Database System Implementation

Project 2B - Implementing a Sorted File

Developed by –
Ghodkari Chowdary, Raghunatha Rao - UFID: 6218-1051
Mullapudi, Aseesh - UFID: 9175-1971

Aim of this project is to extend the DBFile to implement Sorted file capabilities.

We have changed the structure of existing code, we have written a virtual base class GenericDBFIle which contains the virtual functions. There DBFile now contains an internal variable myInternalVar. New classes Heap and SortedFile are written which extend the virtual base class GenericDBFile. The DBFile now creates either a heap or sortedfile object depending on the type of the file required and corresponding functions in their respective classes are invoked.

We have created new SortedFile class by writing the required functions. The job of the SortedFile within the system is to store and retrieve records over the sorted order specified.

Method Definitions:

- ***SortedFile::SortedFile () , SortedFile::SortedFile (OrderMaker *sortOrder, int runlen)***

Constructor used to create instances fields required by thre sortedFile and also to initialize the sortedOrder and the runlength.

- ***int SortedFile::Create (const char *f_path, fType f_type, void *startup)***

This method takes input f_path, f_type and creates a sorted bin file. It also creates a metadata file which stores the required the file type and sorted Order required by the file.

- ***void SortedFile::Load (Schema &f_schema, const char *loadpath)***

This method takes schema, loadpath as input and loads the records from the dataset into the input pipe which prepare it for the BigQ class.

- ***int SortedFile::Open (const char *f_path)***

This method takes the f_path as input and points to the sorted bin file

- ***void SortedFile::MoveFirst ()***

This method points to the first record in the SortedDB File using readBuffer

- ***void SortedFile::Add (Record &rec)***

This method is used to add the records to the sorted DBFile. A new bigq instance is created if the mode is read, else record is added to existing BigQ.

- ***int SortedFile::GetNext (Record &fetchme)***

This method returns the record from the Sorted DBFile instance using readBuffer same as the Heap DBFile.

- ***int SortedFile::GetNext (Record &fetchme, CNF &cnf, Record &literal)***

This method returns the record from the Sorted DBFile if it matches the filter CNF using binary search. This function is little complex prepares the query OrderMaker from the CNF checks compatible with sorted order of file and fetches records.

- ***int SortedFile::Close ()***

If mode is writing it changes mode to reading and call the merge function to merge the sorted file with existing records from the BigQ output pipe.

- ***void SortedFile::Merge()***

Function merges the sorted records coming from the output pipe of the BigQ with existing sortedDB File.

File **gtest.cc** written to unit test the Create, Open and Close function of the Sorted DBFile. Also checking if the bin File is sorted or not.

Instructions to Run the Code:

Compile and run test.cc:

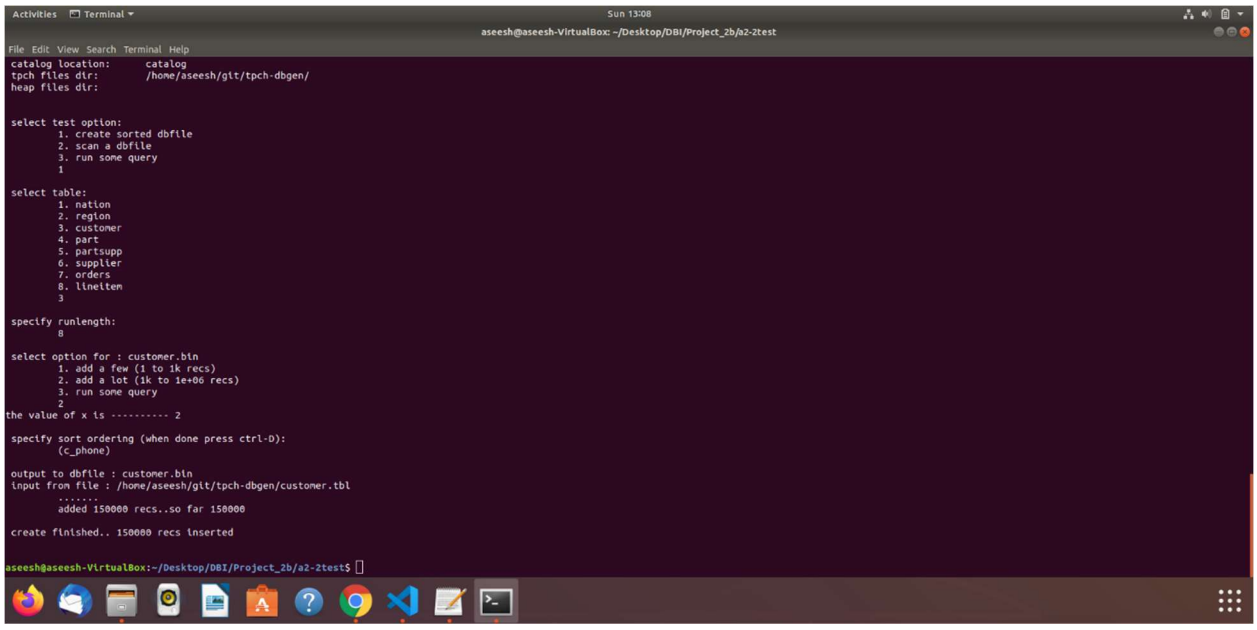
Ensure the directory contains .tbl files generated from the tpch-dbgen so that code has access to datasets. Ensure there is Catalog file in the root folder as well

