PROJECT REPORT

RELATIONAL OPERATORS

Name: Gloria Gilbert Nazareth
UFID: 8221-8035
GatorLink: gnazareth@ufl.edu

Steps to compile and execute:

1. make ./a2test.out : this command builds the files required by a2-test.cc and creates the required object files
2. ./a2test.out : This command will create all the bin files and will save it in the directory mentioned in test.cat
3. make ./test.out : this command builds the program and creates the required object files
4. ./test.out <input value 1-6>: this will execute the program, and ask for user input in order to perform required functionality. Make sure a2test.out is executed before running it.
5. ./runTestCases.sh : This command will execute all the test cases mentioned in the script file and will write the output to output1.txt
6. make ./gtest.out : this command will build the program and create the required object files
7. ./gtest.out : this command will execute the gtests. Make sure a2test.out is executed before running gtests.

Description of the Implemented Function:

The RelOp.cc class has the following functions:

1. void <ClassName>:: Run(<arguments)
   This function maps the arguments to the attributes of the class. It then creates a pthread to perform the required operation. Each of the class in RelOp.cc has a Run function which performs the same task.

2. void <ClassName>::WaitUntilDone ()
   This function joins the thread defined in the respective class, i.e it waits for the thread to complete its task. Each class defined in RelOp.cc has a WaitUntilDone function which performs the same task.

3. void <ClassName>:: Use_n_Pages (int runlen)
   This function assigns the value runlen to the runLength attribute defined in the respective class. All the classes in the RelOp.cc has a Use_n_Pages function which performs the same task.

4. void* SelectFile::performOperation()
   This function reads the data from the input file given to it, checks if it matches the CNF given. If it does, it inserts the record to the outPipe. If the CNF is null, it will insert all the records to the outPipe.

5. void* SelectPipe::performOperation()
   This function reads the data from the input pipe given to it, checks if it matches the CNF given. If it does, it inserts the record to the outPipe. If the CNF is null, it will insert all the records to the outPipe.

6. void* Project::performOperation()
   In this function records retrieved from the input pipe are projected to create new records containing columns which are mentioned in keepMe array. This new record with new schema is added to the outPipe.

7. void* DuplicateRemoval::performOperation()
   This function initially sorts all the records in its inPipe using BigQ. It then retrieves the records and compares adjacent records based on the schema given. If the record contains duplicate values, it inserts only one of them in the outPipe. Thus, the outPipe contains only unique values for schema value.

8. void* Sum::performOperation()
   This function retrieves the records from the inPipe and checks the data type of the column whose sum is to be calculated (whether int or double). It then adds the value to the total sum. Post this it creates a record using Compose function and adds the new record to the outPipe.

