**Visvesvaraya Technological University**

**Belagavi-590018, Karnataka**



A Mini Project Report on

**“Indexing in Sales management system”**

**Submitted in partial fulfilment of the requirement for the**

**File Structures Lab [17ISL68]**

**Bachelor of Engineering**

**in**

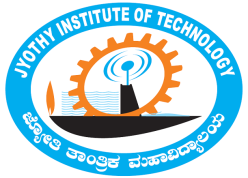
**Information Science and Engineering**

**Submitted by**

**Madhurani L[1JT17IS019]**

**Under the guidance of Mr. Vadiraja**

**Asst.Professor, Department of ISE**



**Department of Information Science and Engineering**

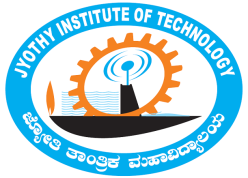
**Jyothy Institute of Technology**

**Tataguni, Bengaluru-560082**

**Jyothy Institute of Technology**

**Tataguni, Bengaluru-560082**

**Department of Information Science and Engineering**



**CERTIFICATE**

Certified that the mini project work entitled **“Sales management system”** carried out by **Madhurani L[1JT17IS019]** bonafide student of Jyothy Institute of Technology, in partial fulfilment for the award of **Bachelor of Engineering** in **Information Science and Engineering** department of the **Vishvesvaraya Technological University, Belagavi** during the year **2019-2020**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The mini project report has been approved as it satisfies the academic requirements in respect of Mini Project work prescribed for the said Degree.

**Mr. Vadiraja I A, Dr. Harshavardan Tiwari,**

Guide, asst.Professor Professor and HoD

Dept. Of ISE Dept. Of ISE

External Viva Examiner Signature with Date:

**ACKNOWLEDGEMENT**

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

I express our sincere thanks to our Principal **Dr. Gopalakrishna K for** providing us 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 his valuable support.

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

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

**Madhurani L [1JT17IS019]**

**ABSTRACT**

Sales management is a key function which helps small and medium size enterprises (SMEs) in monitoring and tracking stock and co-ordinating transaction processing. The efficiency of sales management depends on effective tools and facilities, especially modern information and communication technologies. Despite this, majority of businesses in developing countries, especially those in remote areas do not take full advantage of these technologies due to challenges related to the design of these technologies. This paper presents the design and development of a tailor-made computerized sales management system for SMEs in Northern Ghana. The object-oriented methodology is employed with UML, VB.NET and Microsoft Access Database for the design and development of the system which is flexible and tailor-made for SMEs in the region.

.

|  |  |  |
| --- | --- | --- |
| **Sl.no** | **Description** | **Page no** |
|  | **Chapter 1** |  |
|  | **Introduction** |  |
| 1.1 | Introduction to File Structures | 1-1 |
| 1.2 | Introduction to File System | 1-1 |
| 1.3 | Introduction to Simple Record Structure | 1-2 |
| 1.4 | Introduction to Sales Management System | 2-2 |
| 1.5 | Introduction to Python | 2-2 |
|  | **Chapter 2** |  |
|  | **Design** |  |
| 2.1 | Domain Understanding | 3-3 |
| 2.2 | Time Analysis | 3-3 |
| 2.3 | Hardware Requirement and Software Requirement | 3-4 |
|  | **Chapter 3** |  |
|  | 1. **Implementation** |  |
| 3.1 | Algorithm to Write the Record into File | 5-6 |
| 3.2 | Algorithm to Read the Record Form File: | 7-7 |
| 3.3 | Algorithm to Search Based no Primary Index | 7-8 |
| 3.4 | Algorithm to Search Based no Secondary Index | 9-8 |
| 3.5 | Algorithm to Modify record | 9-12 |
| 3.6 | Algorithm to Delete the record | 12-12 |
|  | **Chapter 4** |  |
|  | **Results And snapshots** |  |
| 4.1 | Inserting Record into File | 13-14 |
| 4.2 | Reading Record from File | 15-15 |
| 4.3 | Searching Record in the File | 15-16 |
| 4.4 | Modifying Record in the File | 16-19 |
| 4.5 | Deleting Record from File | 20-21 |
| 4.6 | Inserting Record to Free Space | 22-23 |
| 4.7 | Time Graph | 23-24 |
| 4.8 | Conclusion and Features | 24-24 |
| 4.9 | Reference | 24-24 |

***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 minimize the access time and the storage space.
* A File Structure is a combination of representations for data in files and of operations for accessing the data.
* A File Structure 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.
* As files grew very large, unaided sequential access was not a good solution.
* Disks allowed for direct access.
* Indexes made it possible to keep a list of keys and pointers in a small file that could be searched very quickly.
* With the key and pointer, the user had direct access to the large, primary file.

**1.2 Introduction to File System**

File system is the part of the operating system which is responsible for file management. It provides a mechanism to store the data and access to the file contents including data and programs. Some Operating systems treats everything as a file.

* **Simple Record Structure** with lines of fixed or variable lengths.
* **Complex Structures** like formatted document or reloadable load files.
* **No Definite Structure** like sequence of words and bytes etc.

**1.3 Introduction to Simple Record Structure**

* **Fixed Length**: Fixed length means having a set length that never varies. In [database systems](https://www.webopedia.com/TERM/D/database_management_system_DBMS.html), a [field](https://www.webopedia.com/TERM/F/field.html) can have afixedor a[variable length](https://www.webopedia.com/TERM/V/variable_length.html). A variable-length field is one whose length can be different in each [record](https://www.webopedia.com/TERM/R/record.html), depending on what [data](https://www.webopedia.com/TERM/D/data.html) is [stored](https://www.webopedia.com/TERM/S/store.html) in the field. 1
* **Variable Length:** Variable-length fields are useful because they save space. Suppose, for example, that you want to define a NAME field. The length of each NAME field will vary according to the data placed in it. For example, John Smith is 10 [characters](https://www.webopedia.com/TERM/C/character.html) long, but Thomas Horatio Jefferson is 24 characters long. With fixed-length fields, you would need to define each field to be long enough to hold the longest name. This would be a waste of space for [records](https://www.webopedia.com/TERM/R/record.html) that had short names. With variable-length fields, the NAME field in each record would be just long enough to hold its data.

