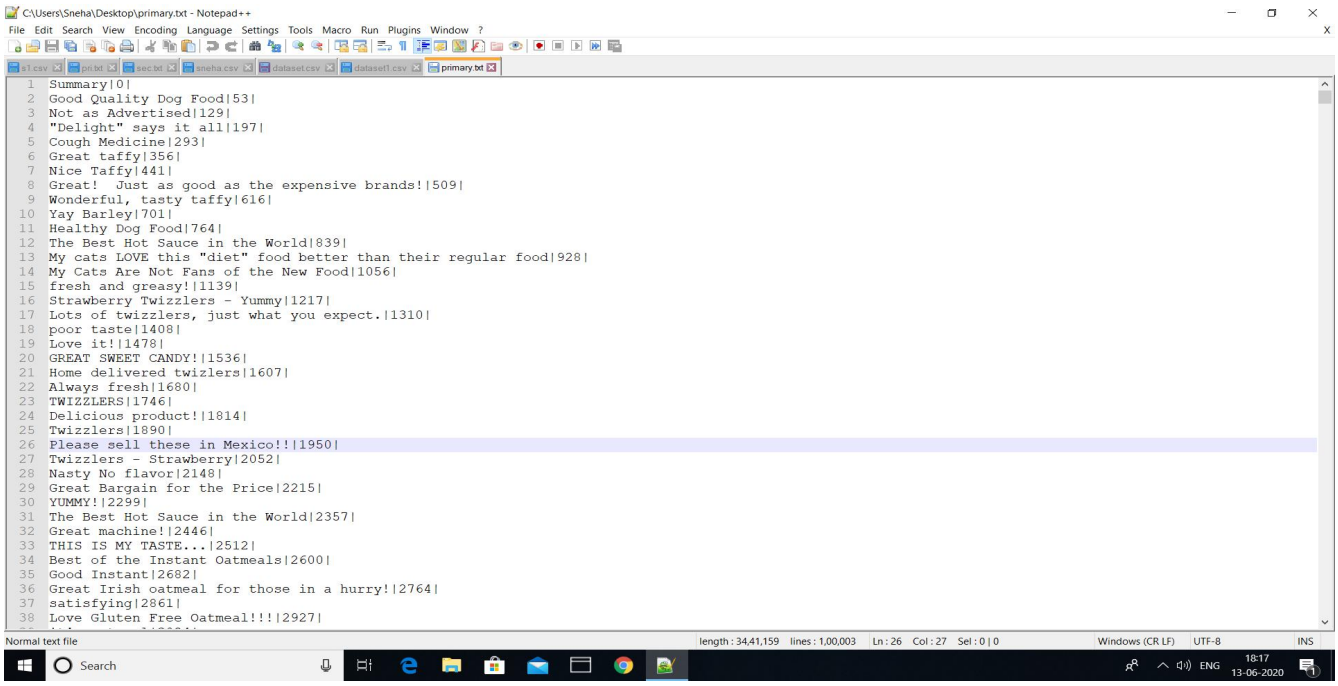


Fig 4.3 index build for product reviews dataset using secondary key

