COP 6726:DataBase System Implementation

Project 2(Part 1):Sorted File Implementation

Venkateswarlu Tanneru
vtanneru@ufl.edu

Rohit Yerramsetty
rohityerramsetty@ufl.edu

Wednesday 3th March, 2022

1 Introduction:

DBFile class was implemented and extended in this assignment. The DBFile class in the database system is responsible for storing and retrieving records from the disk we have aditionally added the sorting purpose function to the DBFile which wil run over along with project 1. We've also used the Google Test Unit Testing Library to write unit test cases for,Every function of the DBFile Class has been implemented.For Project 2 we ran the entire project in Linux system on ubuntu using Virtual Machine.

2 Files Location:

- **MakeFile**: Make File runs the file structure in making the interpreted files in project.
- **tpch**/: Contains .tbl files which were generated using tpch data generator.
- **gtest**/: Google Test Unit testing library.
- **catalog**: Catalog file for the schema of .tbl files
- **test**/: Contains .bin files.

3 Basic Installations:

- make is installed using **sudo apt-get install make**, g++ is installed using **sudo apt-get install g++** and also bison&flex using **sudo apt-get install flex bison**
- Some additional installations like **gtest** and also **dos2unix**.

4 Commands To Run the Program:

- **make clean** to clean the entire build data base systems.
- **make** to build the files interpreted on make file.
- Type **make test.out** to compile the provided test driver.
- Type **./test.out** and then follow the on-screen directions to start the test driver.
- Fill in any CNF from the result section below for the scan and filter operations. After you've finished creating the CNF, press ENTER and then Ctrl+D.
- The unit tests should run and the test results should be shown.
- Additionally added runtestcases file which can which helps in running test cases scripts using command **./runtestcases.sh**

5 Explanation for classes and methods:

5.1 Class Execute:

The class here represents the "Run" which consists of a sequence of pages. In the first phase, records are gathered into a Run, and the next phase does merge-sorting for runs, to output sorted records. It can be noticed that the Run is just an abstract of a page that is stored in a file, therefore the file is the actual disk base for storing records to be sorted.

5.1.1 Methods and important variables of Execute Function:

5.1.2 Execute::Execute(File* file, int startPage, int runLength)

Construct method for run, a file pointer and a startPage index are used which indicate the start page for run, and also define the length of run(number of pages stored).

5.1.3 Record *bestRecord:

The variable is used to store the top record in Execute. The bestRecord is the smallest element at that moment in the Run, since the run is sorted in a given order.

5.1.4 int Run::UpdatebestRecord()

This method is used to pop the current top record, and get the next record to be the top record from Execute. Return 0 if there are no more records in this Run.

5.2 Class CompareRec:

This class acts as a customer comparator for records. It compares records using the given OrderMaker.

5.2.1 ComparerRec::CompareRec(OrderMaker* orderMaker):

This is the construct method for CompareRec, OrderMaker is it's parameter, which is then used for comparing.

5.2.2 bool CompareRec::operator () (Record* left, Record* right):

Customer comparing operator, it compares based on the given OrderMaker.

5.3 Class CompareRun:

This class acts as a customer comparator for runs. It compares runs using the given OrderMaker.

5.3.1 CompareRun::CompareRun(OrderMaker* orderMaker):

This is the construct method for CompareRun, OrderMaker is it's parameter, which is then used for comparing.

5.3.2 bool CompareRun::operator () (Execute* left, Execute* right);:

Customer comparing operator, it compares based on the given OrderMaker.

5.4 Struct WorkerArg:

Struct used for transferring parameters from BigQ to worker's functions.

5.5 Class Big_Q:

Base class for sorting records. Only has a construct method to take inputs.

5.5.1 Big_Q :: Big_Q (Pipe &in, Pipe &out, OrderMaker &sortorder, int runlen):

This is the construct method for BigQ, it builds a WorkerArg which is used in worker's function. It will then create a worker thread, and worker thread handles the rest.

5.6 Work thread Functions:

The work thread retrieves all records from the input pipe, sorts them and then puts sorted records into the output pipe. Two functions(without class) used for work thread are as follows:

5.6.1 void* work_func(void* arg):

Main method executed by worker. Worker retrieves records from the input pipe, sorts these records into runs, puts all these runs into a priority queue, and gets the sorted records from priority queue to the output pipe .

5.6.2 void* record_Queue(priority_queue<Record*, vector<Record*>, CompareRec>& recordQueue, priority_queue<Run*, vector<Run*>, CompareRun>& runQueue, File& file, Page& bufferPage, int& pageIndex):

This function is used to take sequences of pages of records, and then construct a run to hold such records, and finally put the runs into a priority queue.

6 Output1.txt Results:

The output of runTestCases.sh is shown below. They are arranged in the following order: tc1.txt, tc2.txt, tc3.txt, and tc4.txt. I modified OrderMaker somewhat to ensure that the results are actually sorted; otherwise, the data would not be sorted if we used the original OrderMaker.