**1.4 Introduction to Sales Management System**

Sales management system is to maintain the information about co-ordinating transaction processing. This is used in small and medium size enterprises. In this system progress Sales record can be read, write, modify and can be deleted. Where operating system which is responsible for file management. It provides a mechanism to store the data and access.

**1.5 Introduction to Python**

[Python](https://www.geeksforgeeks.org/python-programming-language/) is a widely used general-purpose, high level programming language. It was created by Guido van Rossum in 1991 and further developed by the Python Software Foundation. It was designed with an emphasis on code readability, and its syntax allows programmers to express their concepts in fewer lines of code.

2

***CHAPTER 2***

***DESIGN***

**2.1 Domain Understanding**

In this project variable length record are used to store the record because to save space. Region, country, itemtype, saleschannel, orderpriority, orderdate, orderid, shipdate, unitsold, unitprice, unitcost, totalrevenue, totalcost, totalprofit these are the field used to pack a record. Orderid used as a primary key and itemtype used as a secondary key. For saving space avail list, best fit strategy is used, this method keeps the list in order by size, smallest to largest. In this method, the operating system first searches the whole of the memory according to the size of record and allocates it to the closest-fitting free position in the memory, making it able to use memory efficiently.

**2.2 Time Analysis**

Time Complexities: O(n)

An algorithm is said to have a linear time complexity when the running time increases at most linearly with the size of the input data. This is the best possible time complexity when the algorithm must examine all values in the input data.

In this program, all the record will be read one by one from the file. As the number of record increases time also increases.

**2.3 Hardware Requirement and Software Requirement**

**Hardware Requirement:**

Operating System - Windows 10

Processor – x64 architecture

Memory – 8.00RAM

Hard Disk – 1TB

3

**Software Requirement:**

Software – Anaconda3-2019(Spyder3.3.6)

Programming Language - Python

Front End - HTML, CSS, Java Script

4

***CHAPTER 3***

***IMPLEMENTATION***

**3.1 Algorithm to Write The Record Into File**

Step 1: Take all the required input form user.

Step 2 : As I am dealing with variable length record it will pack the user input using “|” as delimiter.

Eg: pack1=self.region+"|"+self.country+"|"+self.itemtype+"|"+self.saleschannel+"|"+self.orderpriority+"|"+self.orderdate+"|"+self.orderid+"|"+self.shipdate+"|"+self.unitsold+"|"+self.unitprice+"|"+self.unitcost+"|"+self.totalrevenue+"|"+self.totalcost+"|"+self.totalprofit+"|"

Step 3: check length of the packed record and search avail list if there is any space that is less than or equal to packed length then open the file in r+ mode and seek the position and write into the file and note the position.

length=len(pack1)

if len(self.avllist1) !=0:

for i in range (0,len(self.avllist1)):

ab=self.avllist1[i].split("|")

if(length<=int(ab[0])):

file1=open(r"C:\Users\Madhurani L\Desktop\stu.txt","r+")

file1.seek(int(ab[1]))

file1.write(pack1)

file1.write("\n")

Step 4: If avail list element space is greater than length of the packed record then remaining content will be marked as star(deleted) then size of the remaining content and position will be stored in avail list and sort the avail list.

length=len(l2)

a=l2.split("|")

first=a[0][0]

file1.write(l2.replace(first,"\*")

var=""

var=var+len2+"|"+p2+"|" 5

self.avllist1.append(var)

self.avllist1.sort()

Step 5: Next open the primary index file in r+ mode, pack primary index with position using delimiter and write into the primary index file.

File2=open(r"C:\Users\Madhurani L\Desktop\stu1.txt","r+")

pack2=pack2+self.orderid+"|"+position+"|"

file2.write(pack2)

file2.write("\n")

Step 6: Next open the secondary index file in r+ mode, pack secondary index with position using delimiter and write into the secondary index file.

file3=open(r"C:\Users\Madhurani L\Desktop\stu2.txt","r+")

pack3=pack3+self.itemtype+"|"+position+"|"

file3.write(pack3)

file3.write("\n")

Step 7: If there is no any space that is less than or equal to packed length then open the file in append go to empty line, write the content and note the position.

File1=open(r"C:\Users\Madhurani L\Desktop\stu.txt","r+")

position=file1.tell()

line=file1.readline()

while line:

position=file1.tell()

line=file1.readline

file1.write(pack1)

file1.write("\n")

print(pack1)

Step 8: Go to step 5 and step 6.

6

**3.2 Algorithm To Read The Record From File**

Step 1: open the file in read mode.

Step 2: Read the line and store in one variable. Split variable based on “|” while, line in not equal to null and if record is not deleted then print the record.

file1=open(r"C:\Users\Madhurani L\Desktop\stu.txt","r")

line=file1.readline()

a=line.split("|")

while line:

if (a[0][0]!="\*"):

print("Region:",a[0])

print("Country:",a[1])

print("Itemtype: ",a[2])

…….

…….

**3.3 Algorithm To Search Based no Primary Index**

Step 1: Binary Search is based on primary Index, if position is found then open primary index file in r+ mode and seek the position, read line and split based on “|” and get position.

file2=open(r"C:\Users\Madhurani L\Desktop\stu1.txt","r+")

file2.seek(int(self.position2))

self.line2=file2.readline()

self.b=self.line2.split("|")

self.position1=self.b[1]

Step 2: open file with r+ mode and seek the position and read line and split based on “|” and print.

file1=open(r"C:\Users\Madhurani L\Desktop\stu.txt","r+")

file1.seek(int(self.position1))

self.line1=file1.readline()

self.length=len(self.line1)

self.a=self.line1.split("|")

self.Region=self.a[0] 7

self.Country=self.a[1]

self.Itemtype=self.a[2]

………

……….