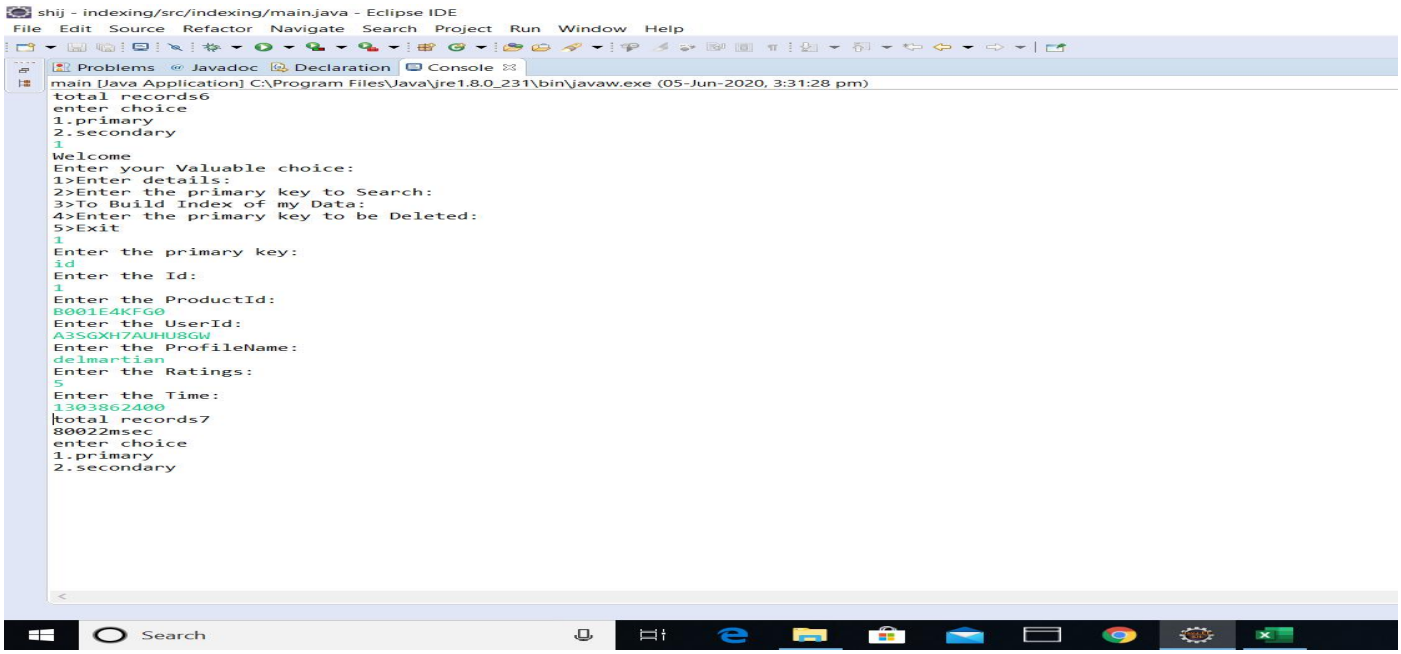


Fig4.4 Insertion of data

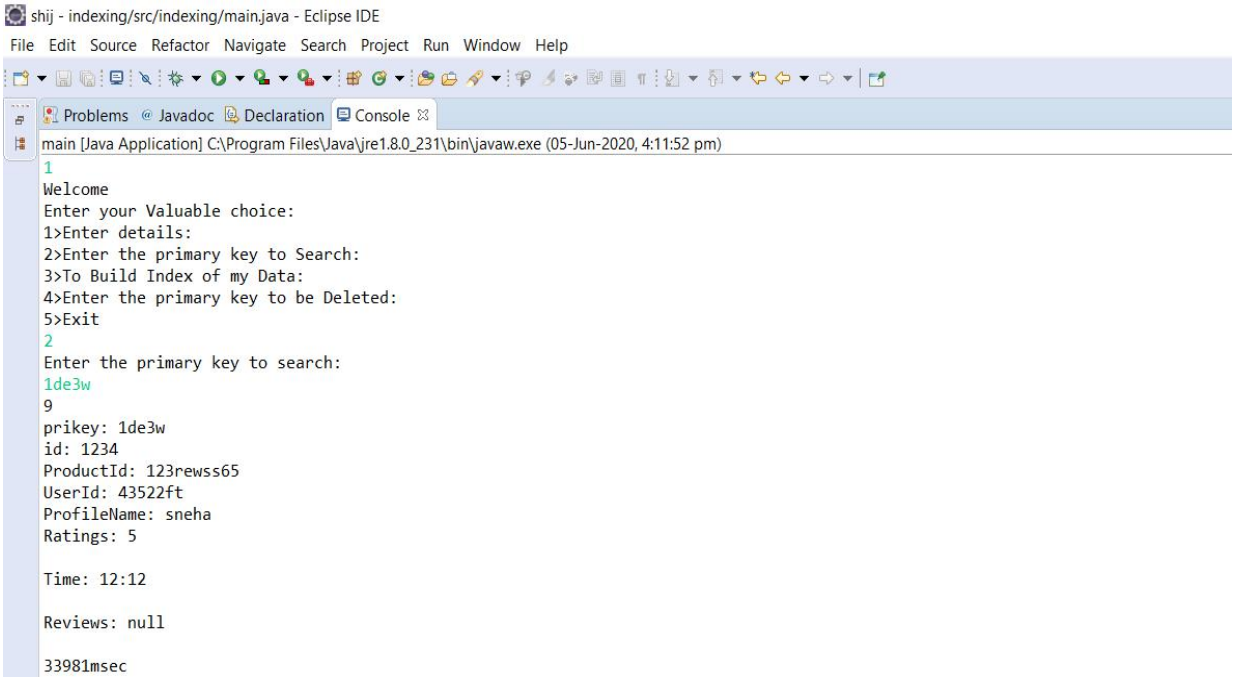