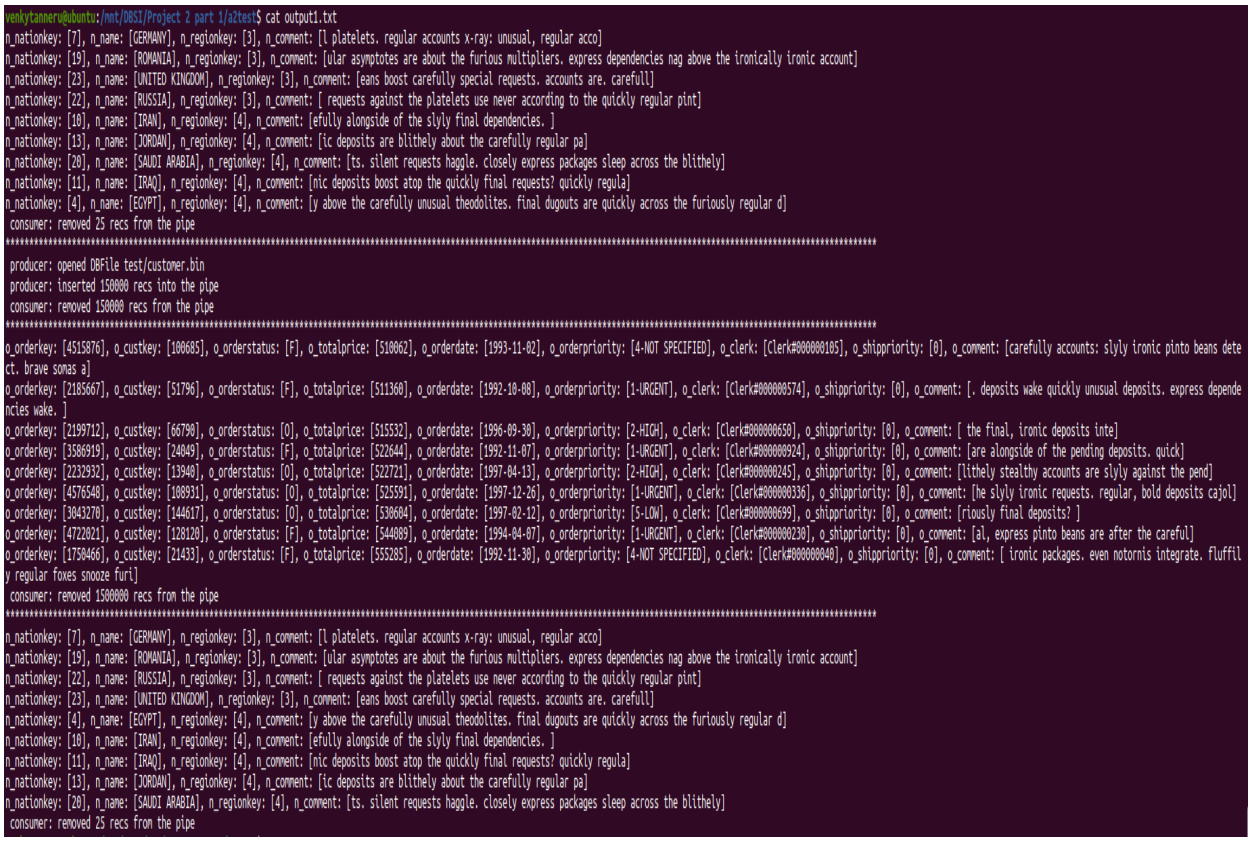

Figure 1: Output File Results

7 Gtest:

To run the gtest, first make sure you enter the root directory of the project, where the gtest.cpp file is contained.

7.1 Commands To Run the Gtest Program:

- Type **make clean** and then **make gtest** to compile Google Test.
- Type **./gtest.out** to run the unit tests.
- The unit tests should run and the gtest results should be shown.

7.2 Gtest results:

There are four tests in use. Testing is carried out on top-of-the-line records. Testing CompareRec on successfully comparing records CompareRun on successfully comparing runs testing non-class method record_Queue on if the records are properly placed into run A screenshot of the gtest is shown below.

```
venkytanneru@ubuntu:/mnt/DBSI/Project 2 part 1/a2test$ ./gtest.out
[==========] Running 4 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 4 tests from BigQTest
[ RUN      ] BigQTest.UpdateTopRecordForRunTest
[       OK ] BigQTest.UpdateTopRecordForRunTest (21 ms)
[ RUN      ] BigQTest.RecordComparerTest
[       OK ] BigQTest.RecordComparerTest (17 ms)
[ RUN      ] BigQTest.RunComparerTest
[       OK ] BigQTest.RunComparerTest (14 ms)
[ RUN      ] BigQTest.recordToRunTest
[       OK ] BigQTest.recordToRunTest (7 ms)
[----------] 4 tests from BigQTest (59 ms total)

[----------] Global test environment tear-down
[==========] 4 tests from 1 test suite ran. (59 ms total)
[  PASSED  ] 4 tests.
```

Figure 2: Gtest Results