Step 3: If position is not found then print record not found.

print("NOT THERE IN THE RECORD")

**3.4 Algorithm to Search Based no Secondary Index**

Step 1: Binary Search is based on secondary Index, if position is found then open secondary index file in r+ mode, seek the position, read line and split based on “|” and get position.

file2=open(r"C:\Users\Madhurani L\Desktop\stu1.txt","r+")

file2.seek(int(self.position2))

self.line2=file2.readline()

self.b=self.line2.split("|")

self.position1=self.b[1]

Step 2: open file with r+ mode and seek the position then read the line and split based on “|” and print.

file1=open(r"C:\Users\Madhurani L\Desktop\stu.txt","r+")

file1.seek(int(self.position1))

self.line1=file1.readline()

self.length=len(self.line1)

self.a=self.line1.split("|")

self.Region=self.a[0]

self.Country=self.a[1]

…….

Step 3: search previous and next line then read the line and split then check if secondary name match are not.

pos=pos-1

n=self.list2[pos].split("|") 8

while(str(n[0])==self.nam):

if(pos<0):

break

self.readfile(n[1])

pos=pos-1

n=self.list2[pos].split("|")

pos=self.flag+1

if(pos+1<=len(self.list2)):

n=self.list2[pos].split("|")

while(str(n[0])==self.nam):

self.readfile(n[1])

pos=pos+1

if(pos+1>=len(self.list2)):

break

n=self.list2[pos].split("|")

Step 4:Go to step 3

**3.5 Algorithm to Modify record**

Step 1: Search based on index.

Step 2: Store length of the line.

Step 2: Ask the user that which field want to get modify, based on that take input from user.

Step 3: pack the content with modified field and store the length.

Step 4: If new pack length is less than or equal to length of old line then open the file with r+ mode and write.

if(length2<=self.length):

file1.seek(int(self.position1))

file1.write(self.var)

print(self.var) 9

print("modify as been done")

file1.close()

return()

Step 5: Else read that line, store the length, extract first character and rewrite it by replacing with “\*”(considering that has been deleted) and also delete in primary index file and secondary index file.

first = self.a[0][0]

file1.seek(int(self.position1))

pos1=self.position1

file1.write(self.line1.replace(first,"\*")

………

first = self.b[0][0]

file2.seek(int(self.position2))

file2.write(self.line2.replace(first,"\*")

……….

Step 6: Append length and position to avail list and sort it.

length1=len(line)

st=""

st=str(length1)+"|"+str(position)

self.avllist1.append(st)

self.avllist1.sort()

Step 7: Check length of the packed record and search avail list if there is any space that is less than or equal to packed length then open the file in r+ mode and seek the position and write into the file and note the position.

length=len(pack1)

if len(self.avllist1) !=0:

for i in range (0,len(self.avllist1)):

ab=self.avllist1[i].split("|")

if(length<=int(ab[0])):

file1=open(r"C:\Users\Madhurani L\Desktop\stu.txt","r+") 10

file1.seek(int(ab[1]))

file1.write(pack1)

file1.write("\n")

Step 8: If avail list element space is greater than length of the packed record then remaining content will be marked as star(deleted) than size of the remaining content and position should be stored to avail list and sort the avail list.

length=len(l2)

a=l2.split("|")

first=a[0][0]

file1.write(l2.replace(first,"\*"))

var=""

var=var+len2+"|"+p2+"|"

self.avllist1.append(var)

self.avllist1.sort()

Step 9: Next open the primary index file in r+ mode, pack the primary index with position using delimiter and write into the primary index file.

File2=open(r"C:\Users\Madhurani L\Desktop\stu1.txt","r+")

pack2=pack2+self.orderid+"|"+position+"|"

file2.write(pack2)

file2.write("\n")

Step 10: Next open the secondary index file in r+ mode, pack the secondary index with position using delimiter and write into the secondary index file.

file3=open(r"C:\Users\Madhurani L\Desktop\stu2.txt","r+")

pack3=pack3+self.itemtype+"|"+position+"|"

file3.write(pack3)

file3.write("\n")

Step 11: Read the lines till line is empty then write the new packed content and store the position. 11

while self.line1:

self.position1=file1.tell()

self.line1=file1.readline()

file1.seek(int(self.position1))

file1.write(self.var)

print("modify as been done")

Step 12: Go to step 9 and step 10

**3.6 Algorithm to Delete the record**

Step 1: Search based on index.

Step 2: Read that line, and store length of the record, extract first character and rewrite it by replacing with “\*” (considering that as been deleted) and also delete in primary index file and secondary index file.

first = self.a[0][0]

file1.seek(int(self.position1))

pos1=self.position1

file1.write(self.line1.replace(first,"\*")

………

first = self.b[0][0]

file2.seek(int(self.position2))

file2.write(self.line2.replace(first,"\*"))

……….