Fig 4.5 Searching of primary key in data

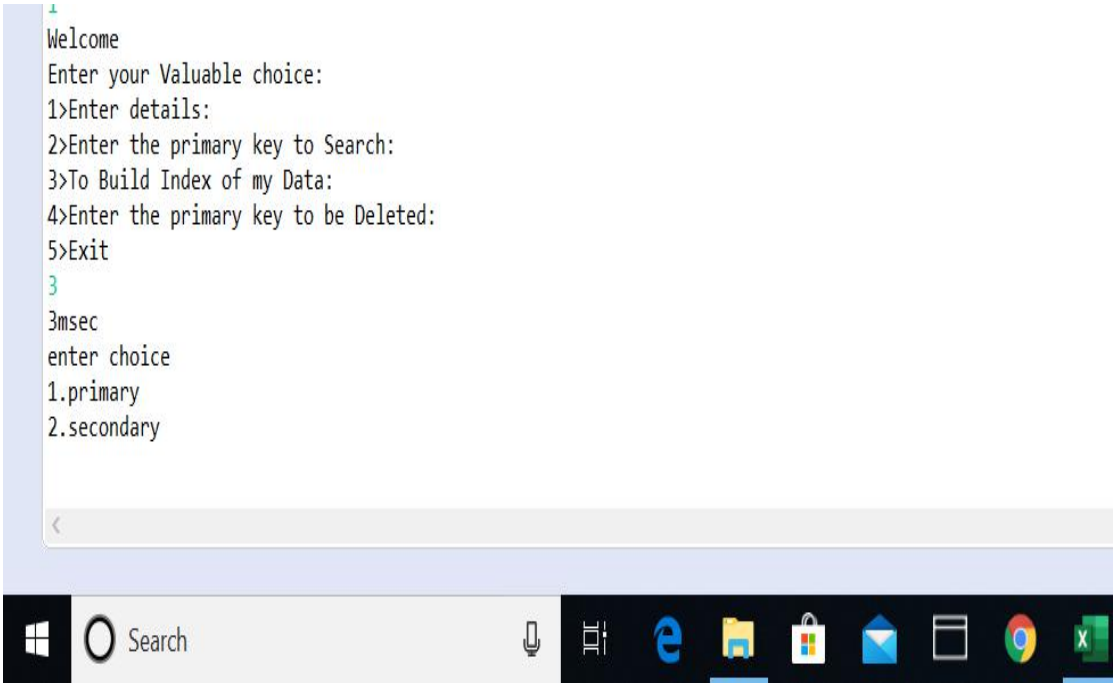
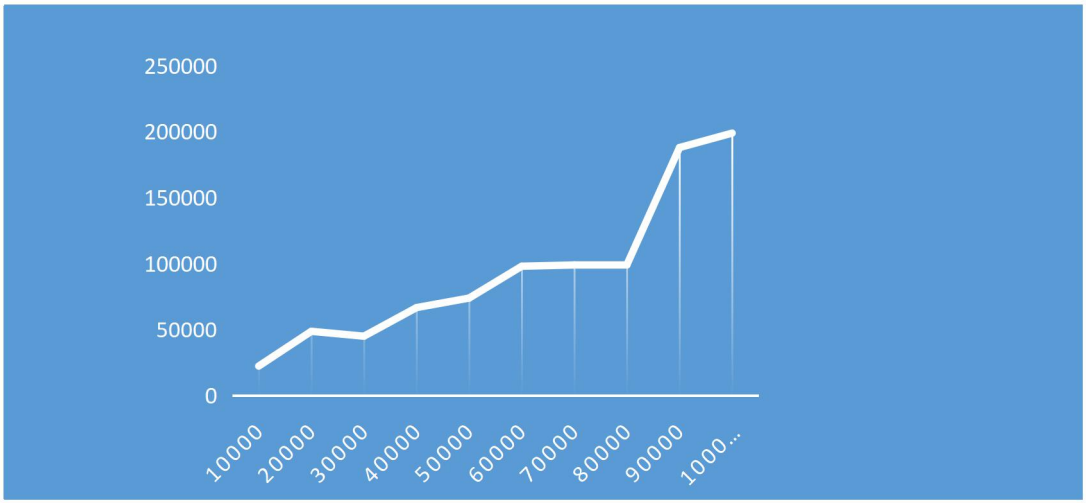


