

**Visvesvaraya Technological  
University  
Belagavi-590018, Karnataka**



A Mini Project Report on

**“Simple Indexing on Human-Resources Records”**

**Submitted in partial fulfillment of the requirement for the  
File Structures Laboratory with Mini Project**

**[17ISL68]**

**Bachelor of Engineering in  
Information Science and Engineering**

**Submitted by**

**Krishna.R [1JT17IS016]**

Under the support and guidance of

**Mr.Vadiraja A**

Asst.Prof, Dept. 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 entitled “**Simple Indexing on Human-Resources Records**” carried out by **KRISHNA R [1JT17IS016]** bonafide student of Jyothy Institute of Technology, in partial fulfillment for the award of **Bachelor of Engineering in Information Science and Engineering** department of **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**

Asst. Professor  
Dept. Of ISE

**Dr.Harshvardhan Tiwari**

Assoc.Professor and HoD  
Dept. Of ISE

External Viva Examiner

- 1.
- 2.

Signature with Date :

# ACKNOWLEDGEMENT

Firstly, I am very grateful to this esteemed institution **Jyothy Institute of Technology** for providing me an opportunity to complete my project.

I express my sincere thanks to our Principal **Dr. Gopalakrishna K** for providing me with adequate facilities to undertake this project.

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

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

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

**Krishna.R [1JT17IS016]**

# **ABSTRACT**

This project titled “Simple Indexing on Human-Resources Records” has been done using Eclipse IDE with the platform Windows and language Java. The database used for the project is ‘Human Resources’ records. The project mainly focuses on building the index for the records which is fed in CSV format file, then various operations with a menu choice is displayed to the end user, such as Insert, Search, Delete and Modify. For the purpose of searching efficiently, binary search algorithm is being used. The inserted record will be initially packed and then will be put in the record file. The user can also Unpack all the records in the file which will be displayed.

The index files generated comprises of a key value and its respective position in the record file. This position will be used for various operations.

So, Indexing is a ‘way to optimize the performance of file access by minimizing the number of disk accesses required to process the required data’.

SL No	Description	Page No
	<b>Chapter 1</b>	
1	<b>INTRODUCTION</b>	1-3
	1.1 Introduction to File Structures	1
	1.2 Introduction to File System	2
	1.3 Introduction to Simple Indexing	3
	<b>Chapter 2</b>	
2	<b>DESIGN</b>	4-5
	2.1 Scope and importance of work	4
	2.2 Classification of Requirements	5
	2.3 System Analysis	5
	<b>Chapter 3</b>	
3	<b>IMPLEMENTATION</b>	6-9
	<b>Chapter 4</b>	
4	<b>RESULTS AND SNAPSHOTS</b>	10-15
	<b>CONCLUSIONS,FUTURE ENHANCEMENTS &amp; REFERENCES</b>	16-17

# ***CHAPTER 1***

## ***INTRODUCTION***

# INTRODUCTION

## 1.1 Introduction to File Structures

In simple terms, a file is a collection of data stored on mass storage (e.g., disk or tape). But there is one important distinction that must be made at the outset when discussing file structures. And that is the difference between the logical and physical organization of the data.

On the whole a file structure will specify the logical structure of the data, that is the relationships that will exist between data items independently of the way in which these relationships may actually be realized within any computer. It is this logical aspect that we will concentrate on. The physical organization is much more concerned with optimizing the use of the storage medium when a particular logical structure is stored on, or in it. Typically for every unit of physical store there will be a number of units of the logical structure (probably records) to be stored in it.

For example, if we were to store a tree structure on a magnetic disk, the physical organization would be concerned with the best way of packing the nodes of the tree on the disk given the access characteristics of the disk.

Like all subjects in computer science the terminology of file structures has evolved higgledy-piggledy without much concern for consistency, ambiguity, or whether it was possible to make the kind of distinctions that were important.

It was only much later that the need for a well-defined, unambiguous language to describe file structures became apparent. In particular, there arose a need to communicate ideas about file structures without getting bogged down by hardware considerations.

## 1.2 Introduction to File Systems

In computing, a file system or file system 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 no way 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 group 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.

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 computer'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. Some file systems are “virtual”. Meaning that the supplied “files” are computed on request or are merely a mapping into a different file system used as a backing store. The file system manages access to both the content of files and the meta data 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 .



## 1.3 Introduction to Simple Indexing

We know that data is stored in the form of records. Every record has a key field, which helps it to be recognized uniquely. Indexing is a data structure technique to efficiently retrieve records from the database files based on some attributes on which the indexing has been done. Indexing in database systems is similar to what we see in books.

Indexing not only helps in querying, but can also be used in any algorithm being built to reduce the time taken by that algorithm.

Simple indexes use simple arrays. An index lets us impose order on a file without rearranging the file. Indexes provide multiple access paths to a file—multiple indexes (like library catalog providing search for author, book and title) An index can provide keyed access to variable-length record files.

Index is sorted (main memory). Records appear in file in the order they entered. An index is defined on one or more columns, called key columns. The key columns (also referred to as the index key) can be likened to the terms listed in a book index. They are the values that the index will be used to search for. As with the index found at the back of a text book, the index is sorted by the key columns.

Primary index is defined on an ordered data file. The data file is ordered on a key field. The key field is generally the primary key of the relation.

Secondary index may be generated from a field which is a candidate key and has a unique value in every record, or a non-key with duplicate values. Many operations can be performed efficiently like search, delete, modify after building the index.

# ***CHAPTER 2***

## ***DESIGN***

## 2.1 Scope and importance of work

When a large number of files are maintained, the necessity of maintaining the index is increased. Indexing increases the utility of filing by providing an easy reference to the files. The very purpose of maintaining the index is that it is easy and quicker to find location of the files.

The advantage of using index lies in the fact is that index makes search operation perform very fast. So, adding an index to a column will allow you to query based on a column faster.

Suppose a table has a several rows of data, each row is 20 bytes wide. If you want to search for the record number 100, the management system must thoroughly read each and every row and after reading  $99 \times 20 = 1980$  bytes it will find record number 100. If we have a index, the management system starts to search for record number 100 not from the table, but from the index. The index, containing only two columns, may be just 4 bytes wide in each of its rows. After reading only  $99 \times 4 = 396$  bytes of data from the index the management system finds an entry for record number 100.

The importance of indexing can be explained with the help of the following points:

### 1. Systematic arrangements of files:

Indexing helps to develop a modern scientific method of filing because indexing is not possible if the documents are not arranged in a systematic manner

### 2. Prompt location of files:

Indexing provides signs, symbol and guide to specific file in drawers. A person who needs a document in a file can make the use of an index to locate it.

### 3. Saves time and effort:

Indexing helps to locate the position of the specific document in files at a short period of time. It helps to make a quick decision by providing necessary information stored in files.

4. Develop efficiency:

Indexing facilitates the systematic arrangement of files and document. It saves the time required to search the information and space required to protect valuable document and information.

5. Reduce expenses:

The systematic arrangement and preservation of file reduce the overhead expenses of the office. The systematic arrangement reduces the space requirement to store the document.

## 2.2 Classification of Requirements

1. JDK - Java Development Kit.
2. Any Text editors (IDE preferred - Eclipse, Net-beans etc.)
3. Any Windows/Mac/Linux distributions

## 2.3 System Analysis

When the program is executed, two index files (primary and secondary index files) are created. Then the user is presented with the menu choice comprising of four operations namely:

**Insert, Search, Modify** and **Delete** which can be performed on both primary and secondary indexes respectively.

The modify operation handles the records in such a way that, if the length of the modified record is greater than the length of the existing record then the modified record will be appended at the end of the record file. Or else it will be modified at the same position.

# ***CHAPTER 3***

## ***IMPLEMENTATION***

## IMPLEMENTATION

The implementation of the entire process has been briefly described in the following stages below.