Step 3: Append length and position to avail list and sort it.

length1=len(line)

st=""

st=str(length1)+"|"+str(position)

self.avllist1.append(st)

self.avllist1.sort() 12

***CHAPTER 4***

***RESULTS AND SNAPSHOTS***

**4.1 Inserting Record into File**

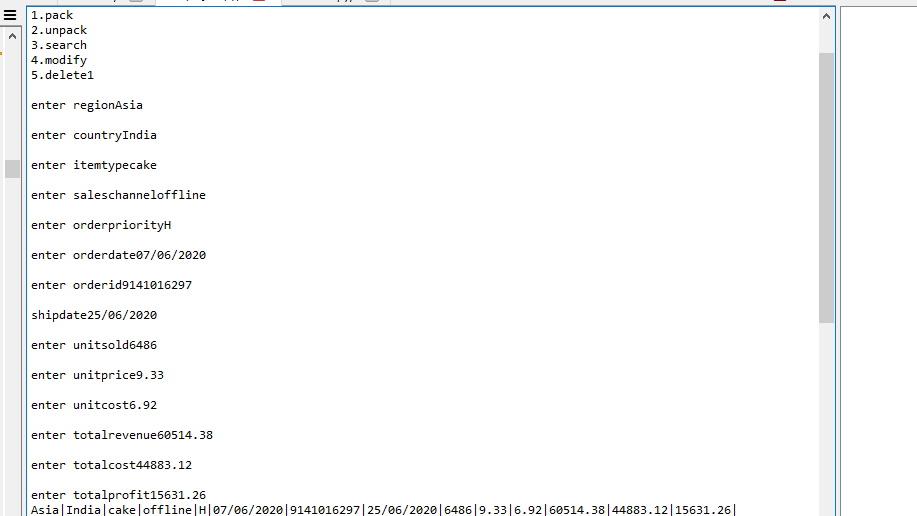
****

Fig 4.1.1

Variable length record will pack the user input using “|” as delimiter and it will open the file and write the record.



Fig 4.1.2

Record will store at end of the file and it will note the position.

13

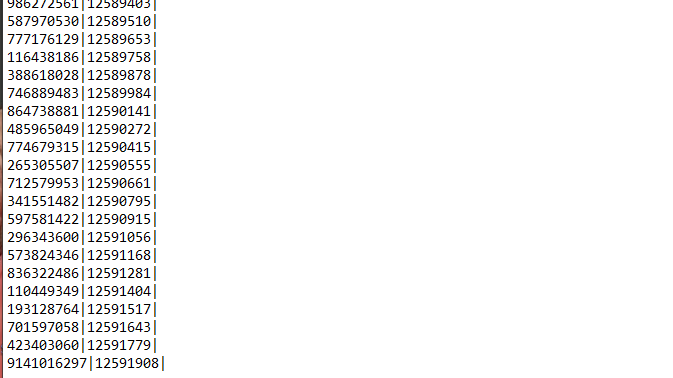
****

Fig 4.1.3

In figure 4.1.3 primary index is stored along with the position of the record. In figure 4.1.4 secondary index is stored along with the position of the record.

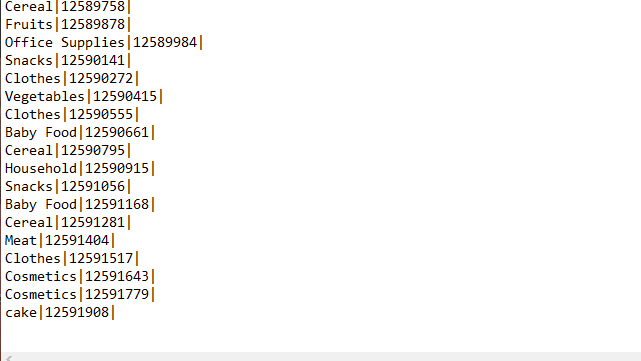
****

Fig 4.1.4

14

**4.2 Reading Record from File**

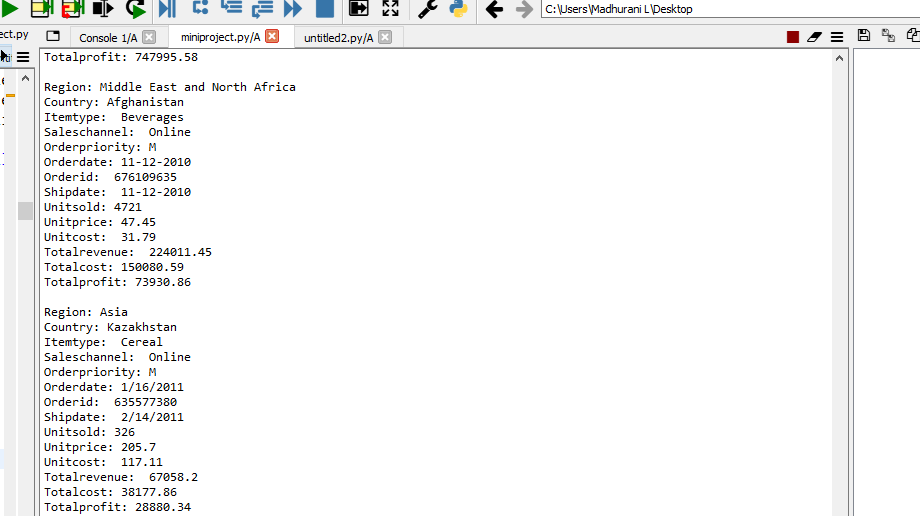
****

Fig 4.2.1

In figure 4.2.1, read all the record line by line and prints.

**4.3 Searching Record in the File**

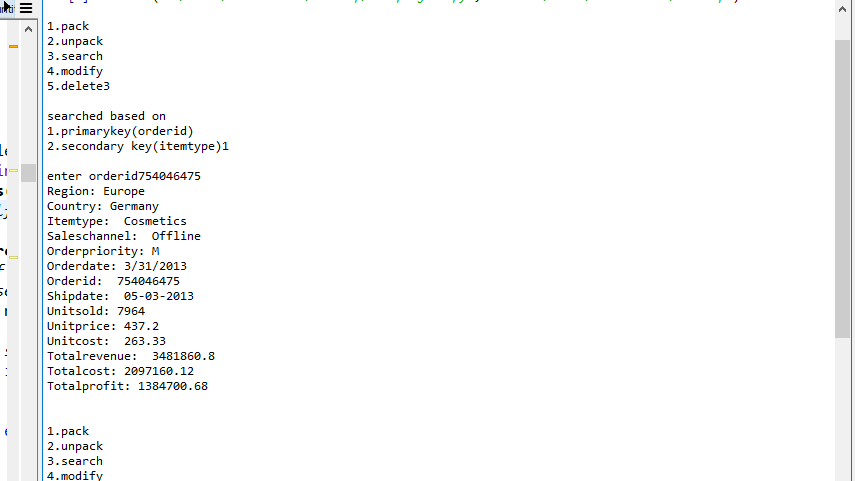


Fig 4.3.1

In figure 4.3.1, search based on primary index.