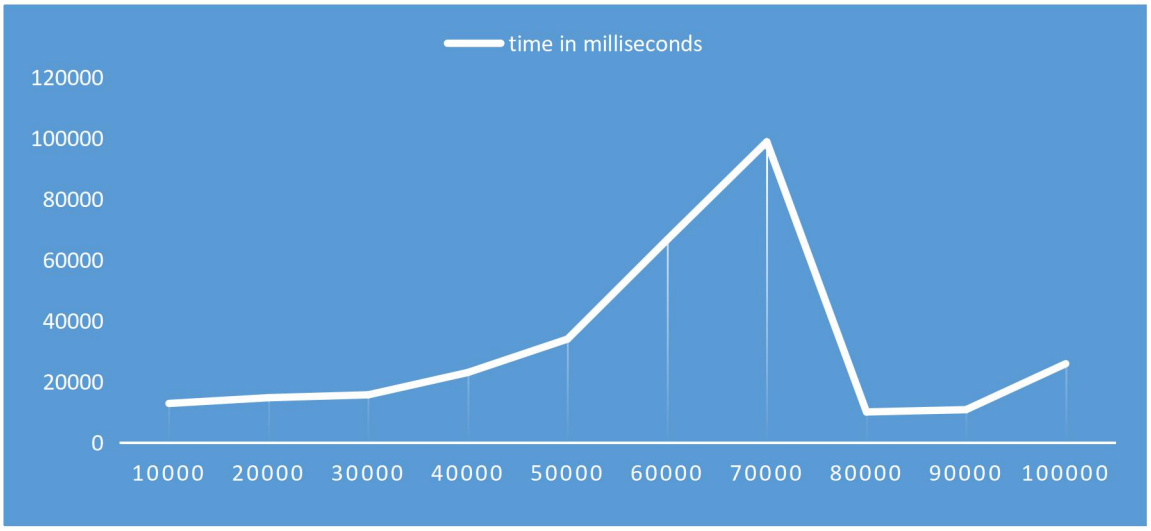
Fig 4.6 Building a index of a data

Time Analysis For Inserting the Record



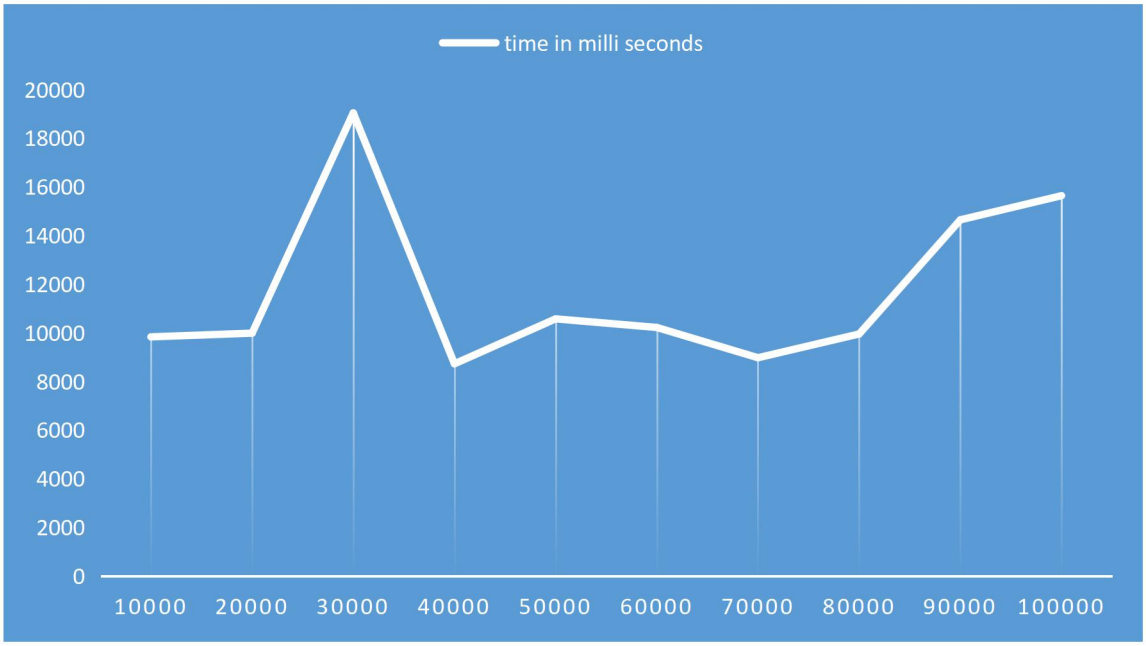
The time taken to insert a record into a data file, as the number of record increases the time also increases.