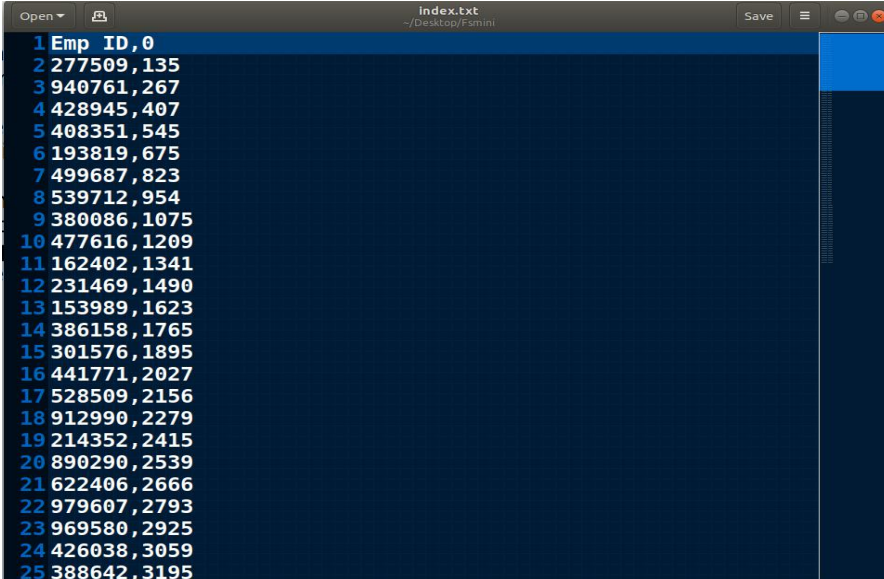
Simple Indexing on Human Resources records creates two index files namely, primary-index file and secondary-index file.

### BUILDING INDEX FILES

#### Primary Index File (index.txt)

The primary index file comprises of the primary key and it's starting position in the record file. Usually the first column in the record file will be unique and it will be considered as primary key column. In the Human Resources records data-set used in this project "Emp\_ID" is the first column which is unique, so it is being used as primary key for the purpose of running various operations based on this primary key like searching the record based on primary index, deleting the record based on primary index and also modifying the records based on this primary index file.

The snapshot of the primary-index file is shown in the Fig-3.1.

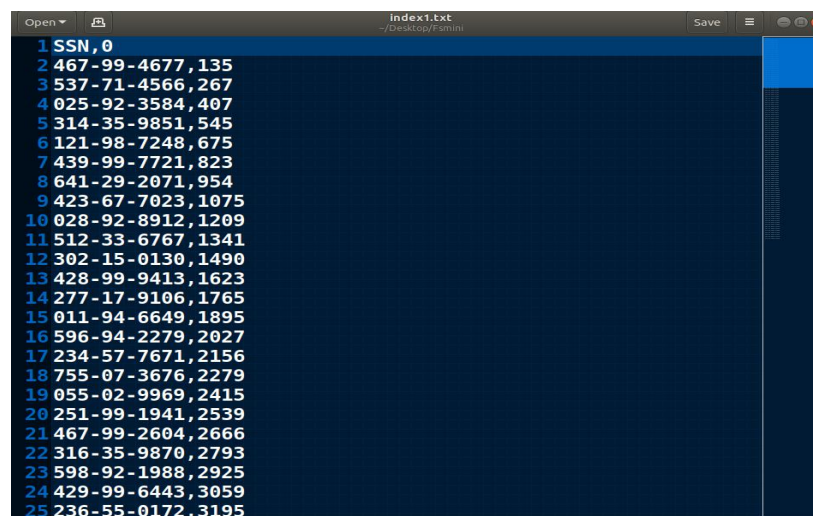


1	Emp ID,0
2	277509,135
3	940761,267
4	428945,407
5	408351,545
6	193819,675
7	499687,823
8	539712,954
9	380086,1075
10	477616,1209
11	162402,1341
12	231469,1490
13	153989,1623
14	386158,1765
15	301576,1895
16	441771,2027
17	528509,2156
18	912990,2279
19	214352,2415
20	890290,2539
21	622406,2666
22	979607,2793
23	969580,2925
24	426038,3059
25	388642,3195

Fig-3.1

### Secondary Index File(index1.txt)

The secondary-index file comprises of the secondary key and it's starting position in the record file. Any column in the record file can be considered as secondary key column. In the Human Resources records data-set used in this project "SSN" is being used as secondary key for the purpose of running various operations based on this secondary key. The snapshot of the secondary-index file is shown in the Fig-3.2.



```
1 SSN,0
2 467-99-4677,135
3 537-71-4566,267
4 025-92-3584,407
5 314-35-9851,545
6 121-98-7248,675
7 439-99-7721,823
8 641-29-2071,954
9 423-67-7023,1075
10 028-92-8912,1209
11 512-33-6767,1341
12 302-15-0130,1490
13 428-99-9413,1623
14 277-17-9106,1765
15 011-94-6649,1895
16 596-94-2279,2027
17 234-57-7671,2156
18 755-07-3676,2279
19 055-02-9969,2415
20 251-99-1941,2539
21 467-99-2604,2666
22 316-35-9870,2793
23 598-92-1988,2925
24 429-99-6443,3059
25 236-55-0172,3195
```

Fig-3.2

## INSERTING THE RECORDS

The user is given with an option to insert the records into the record file. Upon execution of the insertion operation, the new record will be inserted into the record file, primary-index file and also the secondary-index file. This record will be inserted at the end of the file, i.e it will be appended to these files. After insertion the index files are rebuilt once again.

This insertion operation is carried out by *getData()* method which reads all the required fields, and then *add()* method packs all these data and then writes it to corresponding files.

## **SEARCHING & MODIFYING THE RECORDS**

The user can search a record from the record file based on primary-index or secondary-index.

Searching using primary-index file prompts the user to enter the primary key, this key will be used to locate the record in the record file. This search operation makes use of the binary search algorithm in order to efficiently search the primary key and seek its position.

Searching using secondary-index prompts the user to enter the secondary key, this key will be used to locate all the records which contains this secondary key.

The user can also modify the records present in the record file. This modify() method permits the user to modify any field of the record which he has searched.

## **DELETING THE RECORDS**

Similar to search operation, in delete operation the user has the privilege to delete the records. Deletion of records can again be done based on primary-index or secondary-index.

The record to be deleted will be prefixed with a asterisk (\*) symbol, which indicates that the record is deleted and this record will not be considered while unpacking the records.

## **UNPACKING THE RECORDS**

Unpacking is another operation which is used to display all the records present in the record file. This operation is carried out by unpack() method which ignores the separator/delimiter (in this case ",") and considers only the fields present in the record file to display. While displaying the records, if a record starts with an asterisk(\*) then that record will not be unpacked or displayed.



## TIME COMPLEXITY

In Simple Indexing, the more the number of records in the record file, the more time it takes to build the index files. The graphical representation of the same is shown in the below figures.