15

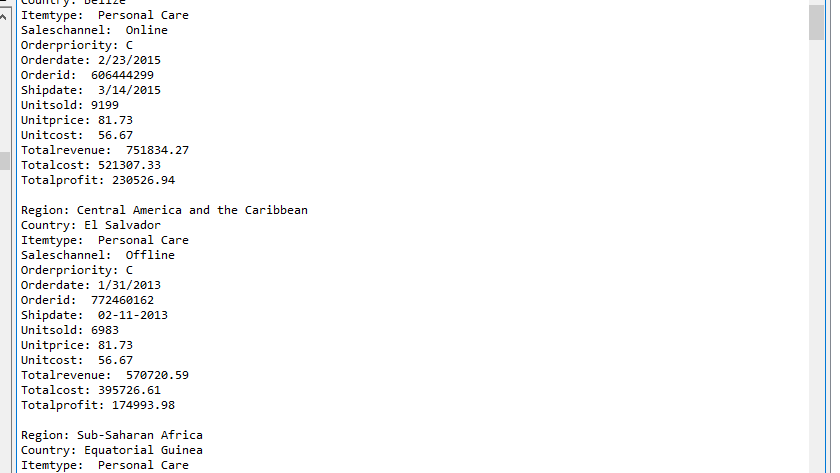
****

Fig 4.3.2

In figure 4.3.2, search based on secondary index. There will be having same itemtype for two or more records. All that records will be displayed.

**4.4 Modifying Record in the File**

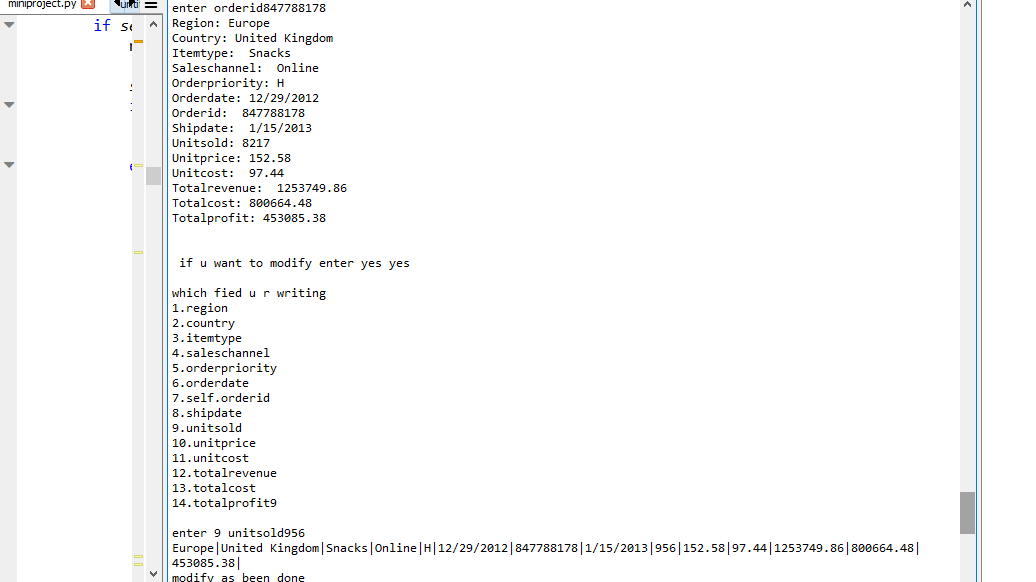


Fig 4.4.1

Modify based on primary index. it will ask input for modify field and pack with new input.