Time analysis for searching a primary key record



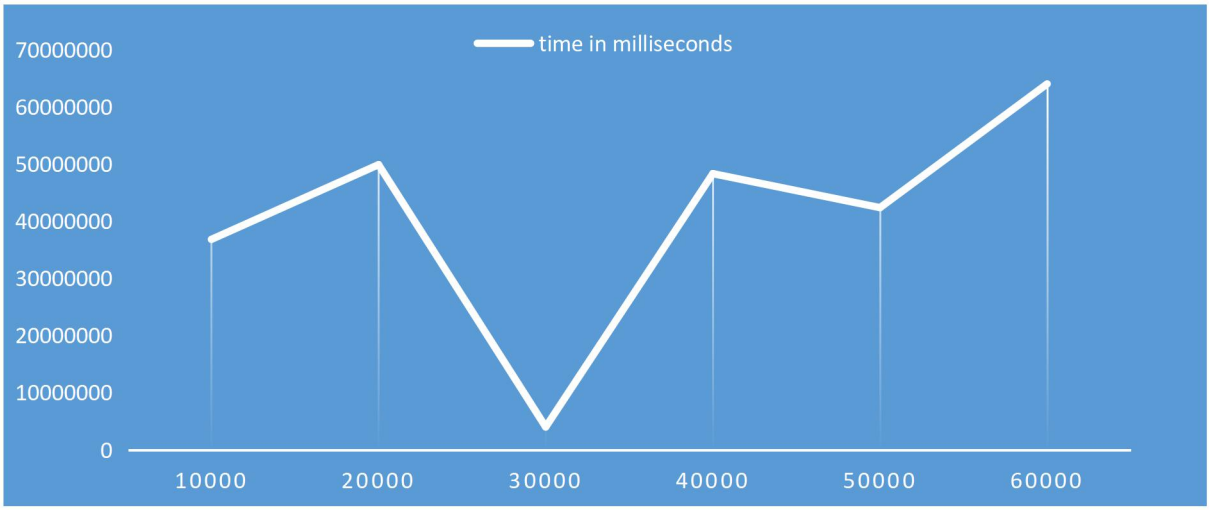
The time taken to search a primary key record in the data file, as the number of record increases, the time will be delayed to search a record.

Time Analysis for Deleting a primary key record



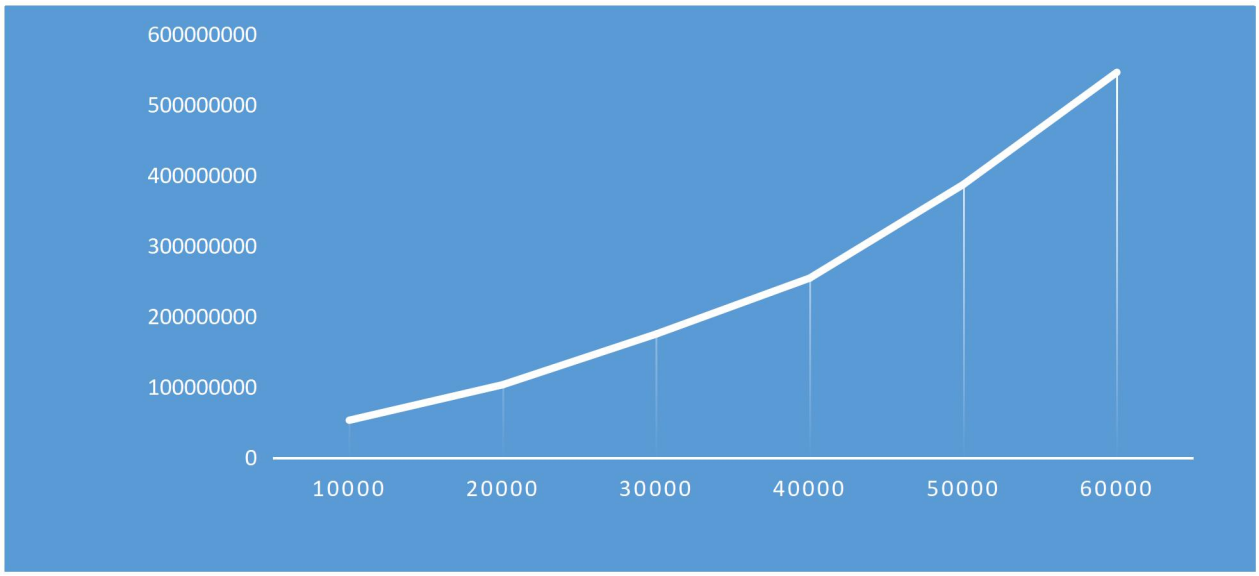
The time taken to delete from the data file, decreases as the number of record increases.