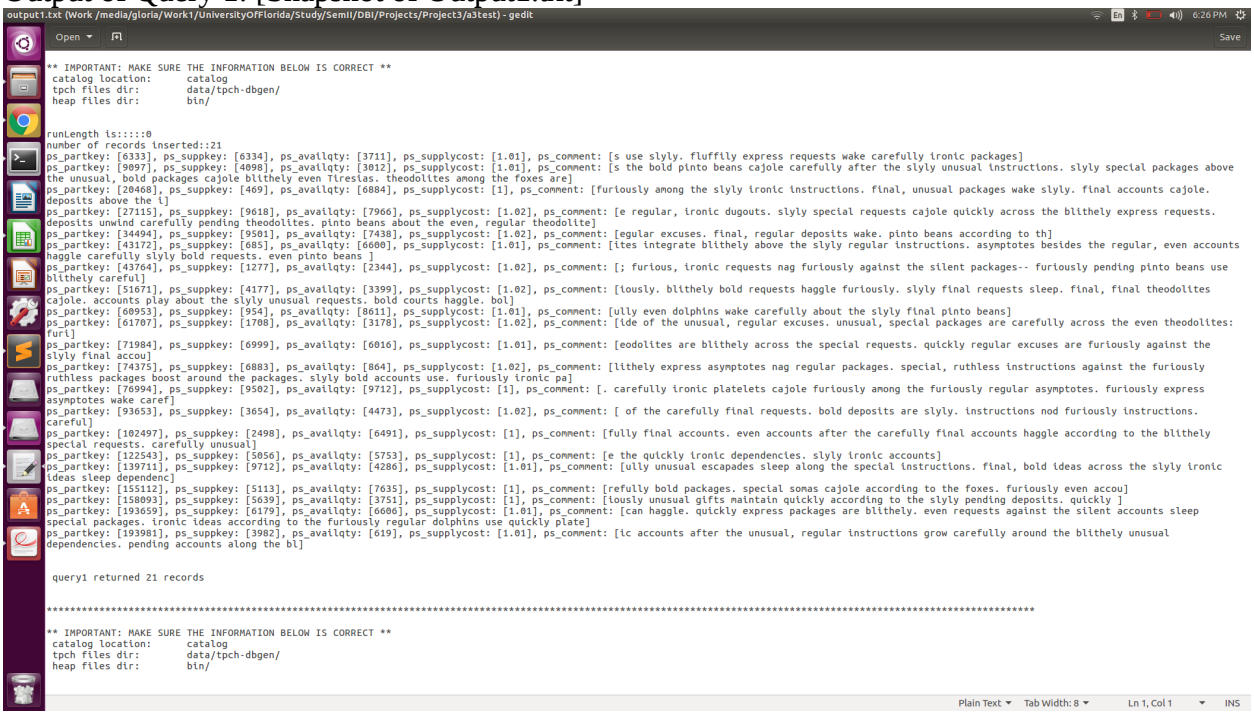
9. void* GroupBy::performOperation()
   This function performs grouping of records based on the grouping attributes and also calculates the sum for each group. It initially sorts all the records in the inPipe using BigQ. Post this it compares adjacent records and checks if they belong to the same group. Sum is calculated using the logic in point 8 for each group, and the new record for each group is composed of grouping attribute along with sum which is added to the outPipe.

10. void* Join::performOperation()
    This function joins two tables based on CNF given. Initially both the tables are sorted using BigQ. If CNF is given, we retrieve the records from the left and right table, and check if they match, else based on the values of the comparison we retrieve values from respective tables. Once the right and left have matched, we check if there are more records in right that match the same left record, and vice versa and add it in left and right vectors. Each record in each of the vector is merged with the other. If the CNF is not given all the records of the right table merged with that of the left (like cross product).

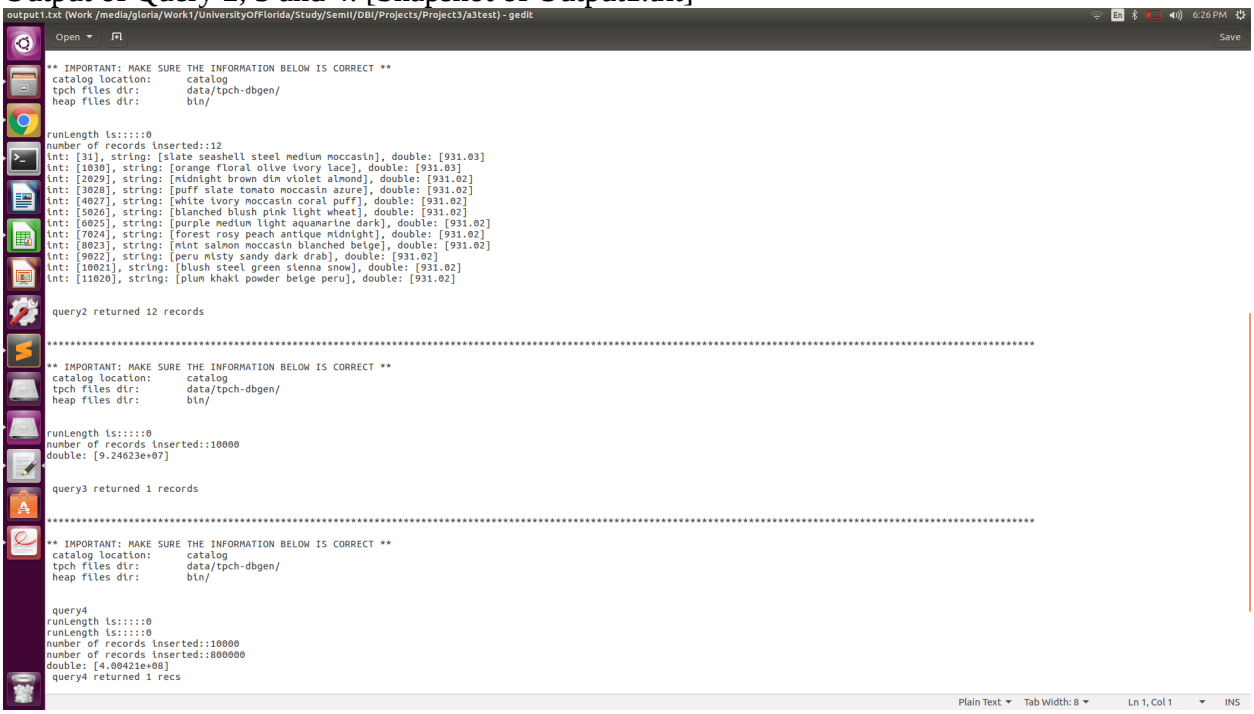
11. void* WriteOut::performOperation()
    This function retrieves the data from the input file and writes it to the outFile in the format of the data given in tbl files.