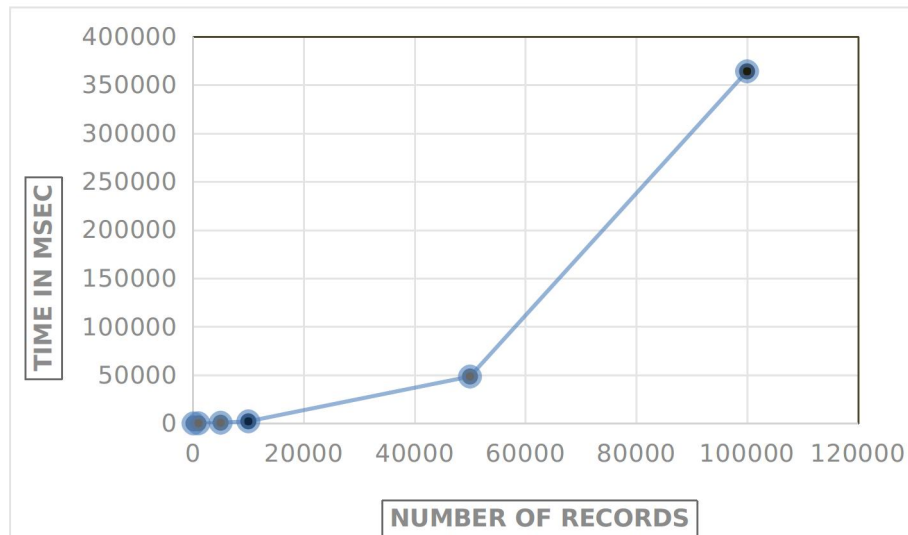


Fig-3.3 : Time taken for building primary index

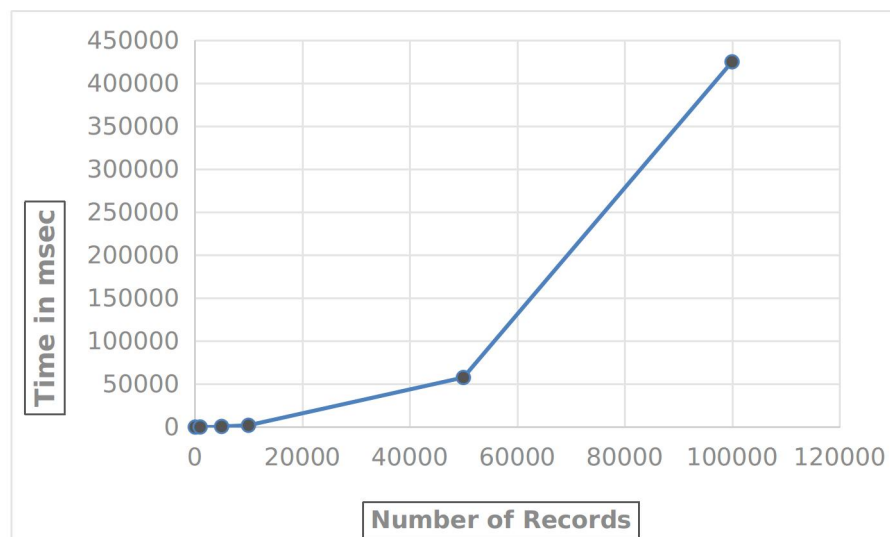
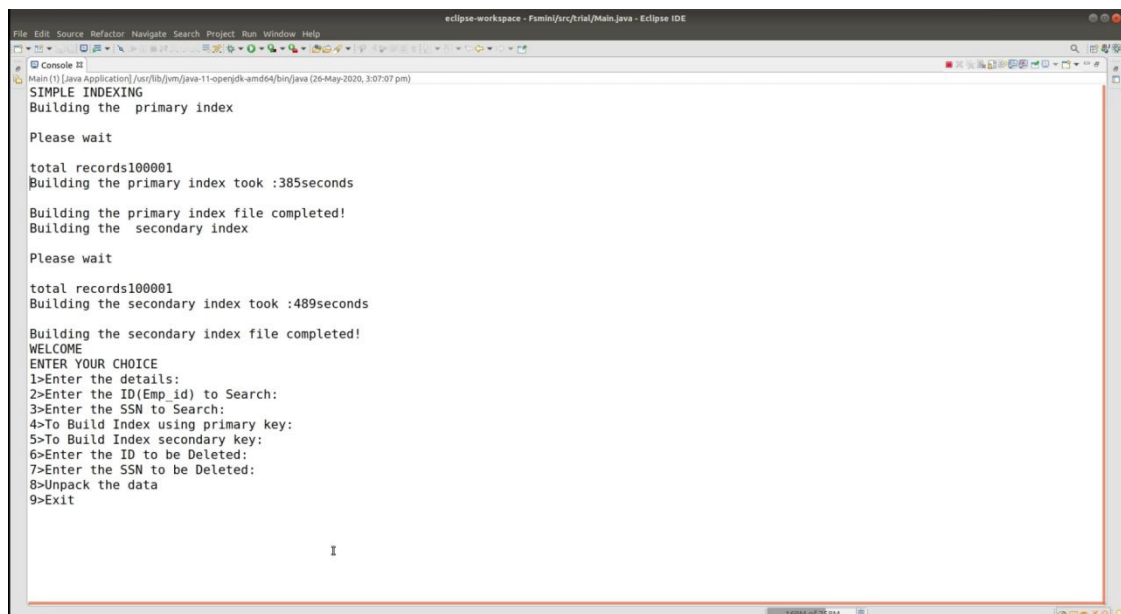


Fig-3.4 : Time taken for building secondary index

# ***CHAPTER 4***

## ***RESULTS AND SNAPSHOTS***

### BUILDING PRIMARY INDEX AND SECONDARY INDEX FILES:



```
eclipse-workspace - Fminj/src/trial/Main.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Main[1] [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (26-May-2020, 3:07:07 pm)
SIMPLE INDEXING
Building the primary index

Please wait

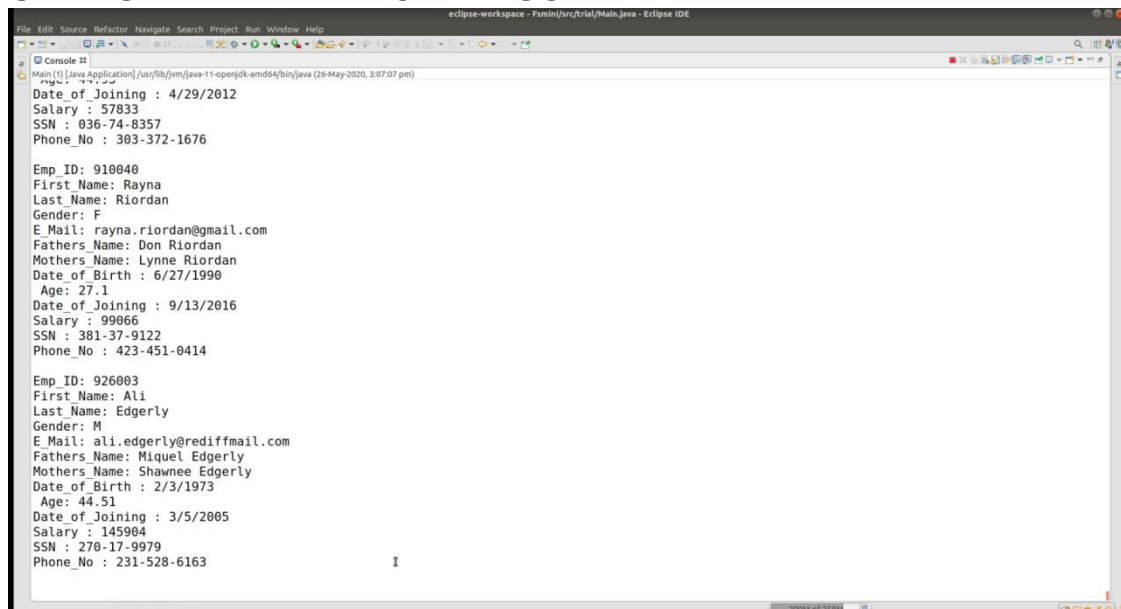
total records100001
Building the primary index took :385seconds
Building the primary index file completed!
Building the secondary index

Please wait

total records100001
Building the secondary index took :489seconds
Building the secondary index file completed!
WELCOME
ENTER YOUR CHOICE
1>Enter the details:
2>Enter the ID(Emp_id) to Search:
3>Enter the SSN to Search:
4>To Build Index using primary key:
5>To Build Index secondary key:
6>Enter the ID to be Deleted:
7>Enter the SSN to be Deleted:
8>Unpack the data
9>Exit
```