Time Analysis for searching a secondary key record



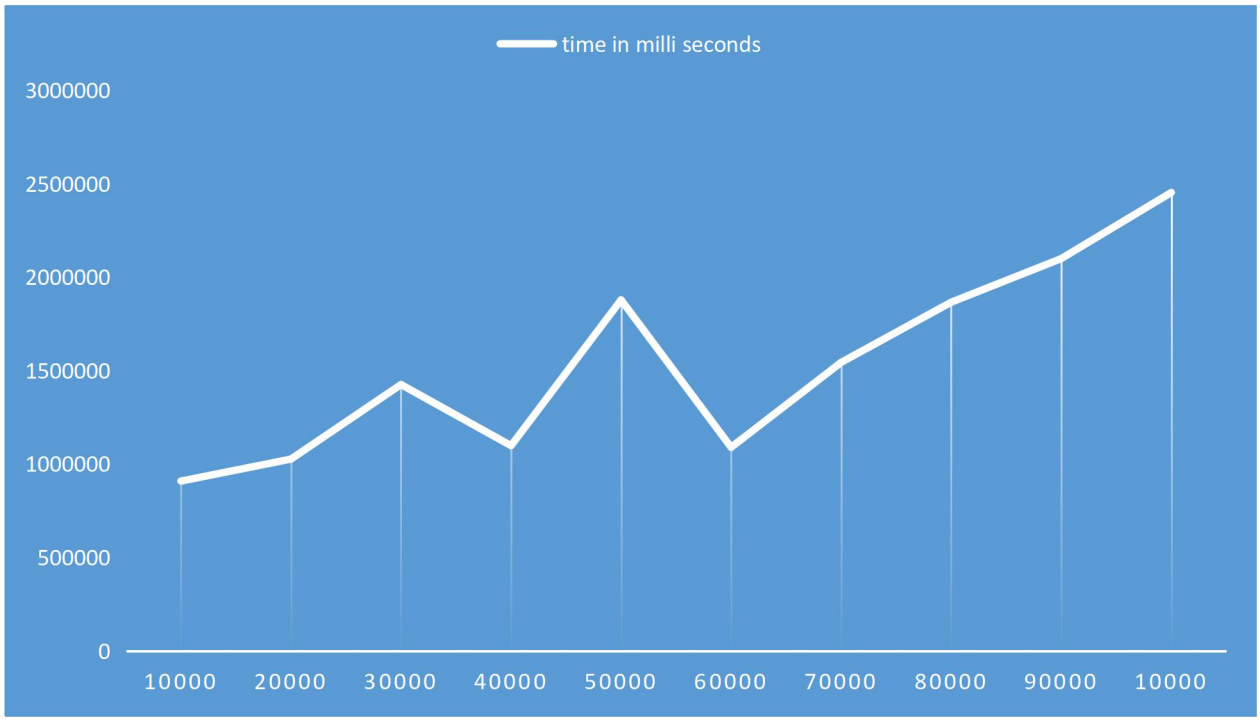
The time taken to search the record in a data file, as the number of record increases the time will delayed to search of record.

Time Analysis for inserting a secondary key record



The time taken to insert a record into a data file, as the number of record increases the time also increases.

Time Analysis for Deleting a secondary key record



The time taken to delete from the data file, decreases as the number of record increases.

CONCLUSION

We have successfully used various functionalities of JAVA and created the file structures.

- Indexes form an important part of designing, creating and testing information.
- Users search in a hurry for information to help them and give up after two or three tries.
- An index can point the way in harmony with user expectations or not.
- Indexing is an interactive analysis and creative process throughput the entire documentation.

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.

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.

REFERENCES

- Complete Java Reference Book
- Algorithms by Robert sedgewick & Kevin Wayne
- Core Java Volume 1-Fundamentals
- www.github.comMicheal J.Folk, Bill Zoellick, Greg Riccardi File Structure An Object Oriented