16

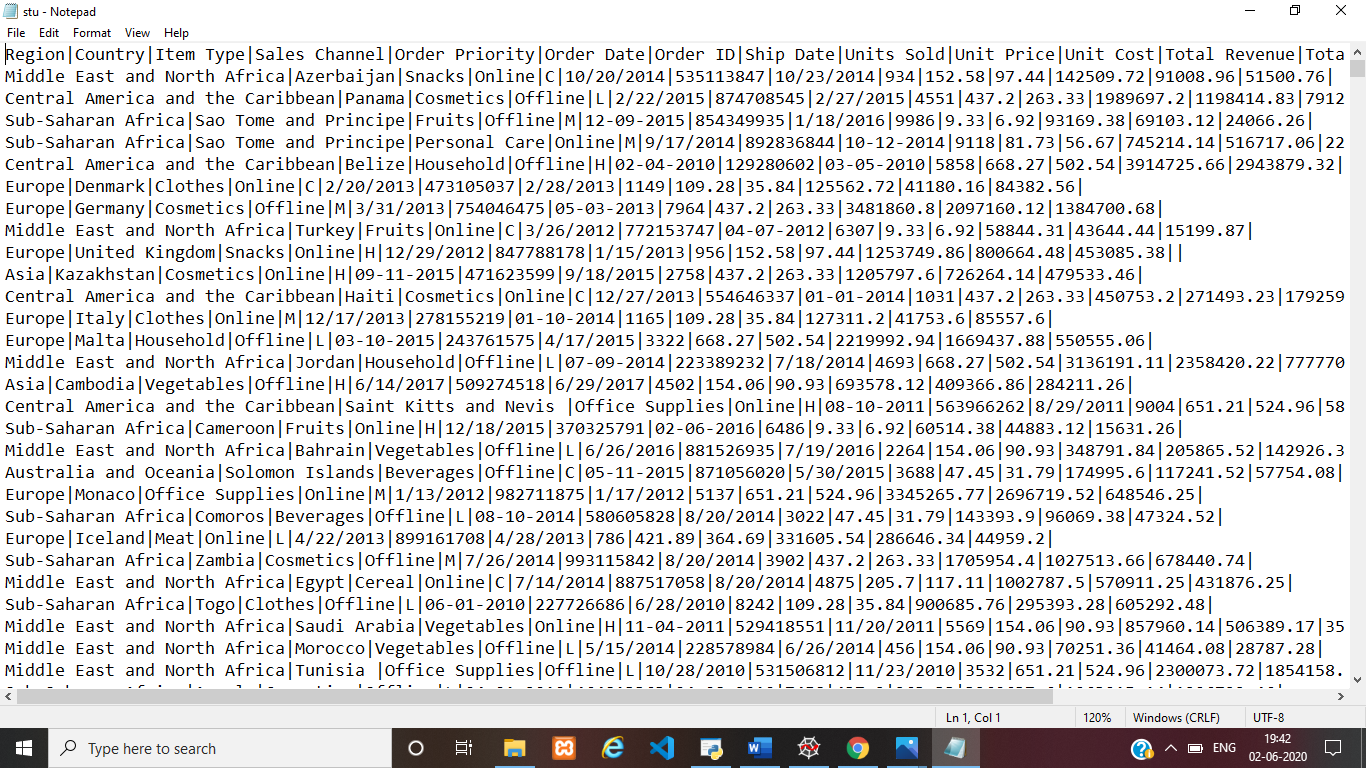
****

Fig 4.4.2

If new pack length is less than or equal to length of old line then open the file and write in the same place.

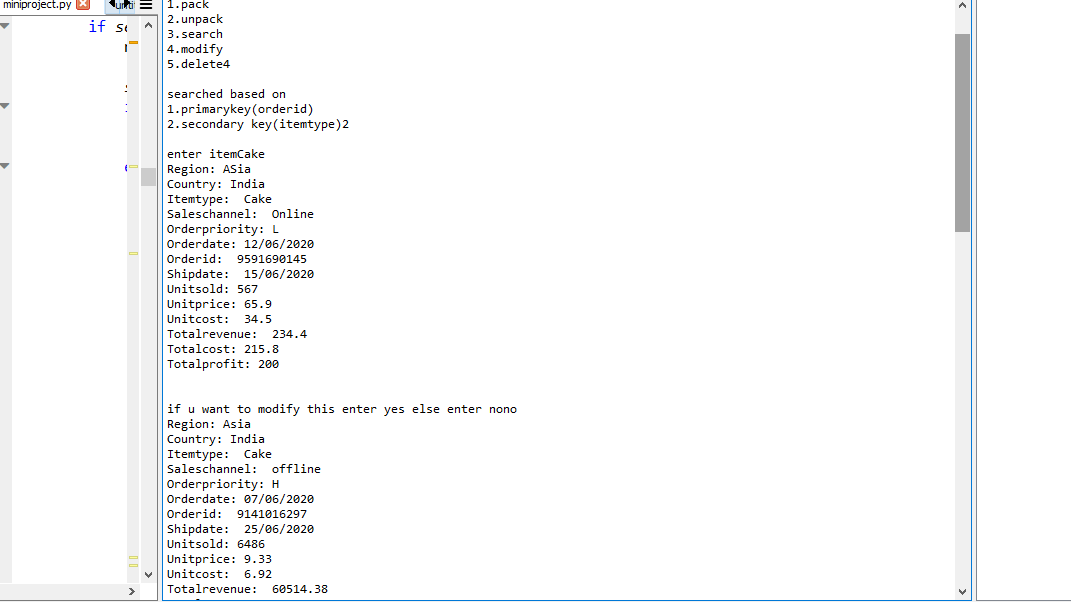


Fig 4.4.3

In figure 4.4.3 modify based on secondary index.

17

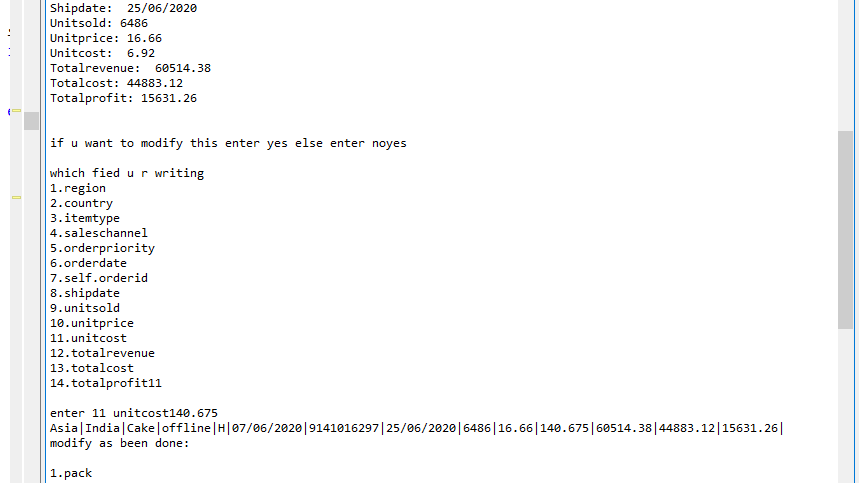


Fig 4.4.4

In figure 4.4.4, it will ask input to modify field and pack with new input.