Fig - 4.1

### UNPACKED DATA FROM RECORD FILE :



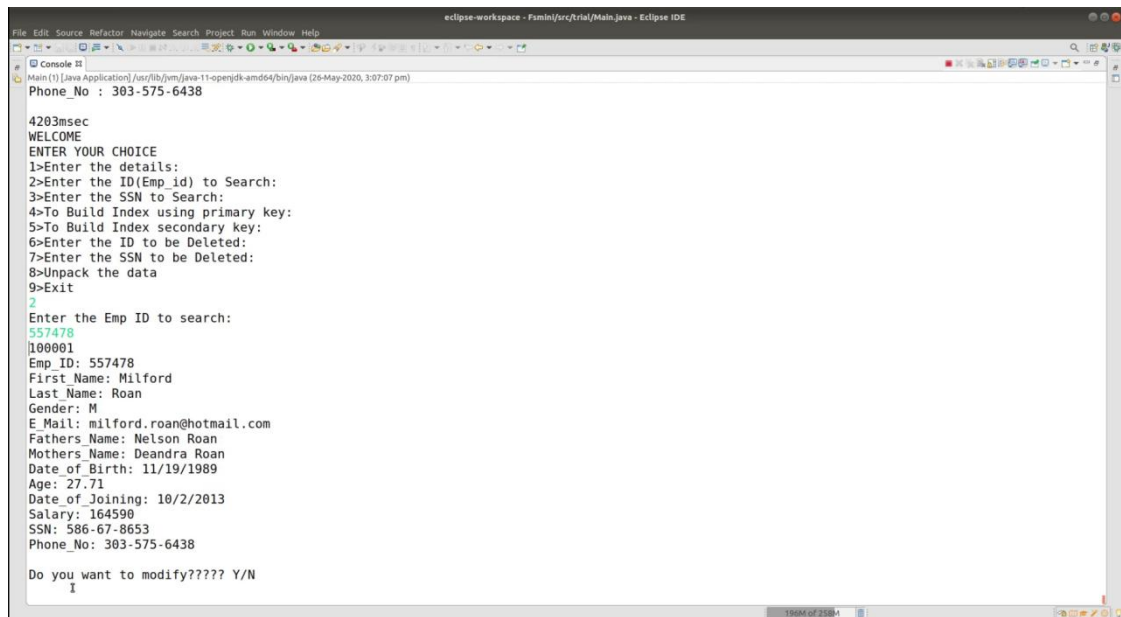
```
eclipse-workspace - Fminj/src/trial/Main.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Main[1] [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (26-May-2020, 3:07:07 pm)
Date_of_Joining : 4/29/2012
Salary : 57833
SSN : 036-74-8357
Phone_No : 303-372-1676

Emp ID: 910040
First Name: Rayna
Last Name: Riordan
Gender: F
E-Mail: rayna.riordan@gmail.com
Fathers Name: Don Riordan
Mothers Name: Lynne Riordan
Date_of_Birth : 6/27/1990
Age: 27.1
Date_of_Joining : 9/13/2016
Salary : 99066
SSN : 381-37-9122
Phone_No : 423-451-0414

Emp ID: 926003
First Name: Ali
Last Name: Edgerly
Gender: M
E-Mail: ali.edgerly@rediffmail.com
Fathers Name: Miquel Edgerly
Mothers Name: Shawnee Edgerly
Date_of_Birth : 2/3/1973
Age: 44.51
Date_of_Joining : 3/5/2005
Salary : 145904
SSN : 270-17-9979
Phone_No : 231-528-6163
```

Fig - 4.2

### SEARCHING RECORD USING PRIMARY KEY :



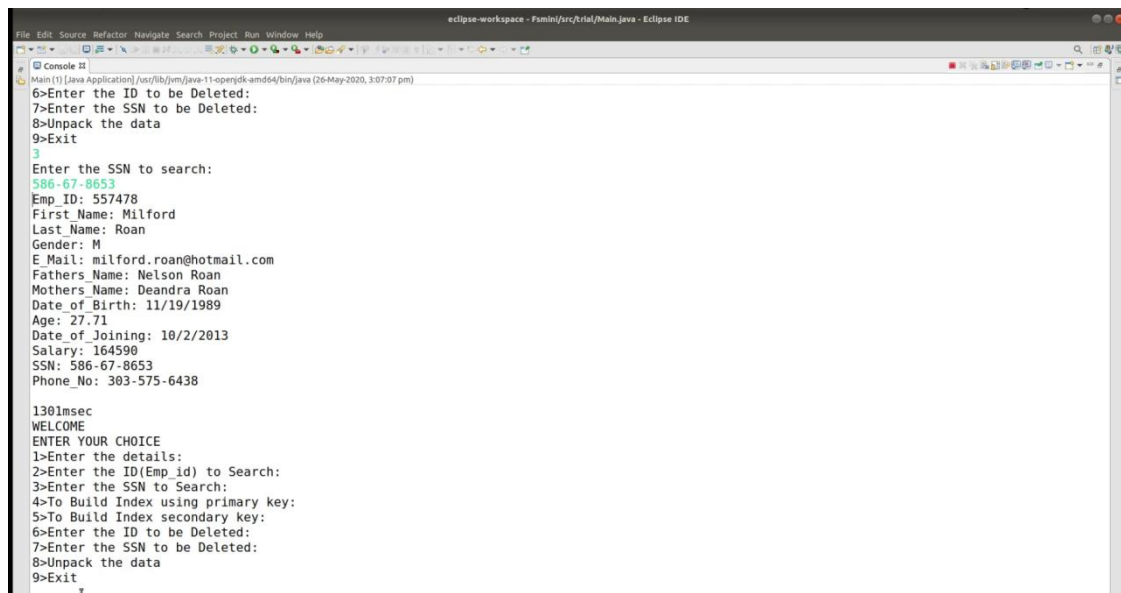
```
eclipse-workspace - Famin/jrc/Trial/Main.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Main [1] [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (26-May-2020, 3:07:07 pm)
Phone_No : 303-575-6438

4203msec
WELCOME
ENTER YOUR CHOICE
1>Enter the details:
2>Enter the ID(Emp_id) to Search:
3>Enter the SSN to Search:
4>To Build Index using primary key:
5>To Build Index secondary key:
6>Enter the ID to be Deleted:
7>Enter the SSN to be Deleted:
8>Unpack the data
9>Exit
2
Enter the Emp ID to search:
557478
100001
Emp_ID: 557478
First_Name: Milford
Last_Name: Roan
Gender: M
E-Mail: milford.roan@hotmail.com
Fathers_Name: Nelson Roan
Mothers_Name: Deandra Roan
Date_of_Birth: 11/19/1989
Age: 27.71
Date_of_Joining: 10/2/2013
Salary: 164590
SSN: 586-67-8653
Phone_No: 303-575-6438

Do you want to modify???? Y/N
I
```