Output on 1GB data:

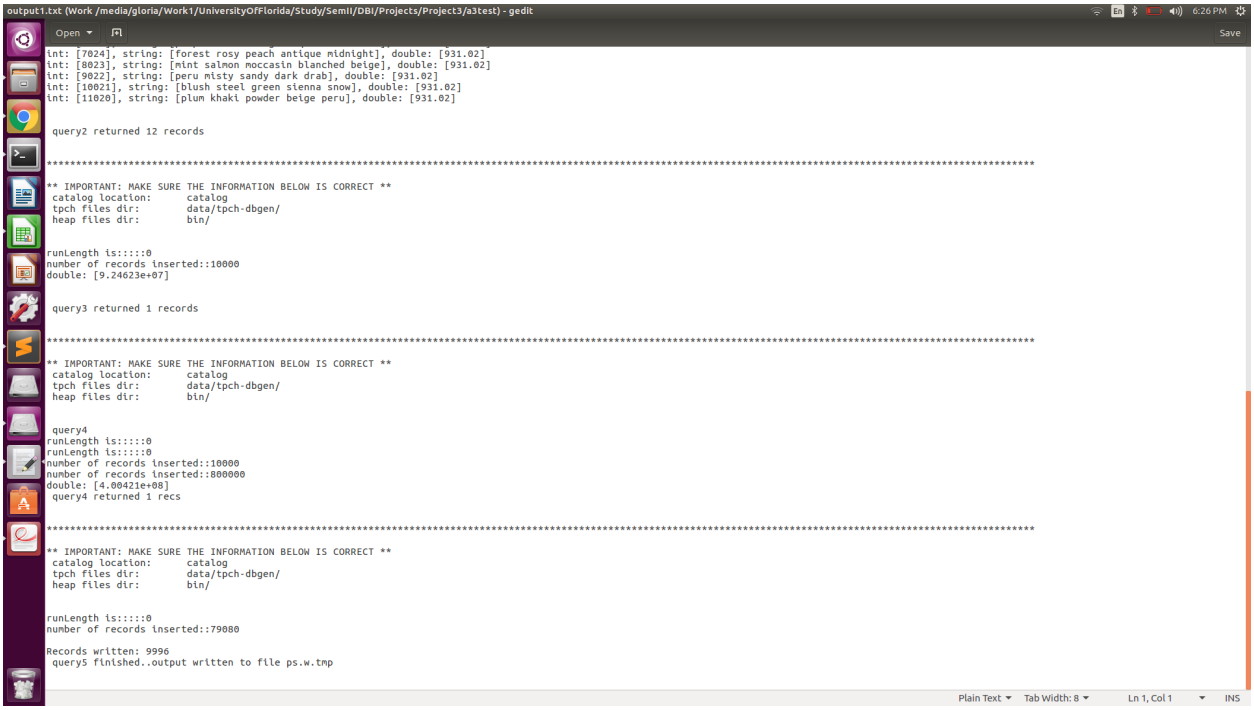
1. Output of Query 1: [Snapshot of Output1.txt]



2. Output of Query 2, 3 and 4: [Snapshot of Output1.txt]



3. Output of Query 5: [Snapshot of Output1.txt]



Gtest Results:

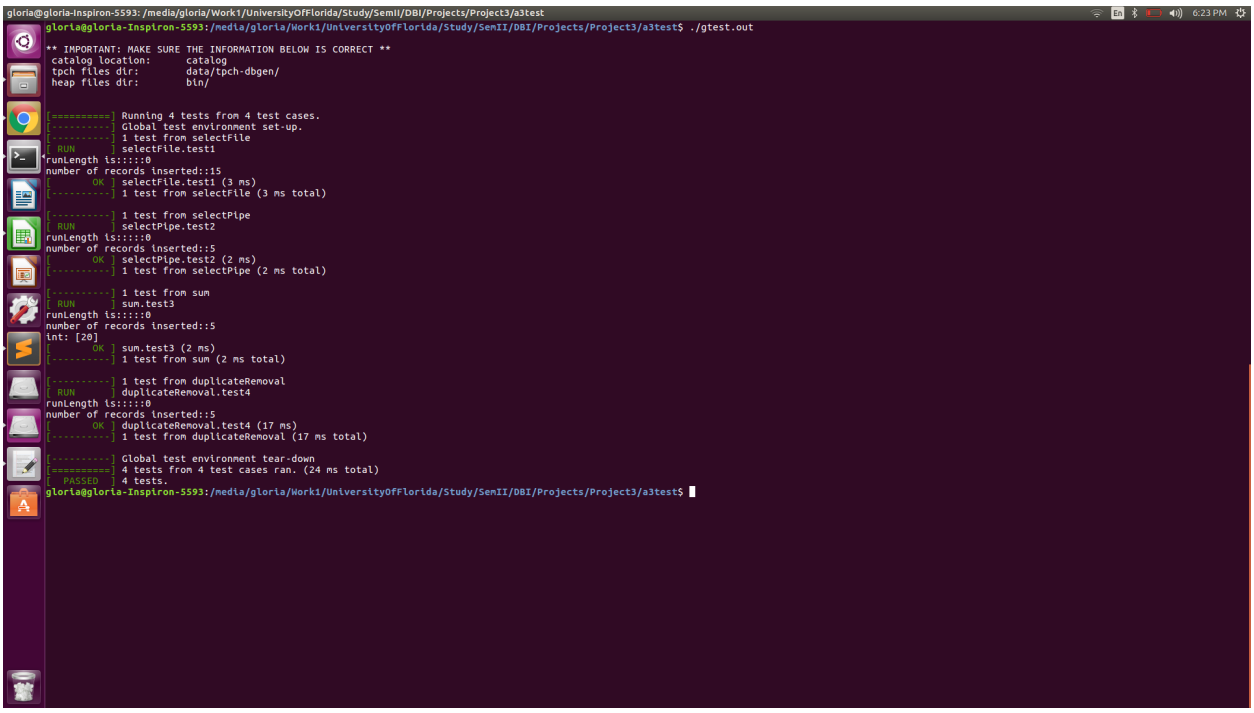
1. TEST(selectFile,test1)
   This test selects data from the nation bin file and filters for records whose region key is <3 and returns output(_o). The count of records in the output pipe is matched in the assert.

2. TEST(selectPipe,test2)
   This test selects data from the nation bin file puts it in pipe then filters for records whose region key is >3 and returns output(_o). The count of records in the output is matched in the assert.

3. TEST(sum,test3)
   This test selects data from the nation bin file then filters for records whose region key is >3. Post this it calculates the sum region keys of all the filtered records and returns output(_out). If a single sum record is returned as expected, the test is pass.

4. TEST(duplicateRemoval,test4)
   This test case checks for unique values of n_regionkey present in the table and adds it in the __ps pipe. The count of the records is matched in the assert function.