1. make
2. ./test.out

Output of the required Queries on 1GB tpch-dbgen files

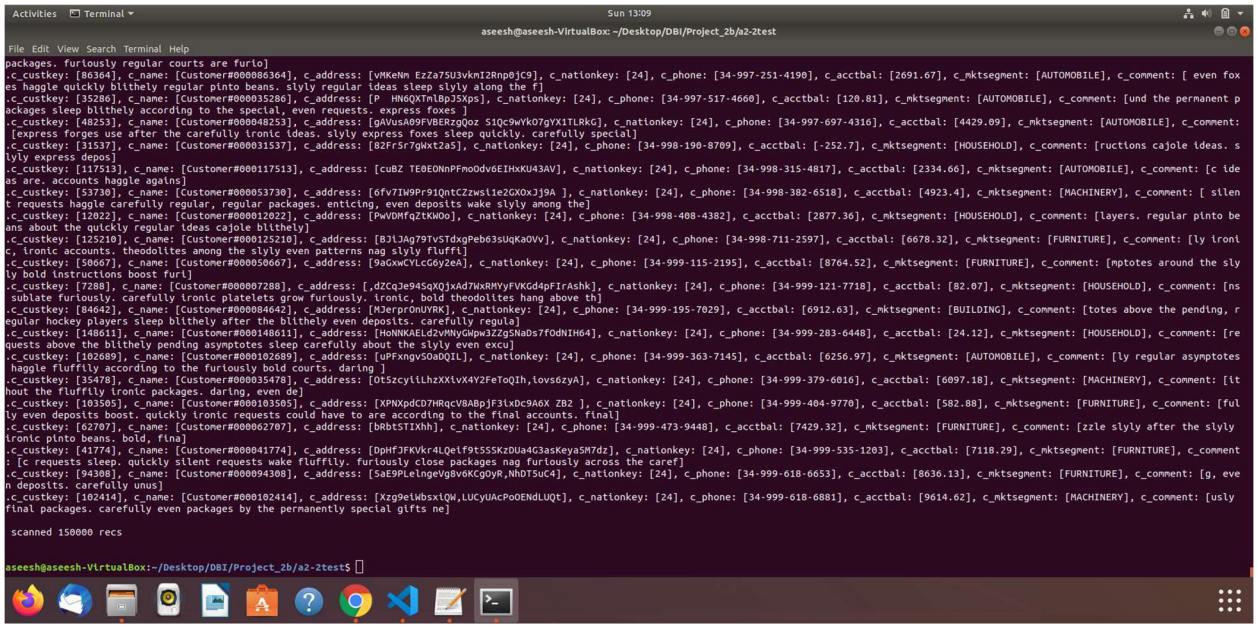
(Q1)

- create Sorted DBFile: 1
- table Customer: 3
- runlength: 8
- option to add records: 2
- sort Order: (c_phone)



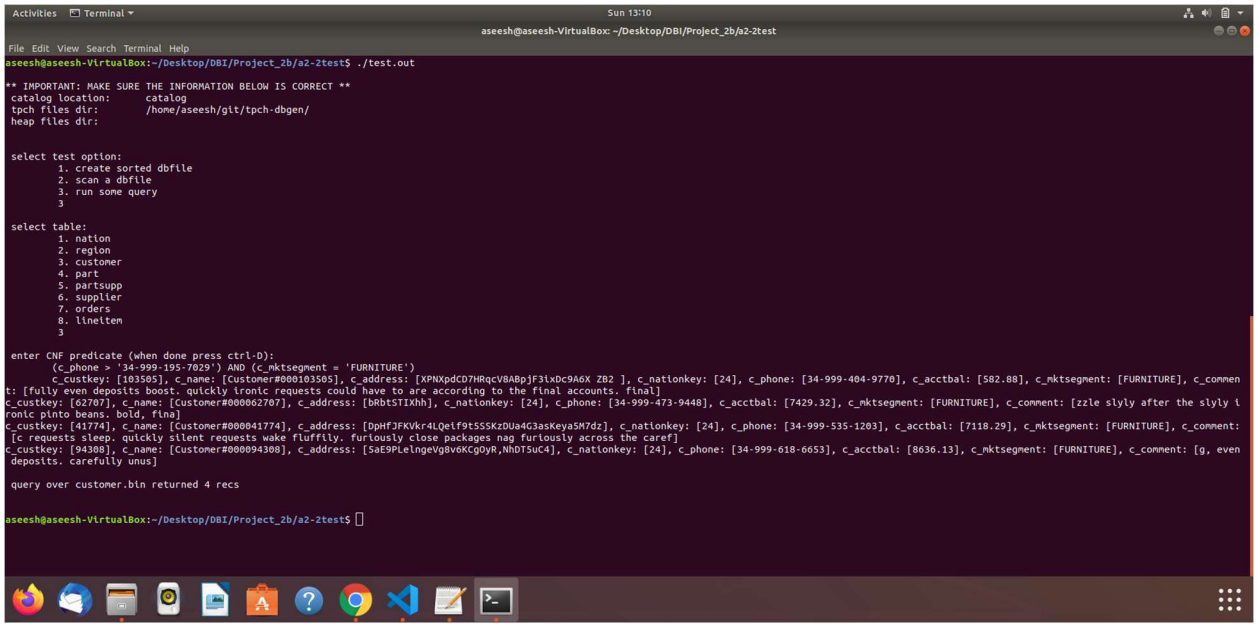
(Q2)

- Scan Sorted DBFile: 2
- table Customer: 3



(Q3)

- run Query over Sorted DBFile: 3
- table Customer: 3
- CNF literal: (c_phone > ’34-999-195-7029’) AND (c_mktsegment = ‘FURNITURE’)



Instructions to run gtests:

Compile and run code gtest.cc:

Ensure the directory contains the .tbl files generated from the tpch-dbgen so that code has access to datasets

1. make gtest
2. ./gtest

The screenshot of ./gtest is present on the next page