Fig - 4.3

### SEARCHING RECORDS USING SECONDARY KEY:

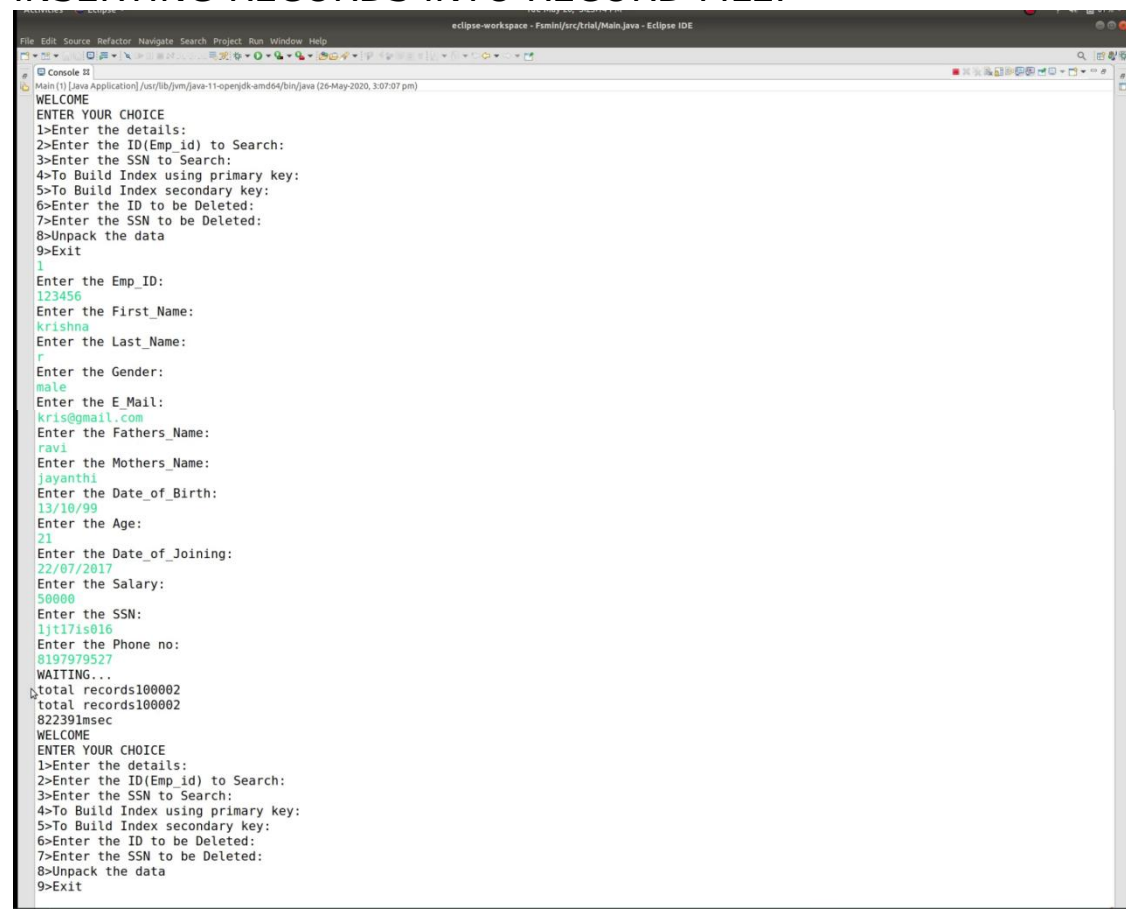


```
eclipse-workspace - Famin/jrc/Trial/Main.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Main [1] [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (26-May-2020, 3:07:07 pm)
6>Enter the ID to be Deleted:
7>Enter the SSN to be Deleted:
8>Unpack the data
9>Exit
3
Enter the SSN to search:
586-67-8653
Emp_ID: 557478
First_Name: Milford
Last_Name: Roan
Gender: M
E-Mail: milford.roan@hotmail.com
Fathers_Name: Nelson Roan
Mothers_Name: Deandra Roan
Date_of_Birth: 11/19/1989
Age: 27.71
Date_of_Joining: 10/2/2013
Salary: 164590
SSN: 586-67-8653
Phone_No: 303-575-6438

1301msec
WELCOME
ENTER YOUR CHOICE
1>Enter the details:
2>Enter the ID(Emp_id) to Search:
3>Enter the SSN to Search:
4>To Build Index using primary key:
5>To Build Index secondary key:
6>Enter the ID to be Deleted:
7>Enter the SSN to be Deleted:
8>Unpack the data
9>Exit
I
```

Fig - 4.4

### INSERTING RECORDS INTO RECORD FILE:



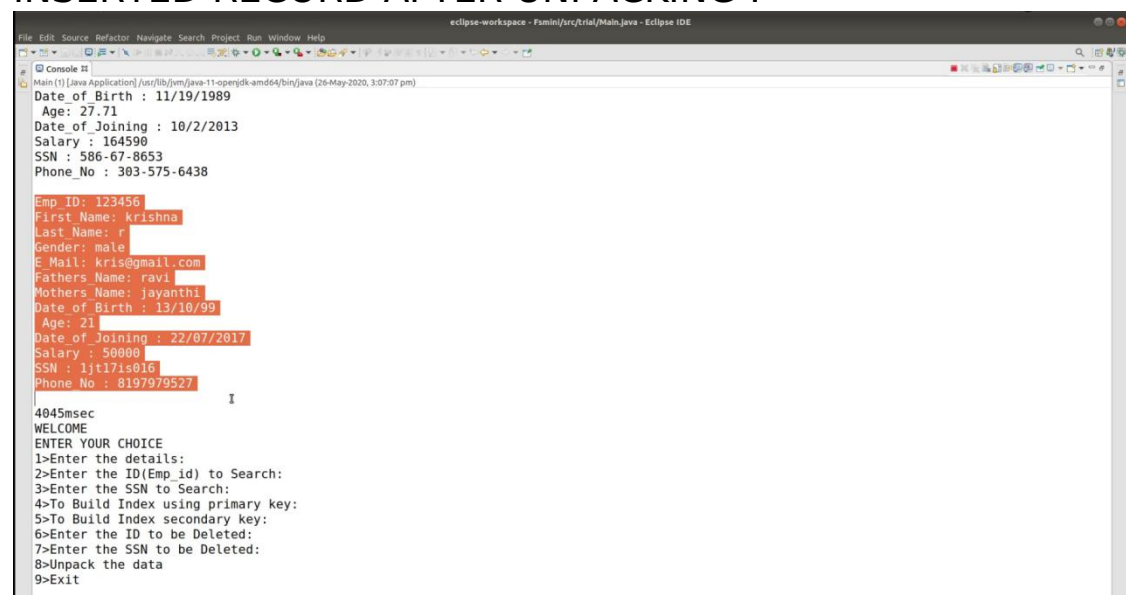
```
File Edit Source Refactor Navigate Search Project Run Window Help
eclipse-workspace - F:\min\src\trial\Main.java - Eclipse IDE

Main (1) [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (26-May-2020, 3:07:07 pm)

WELCOME
ENTER YOUR CHOICE
1>Enter the details:
2>Enter the ID(Emp_id) to Search:
3>Enter the SSN to Search:
4>To Build Index using primary key:
5>To Build Index secondary key:
6>Enter the ID to be Deleted:
7>Enter the SSN to be Deleted:
8>Unpack the data
9>Exit
1
Enter the Emp_ID:
123456
Enter the First_Name:
krishna
Enter the Last_Name:
r
Enter the Gender:
male
Enter the E-Mail:
kris@gmail.com
Enter the Fathers_Name:
ravi
Enter the Mothers_Name:
jayanthi
Enter the Date_of_Birth:
13/10/99
Enter the Age:
21
Enter the Date_of_Joining:
22/07/2017
Enter the Salary:
50000
Enter the SSN:
1jt17is016
Enter the Phone no:
8197979527
WAITING...
total records100002
total records100002
822391msec
WELCOME
ENTER YOUR CHOICE
1>Enter the details:
2>Enter the ID(Emp_id) to Search:
3>Enter the SSN to Search:
4>To Build Index using primary key:
5>To Build Index secondary key:
6>Enter the ID to be Deleted:
7>Enter the SSN to be Deleted:
8>Unpack the data
9>Exit
```

Fig - 4.5

### INSERTED RECORD AFTER UNPACKING :



```
File Edit Source Refactor Navigate Search Project Run Window Help
eclipse-workspace - F:\min\src\trial\Main.java - Eclipse IDE

Main (1) [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (26-May-2020, 3:07:07 pm)

Date of Birth : 11/19/1989
Age : 27.71
Date of Joining : 10/2/2013
Salary : 164590
SSN : 586-67-8653
Phone_No : 303-575-6438

Emp_ID: 123456
First Name: krishna
Last Name: r
Gender: male
E-Mail: kris@gmail.com
Fathers Name: ravi
Mothers Name: jayanthi
Date of Birth : 13/10/99
Age: 21
Date of Joining : 22/07/2017
Salary : 50000
SSN : 1jt17is016
Phone No : 8197979527

4045msec
WELCOME
ENTER YOUR CHOICE
1>Enter the details:
2>Enter the ID(Emp_id) to Search:
3>Enter the SSN to Search:
4>To Build Index using primary key:
5>To Build Index secondary key:
6>Enter the ID to be Deleted:
7>Enter the SSN to be Deleted:
8>Unpack the data
9>Exit
```