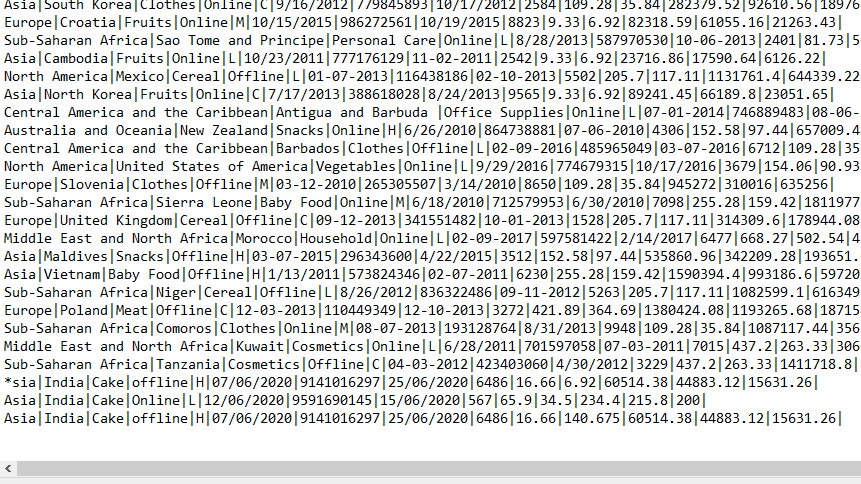


Fig 4.4.5

If new pack length is greater than length of old line then open the file and delete old line and write in the new line.

18

****

Fig 4.4.6

After deleting older line, it will be deleted in primary index and secondary index and it should be newly written in both primary index and secondary index with new address .



Fig 4.4.7

19

**4.5 Deleting Record from File**

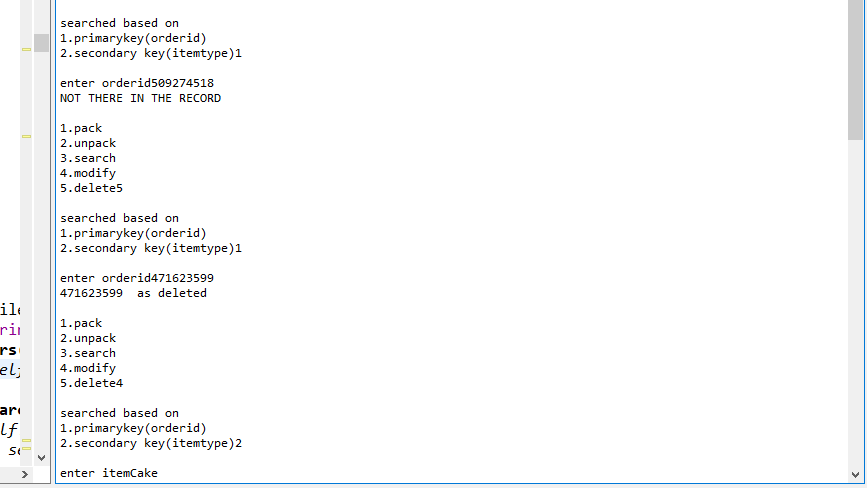
****

Fig 4.5.1

In figure 4.5.1 shows deleting using primary and secondary indexing.

****

Fig 4.5.2

Figure 4.5.2 shows deletion in the file. If line 1st character in “\*” then it is considered as deleted.

20

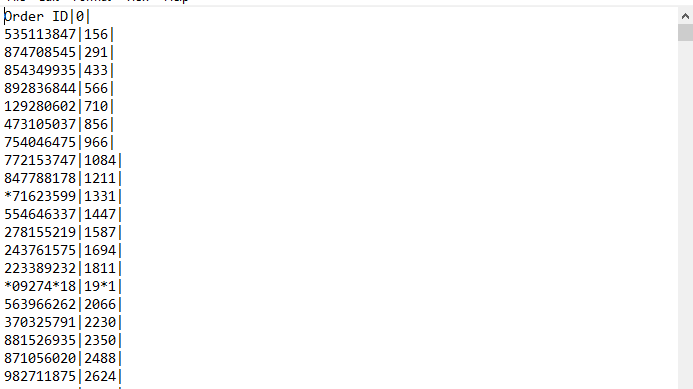
****

Fig 4.5.3

Figure 4.5.3 shows that deletion in primary index file. If line 1st character in “\*” then it is considered as deleted.

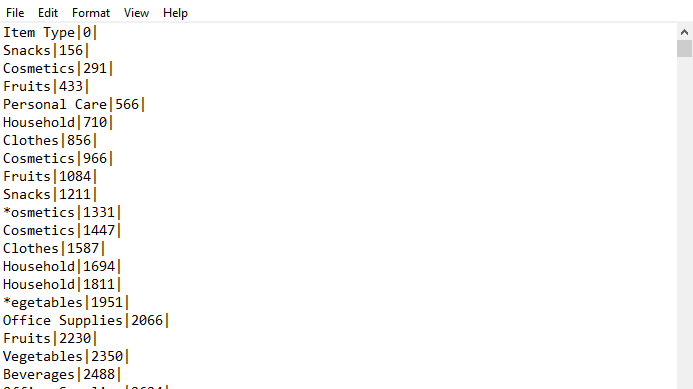
****

Fig 4.5.4

Figure 4.5.4 shows that deletion in secondary index file. If line 1st character in “\*” then it is considered as deleted.