Fig - 4.6

## MODIFICATION OF RECORD:

```

File Edit Source Refactor Navigate Search Project Run Window Help
Main [1] Java Application /usr/lib/jvm/java-11-openjdk-amd64/bin/java (26-May-2020, 3:07:07 pm)
3>Enter the SSN to Search:
4>To Build Index using primary key:
5>To Build Index secondary key:
6>Enter the ID to be Deleted:
7>Enter the SSN to be Deleted:
8>Unpack the data
9>Exit
2
Enter the Emp ID to search:
123456
100002
Emp ID: 123456
First Name: krishna
Last Name: r
Gender: male
E-Mail: kris@gmail.com
Fathers Name: ravi
Mothers Name: jayanthi
Date of Birth: 13/10/99
Age: 21
Date of Joining: 22/07/2017
Salary: 50000
SSN: 1jt17is016
Phone_No: 8197979527

Do you want to modify???? Y/N
y
What do you want to change
Enter your choice
1.Emp_ID
2. First Name
3.Last Name
3
Enter the Last Name I
ravi
offset: 13440514
45688msec
WELCOME
ENTER YOUR CHOICE
1>Enter the details:
2>Enter the ID(Emp_id) to Search:
3>Enter the SSN to Search:
4>To Build Index using primary key:
5>To Build Index secondary key:
6>Enter the ID to be Deleted:
7>Enter the SSN to be Deleted:
8>Unpack the data
9>Exit
I
  
```

Fig - 4.7

## MODIFIED RECORD IN RECORD FILE :

```

humanresources3.txt
7/15/1963,54.07,3/26/1986,162787,219-89-5939,209-443-2183
99985 566415,Dan,January,M,dan.january@aol.com,Beau January,Leslee January,
10/20/1989,27.79,12/4/2010,130886,010-94-0828,201-733-0293
99986 845549,Ned,Wake,M,ned.wake@gmail.com,Wilmer Wake,Agatha Wake,6/28/1960,57.12,5/8/1997,90407,502-37-6525,423-757-6670
99987 555619,Melda,Pinard,F,melda.pinard@earthlink.net,Billie Pinard,Ola Pinard,
3/16/1975,42.4,10/1/1996,50387,003-08-8116,217-756-9751
99988 394104,Larhonda,Thorson,F,larhonda.thorson@gmail.com,Connie Thorson,Jolie Thorson,
5/7/1983,34.25,11/29/2005,124811,761-12-0276,212-547-5324
99989 361451,Jeff,Boynton,M,jeff.boynton@gmail.com,Ben Boynton,Charlie Boynton,
2/13/1971,46.48,3/15/2012,188445,126-98-5217,307-260-4814
99990 399895,Angila,Rapp,F,angila.rapp@aol.com,Sterling Rapp,Kiesha Rapp,5/18/1968,49.23,2/25/2012,126830,332-11-4287,252-403-6060
99991 851999,Eda,Pemberton,F,eda.pemberton@msn.com,Dannie Pemberton,Willie Pemberton,
7/19/1993,24.04,9/20/2015,153814,082-02-2239,210-255-6599
99992 182264,Rico,Schaller,M,rico.schaller@outlook.com,Neville Schaller,Dolores Schaller,
6/28/1986,31.1,2/29/2016,176446,165-86-5780,201-574-9448
99993 965070,Stephnie,Jacobsen,F,stephnie.jacobsen@hotmail.com,Steve Jacobsen,Toshiko Jacobsen,
12/4/1983,33.67,2/19/2013,187800,149-23-0430,209-240-1382
99994 684740,Jonah,Delk,M,jonah.delk@bp.com,Kevin Delk,Nam Delk,3/4/1983,34.42,12/24/2007,185993,343-11-2262,212-416-3717
99995 154798,Mattie,Herzog,F,mattie.herzog@hotmail.com,Cory Herzog,Pennie Herzog,
3/12/1963,54.42,2/18/1993,84646,221-13-3591,212-641-5592
99996 246306,Mac,Savino,M,mac.savino@gmail.com,John Savino,Earnestine Savino,
11/26/1995,21.68,11/22/2016,98988,500-29-4176,503-942-9749
99997 855626,Dionne,Starkes,F,dionne.starkes@aol.com,Ashley Starkes,Lennie Starkes,
10/11/1961,55.83,4/15/1990,115053,761-12-1977,314-371-0817
99998 987249,Jessi,Bitting,F,jessi.bitting@hotmail.com,Lupe Bitting,Anja Bitting,
7/25/1991,26.03,7/14/2014,79584,588-09-0064,217-673-6086
99999 435859,Alfredo,Leiva,M,alfredo.leiva@gmail.com,Laverne Leiva,Nina Leiva,
4/16/1959,58.32,4/14/1994,57815,196-84-8703,319-873-8798
100000 504037,Rocky,Winston,M,rocky.winston@hotmail.com,Seth Winston,Joycelyn Winston,
5/26/1986,31.19,6/8/2007,137646,422-67-4218,215-990-3923
100001 557478,Milford,Roan,M,milford.roan@hotmail.com,Nelson Roan,Deandra Roan,
11/19/1989,27.71,10/2/2013,164590,586-67-8653,303-575-6438
100002 123456,krishna,ravi,male,kris@gmail.com,ravi,jayanthi,13/10/99,21,22/07/2017,50000,1jt17is016,8197979527
100003 123456,krishna,ravi,male,kris@gmail.com,ravi,jayanthi,13/10/99,21,22/07/2017,50000,1jt17is016,8197979527
  
```

Fig - 4.8



## DELETING THE RECORD USING PRIMARY KEY :

```

eclipse-workspace - F:\min\src\trial\Main.java - Eclipse IDE
Main (1) [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (26-May-2020, 3:07:07 pm)
3.Last_Name
3
Enter the Last_Name
ravi
offset: 13440514
45688msec
WELCOME
ENTER YOUR CHOICE
1>Enter the details:
2>Enter the ID(Emp_id) to Search:
3>Enter the SSN to Search:
4>To Build Index using primary key:
5>To Build Index secondary key:
6>Enter the ID to be Deleted:
7>Enter the SSN to be Deleted:
8>Unpack the data
9>Exit
6
Enter the primary key to delete record
557478
WAIT FOR FEW SECONDS....:
13157msec
WELCOME
ENTER YOUR CHOICE
1>Enter the details:
2>Enter the ID(Emp_id) to Search:
3>Enter the SSN to Search:
4>To Build Index using primary key:
5>To Build Index secondary key:
6>Enter the ID to be Deleted:
7>Enter the SSN to be Deleted:
8>Unpack the data
9>Exit
1

```

Fig - 4.9

## DELETED RECORD (\*)

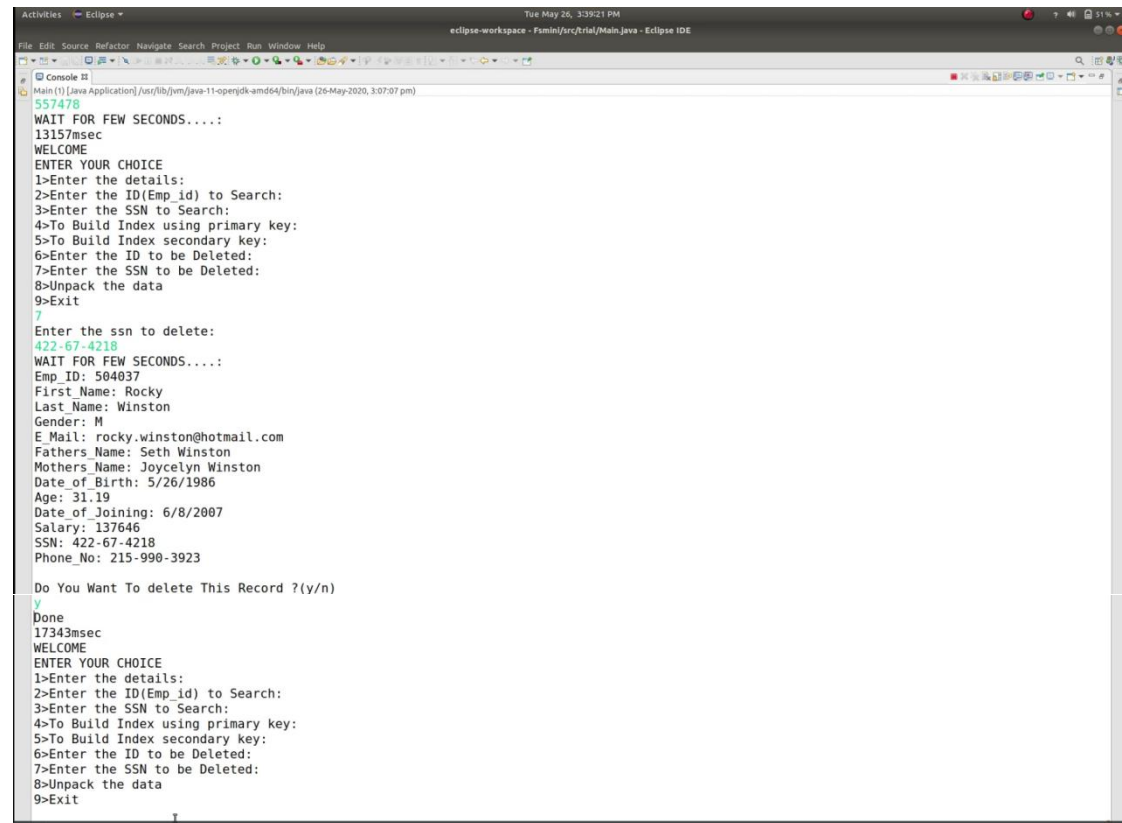
```

humanresources3.txt
7/15/1963, 54.07, 3/26/1986, 162787, 219-89-5939, 209-443-2183
99985 566415, Dan, January, M, dan.january@aol.com, Beau January, Leslee January,
10/20/1989, 27.79, 12/4/2010, 130886, 010-34-0828, 201-733-0293
99986 845549, Ned, Wake, M, ned.wake@gmail.com, Wilmer Wake, Agatha Wake, 6/28/1960, 57.12, 5/8/1997, 90407, 502-37-6525, 423-757-6670
99987 555619, Melda, Pinard, F, melda.pinard@earthlink.net, Billie Pinard, Ola Pinard,
3/16/1975, 42.4, 10/1/1996, 50387, 003-08-8116, 217-756-9751
99988 394104, Larhonda, Thorson, F, larhonda.thorson@gmail.com, Connie Thorson, Jolie Thorson,
5/7/1983, 34.25, 11/29/2005, 124811, 761-12-0276, 212-547-5324
99989 361451, Jeff, Boynton, M, jeff.boynton@gmail.com, Ben Boynton, Charlie Boynton,
2/13/1971, 46.48, 3/15/2012, 188445, 126-98-5217, 307-260-4814
99990 399895, Angila, Rapp, F, angila.rapp@aol.com, Sterling Rapp, Kiesha Rapp, 5/18/1968, 49.23, 2/25/2012, 126830, 332-11-4287, 252-403-6060
99991 851999, Eda, Pemberton, F, eda.pemberton@msn.com, Dannie Pemberton, Willie Pemberton,
7/19/1993, 24.04, 9/20/2015, 153814, 082-02-2239, 210-255-6599
99992 182264, Rico, Schaller, M, rico.schaller@outlook.com, Neville Schaller, Dolores Schaller,
6/28/1986, 31.1, 2/29/2016, 176446, 165-86-5780, 201-574-9448
99993 965070, Stephanie, Jacobsen, F, stephnie.jacobsen@hotmail.com, Steve Jacobsen, Toshiko Jacobsen,
12/4/1983, 33.67, 2/19/2013, 107800, 149-23-0430, 209-240-1382
99994 684740, Jonah, Delk, M, jonah.delk@bp.com, Kevin Delk, Nam Delk, 3/4/1983, 34.42, 12/24/2007, 185993, 343-11-2262, 212-416-3717
99995 154798, Mattie, Herzog, F, mattie.herzog@hotmail.com, Cory Herzog, Pennie Herzog,
3/12/1963, 54.42, 2/18/1993, 84646, 221-13-3591, 212-641-5592
99996 246306, Mac, Savino, M, mac.savino@gmail.com, John Savino, Earnestine Savino,
11/26/1995, 21.68, 11/22/2016, 98988, 500-29-4176, 503-942-9749
99997 855626, Dionne, Starkes, F, dionne.starkes@aol.com, Ashley Starkes, Lennie Starkes,
10/11/1961, 55.83, 4/15/1990, 115053, 761-12-1977, 314-371-0817
99998 987249, Jessi, Bitting, F, jessi.bitting@hotmail.com, Lupe Bitting, Anja Bitting,
7/25/1991, 26.03, 7/14/2014, 79584, 588-09-0064, 217-673-6086
99999 435859, Alfredo, Leiva, M, alfredo.leiva@gmail.com, Laverne Leiva, Nina Leiva,
4/16/1959, 58.32, 4/14/1994, 57815, 196-84-8703, 319-873-8798
100000 504037, Rocky, Winston, M, rocky.winston@hotmail.com, Seth Winston, Joycelyn Winston,
5/26/1986, 31.19, 6/8/2007, 137646, 422-67-4218, 215-990-3923
100001 *57478, Milford, Roan, M, milford.roan@hotmail.com, Nelson Roan, Deandra Roan,
11/19/1989, 27.71, 10/2/2013, 164590, 586-67-8653, 303-575-6438
100002 *23456, Krishna, R, male, kris@gmail.com, ravi, jayanthi, 13/10/99, 21, 22/07/2017, 50000, 1jt17is016, 8197979527
100003 123456, Krishna, ravi, male, kris@gmail.com, ravi, jayanthi, 13/10/99, 21, 22/07/2017, 50000, 1jt17is016, 8197979527