21

**4.6 Inserting Record to Free Space**

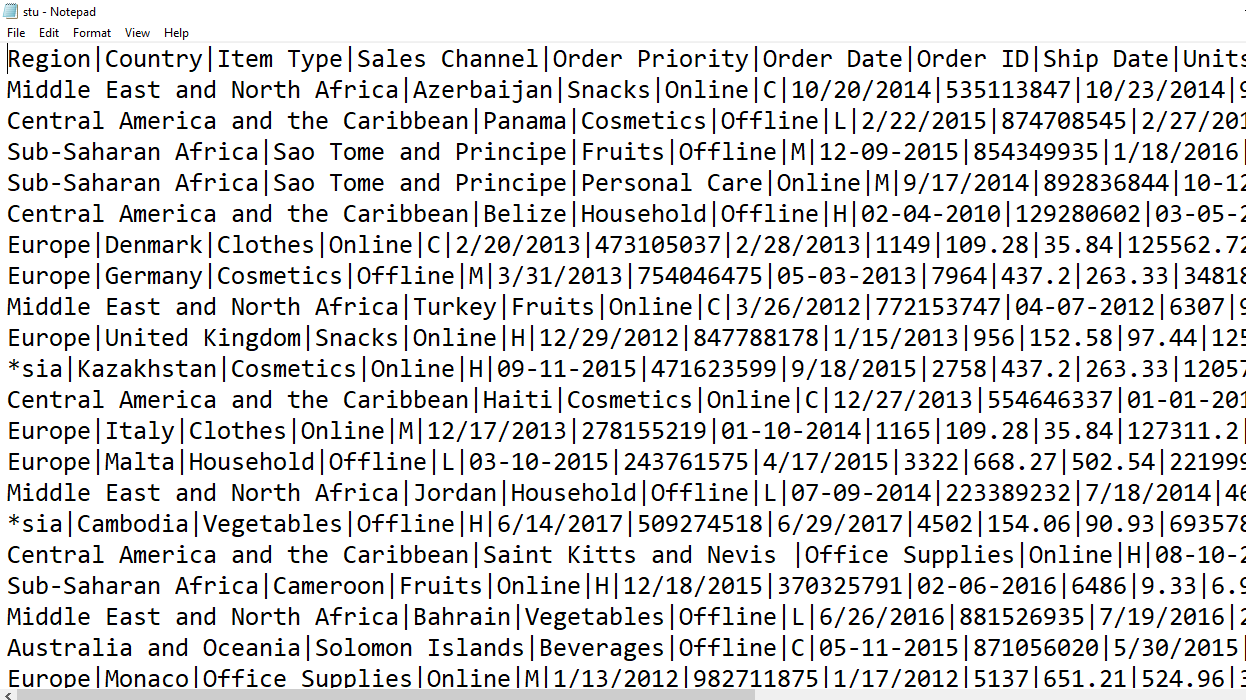
****

Fig 4.6.1

Figure 4.6.1 shows, before inserting a record into file. Under lined records are deleted and its size and position is stored in avail list.

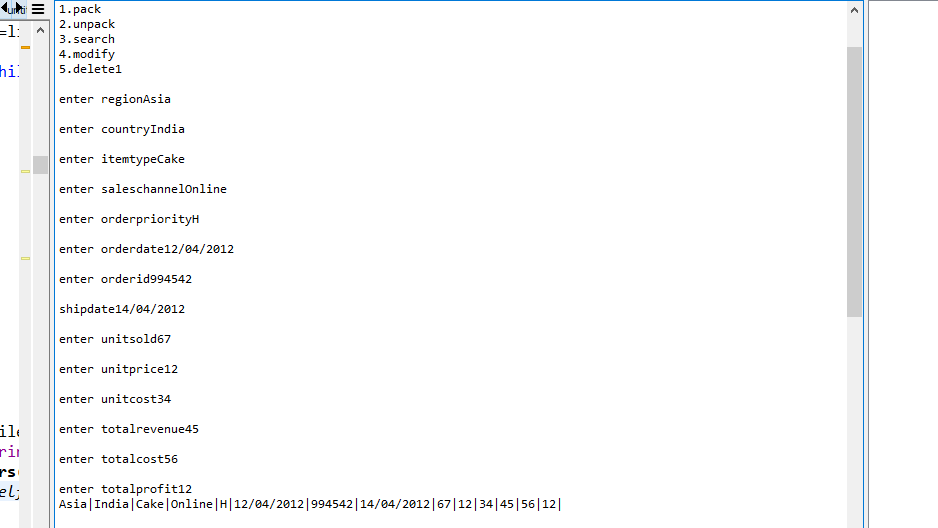


Fig 4.6.2

Taking input from user and packing record using “|” delimiter.

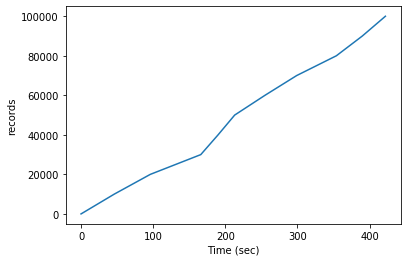
22



Fig 4.6.3

In figure 4.6.3, record is inserted in position of deleted or modified space according to its length. If deleted length is greater than new record length then free space will be added to avail list and it should be sorted, after inserting record into file, address should be stored with primary and secondary key in their respective file.

**4.7 Time Graph**

****

23

X-axis: time in second

Y-axis: record in the file

As number of record increases time also increases. If number of lines increase in a file, then reading the lines will increase the time

**2.8 Conclusion and Features**

**Conclusion**

Sales management system will help the sales industry to maintain record details of every product. The key concept is to minimize the amount of paper work and work is digitalized with easy in computerized system. The system will allow valid Admins to add sales details, deleting, modify.

It has been developed in php, html keeping in mind the specifications of the system. For designing the system. Overall, the project teaches us the essential skills like using system analysis and design techniques like time analysis in designing the system and Understanding the Simple Record Structure in file structure.

**Features**

System can hold good only for sales, it needs to be designed to all type of marketing. Managing sales record need to improvised. Additional scopes can be added. Utmost care and back-up procedures must be established to ensure 100% successful implementation of the computerized sales system. In case of system failure, the organization should be able to process the transaction with another organization or if the worst comes to the worst, it should be in a position to complete it manually.

**4.9 Reference**

1. Michael J. Folk, Bill Zoellick, Greg Riccardi: File Structures-An Object Oriented Approach with C++, 3rd Edition.
2. <https://stackoverflow.com/>

24