```

Fig - 4.10

### DELETING THE RECORD USING SECONDARY KEY :



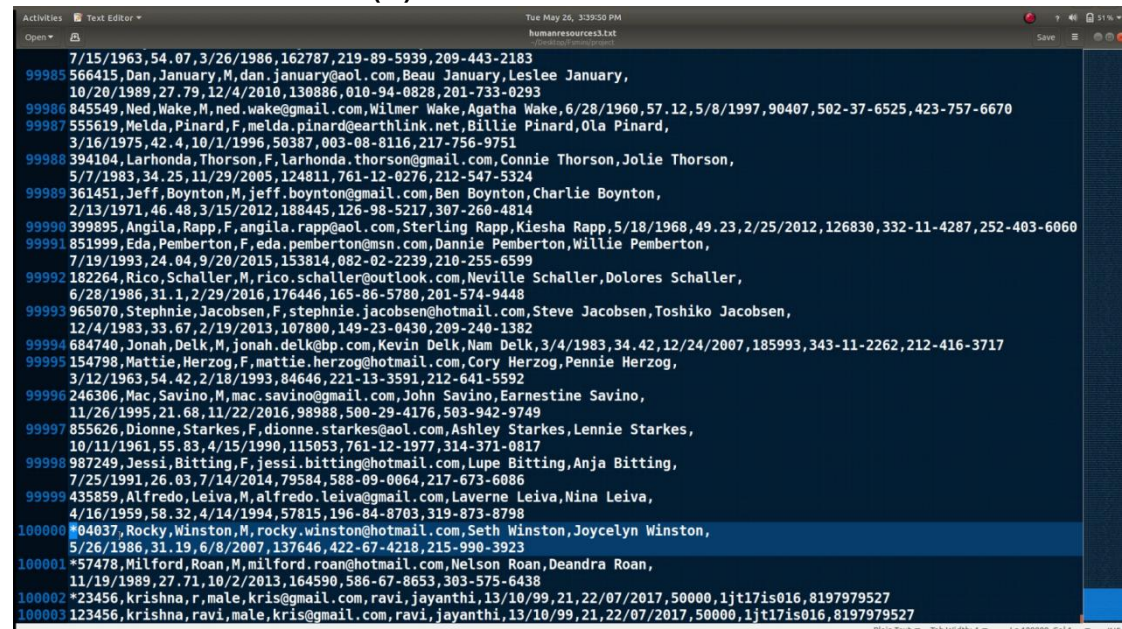
```
Activities - Eclipse
Tue May 26, 3:39:21 PM
eclipse-workspace - Pamin/jsrc/trial/Main.java - Eclipse IDE

Main [1] [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (26-May-2020, 3:07:07 pm)
557478
WAIT FOR FEW SECONDS....:
13157msec
WELCOME
ENTER YOUR CHOICE
1>Enter the details:
2>Enter the ID(Emp_id) to Search:
3>Enter the SSN to Search:
4>To Build Index using primary key:
5>To Build Index secondary key:
6>Enter the ID to be Deleted:
7>Enter the SSN to be Deleted:
8>Unpack the data
9>Exit
7
Enter the ssn to delete:
422-67-4218
WAIT FOR FEW SECONDS....:
Emp_ID: 504037
First Name: Rocky
Last Name: Winston
Gender: M
E Mail: rocky.winston@hotmail.com
Fathers Name: Seth Winston
Mothers Name: Joycelyn Winston
Date of Birth: 5/26/1986
Age: 31.19
Date of Joining: 6/8/2007
Salary: 137646
SSN: 422-67-4218
Phone_No: 215-990-3923

Do You Want To delete This Record?(y/n)
y
Done
17343msec
WELCOME
ENTER YOUR CHOICE
1>Enter the details:
2>Enter the ID(Emp_id) to Search:
3>Enter the SSN to Search:
4>To Build Index using primary key:
5>To Build Index secondary key:
6>Enter the ID to be Deleted:
7>Enter the SSN to be Deleted:
8>Unpack the data
9>Exit
```

Fig - 4.11

### DELETED RECORD (\*)



```
Activities - Text Editor
Tue May 26, 3:39:50 PM
humanresources.txt
Save

7/15/1963,54.07,3/26/1986,162787,219-89-5939,209-443-2183
99985 566415,Dan,January,M,dan.january@aol.com,Beau January,Leslee January,
10/20/1989,27.79,12/4/2010,130886,010-94-0828,201-733-0293
99986 845549,Ned,Wake,M,ned.wake@gmail.com,Wilmer Wake,Agatha Wake,6/28/1960,57.12,5/8/1997,90407,502-37-6525,423-757-6670
99987 555619,Melda,Pinard,F,melda.pinard@earthlink.net,Billie Pinard,Ola Pinard,
3/16/1975,42.4,10/1/1996,50387,003-08-8116,217-756-9751
99988 394104,Larhonda,Thorson,F,larhonda.thorson@gmail.com,Connie Thorson,Jolie Thorson,
5/7/1983,34.25,11/29/2005,124811,761-12-0276,212-547-5324
99989 361451,Jeff,Boynton,M,jeff.boynton@gmail.com,Ben Boynton,Charlie Boynton,
2/13/1971,46.48,3/15/2012,188445,126-98-5217,307-260-4814
99990 399895,Angila,Rapp,F,angila.rapp@aol.com,Sterling Rapp,Kiesha Rapp,5/18/1968,49.23,2/25/2012,126830,332-11-4287,252-403-6060
99991 851999,Eda,Pemberton,F,eda.pemberton@msn.com,Dannie Pemberton,Willie Pemberton,
7/19/1993,24.04,9/20/2015,153814,082-02-2239,210-255-6599
99992 182264,Rico,Schaller,M,rico.schaller@outlook.com,Neville Schaller,Dolores Schaller,
6/28/1986,31.1,2/29/2016,176446,165-86-5780,201-574-9448
99993 965070,Stephnie,Jacobsen,F,stephnie.jacobsen@hotmail.com,Steve Jacobsen,Toshiko Jacobsen,
12/4/1983,33.67,2/19/2013,107800,149-23-0430,209-240-1382
99994 684740,Jonah,Delk,M,jonah.delk@bp.com,Kevin Delk,Nam Delk,3/4/1983,34.42,12/24/2007,185993,343-11-2262,212-416-3717
99995 154798,Mattie,Herzog,F,mattie.herzog@hotmail.com,Cory Herzog,Pennie Herzog,
3/12/1963,54.42,2/18/1993,84646,221-13-3591,212-641-5592
99996 246306,Mac,Savino,M,mac.savino@gmail.com,John Savino,Earnestine Savino,
11/26/1995,21.68,11/22/2016,98988,500-29-4176,503-942-9749
99997 855626,Dionne,Starkes,F,dionne.starkes@aol.com,Ashley Starkes,Lennie Starkes,
10/11/1961,55.83,4/15/1990,115053,761-12-1977,314-371-0817
99998 987249,Jessi,Bitting,F,jessi.bitting@hotmail.com,Lupe Bitting,Anja Bitting,
7/25/1991,26.03,7/14/2014,79584,588-09-0064,217-673-6086
99999 435859,Alfredo,Leiva,M,alfredo.leiva@gmail.com,Laverne Leiva,Nina Leiva,
4/16/1959,58.32,4/14/1994,57815,196-84-8703,319-873-8798
100000 504037,Rocky,Winston,M,rocky.winston@hotmail.com,Seth Winston,Joycelyn Winston,
5/26/1986,31.19,6/8/2007,137646,422-67-4218,215-990-3923
100001 57478,Milford,Roan,M,milford.roan@hotmail.com,Nelson Roan,Deandra Roan,
11/19/1989,27.71,10/2/2013,164590,586-67-8653,303-575-6438
100002 223456,krishna,r,male,kris@gmail.com,ravi,jayanthi,13/10/99,21,22/07/2017,50000,1jtl7is016,8197979527
100003 123456,krishna,ravi,male,kris@gmail.com,ravi,jayanthi,13/10/99,21,22/07/2017,50000,1jtl7is016,8197979527
```

Fig - 4.12



## **Conclusion**

The main objective of Indexing is to optimize the performance of file access by minimizing the number of disk accesses required to process the required data' has been achieved.

Indexes are used to quickly locate data without having to search every row in a database, every time a database record is accessed.

## **Future Enhancements**

In this project, building the index files takes up a lot of time depending on the size of the record file. The larger the size of the record file, the more time it takes to build the index files. In case when the size of the index file is too large and exceeds the size of the main memory, then handling the index file itself will not be possible.

Therefore, we can rely on other data structures such as B-trees, Hashing, Extensible Hashing etc to build index files for large record files efficiently and within short period of time.

## References:

- [1] <http://reddyfsproject.blogspot.com/2012/11/indexing.html>
- [2] <https://www.kullabs.com/classes/subjects/units/lessons/notes/note-detail/5668#:~:text=Indexing%20helps%20to%20locate%20the,time%20and%20effort%20of%20employees>
- [3] [https://www.cs.uct.ac.za/mit\\_notes/database/htmls/chp11.html](https://www.cs.uct.ac.za/mit_notes/database/htmls/chp11.html)
- [4] [https://www.tutorialspoint.com/dbms/dbms\\_indexing.htm](https://www.tutorialspoint.com/dbms/dbms_indexing.htm)
- [5] [https://www.youtube.com/playlist?list=PLhpq7\\_v\\_\\_PAzRbm4a6GrvFs0Q6zcW\\_Ezd](https://www.youtube.com/playlist?list=PLhpq7_v__PAzRbm4a6GrvFs0Q6zcW_Ezd)
- [6] file structures an object-oriented approach with c++ 3rd edition
- [7] [https://www.researchgate.net/publication/333843847\\_A\\_Study\\_on\\_Indexes\\_and\\_Index\\_Structures](https://www.researchgate.net/publication/333843847_A_Study_on_Indexes_and_Index_Structures)