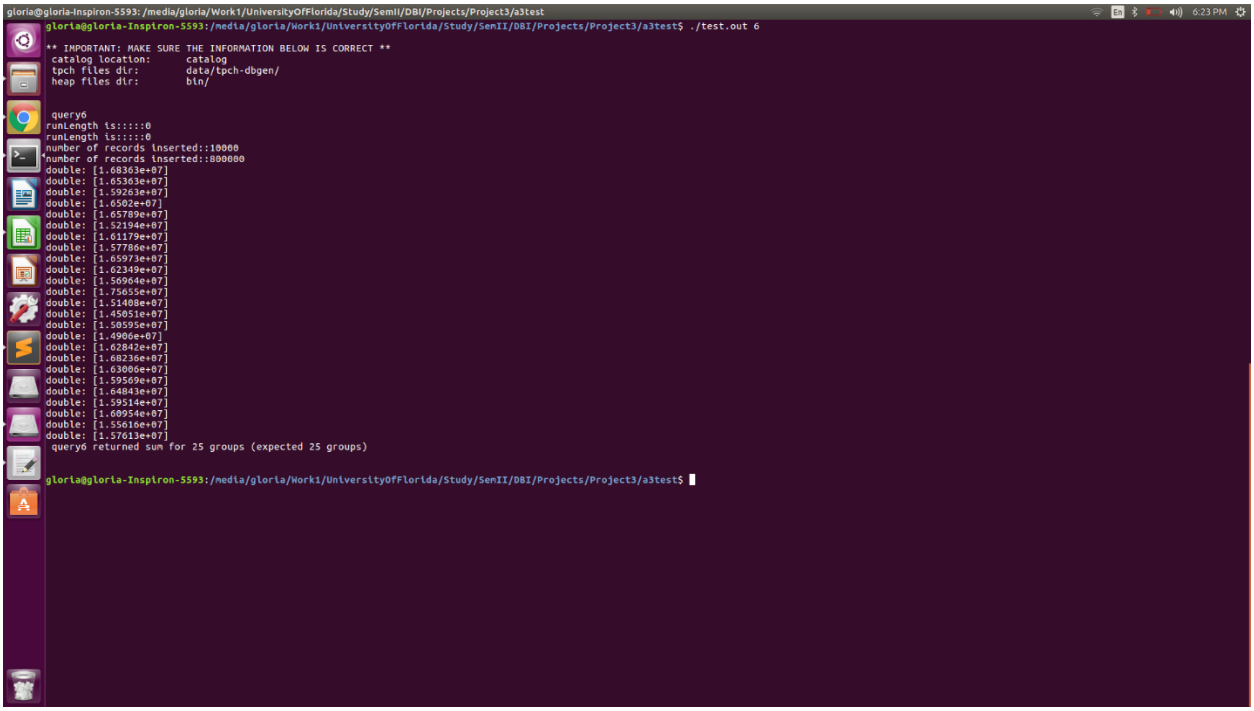
   Screenshot of Gtest.out



Issues in Code:

Could not run the q6 function given in test.cc as the OrderMaker required for groupBy was not defined. Defined the OrderMaker adding the below mentioned lines of code and executed it successfully.

```
OrderMaker grp_order;
grp_order.numAtts=1;
int n = join_sch.GetNumAtts();
Attribute *myAtts=join_sch.GetAtts();
for(int i=0;i<n;i++)
{
    if(i==3)
    {
        grp_order.whichAtts[0]=i;
        grp_order.whichTypes[0]=Int;
    }
}
```


gloria@gloria-Inspiron-5593: /media/gloria/Work1/UniversityOfFlorida/Study/SemII/DBI/Projects/Project3/a3test
6:23 PM
gloria@gloria-Inspiron-5593:/media/gloria/Work1/UniversityOfFlorida/Study/SemII/DBI/Projects/Project3/a3test$ ./test.out 6
** IMPORTANT: MAKE SURE THE INFORMATION BELOW IS CORRECT **
catalog location: catalog
tpch files dir: data/tpch-dbgen/
heap files dir: bin/
query6
runLength is:::::0
runLength is:::::0
number of records inserted::10000
number of records inserted::800000
double: [1.68363e+07]
double: [1.65363e+07]
double: [1.59263e+07]
double: [1.6502e+07]
double: [1.65789e+07]
double: [1.52194e+07]
double: [1.61179e+07]
double: [1.57786e+07]
double: [1.65973e+07]
double: [1.62349e+07]
double: [1.56964e+07]
double: [1.75655e+07]
double: [1.51408e+07]
double: [1.45051e+07]
double: [1.58595e+07]
double: [1.4906e+07]
double: [1.62842e+07]
double: [1.68236e+07]
double: [1.63006e+07]
double: [1.59569e+07]
double: [1.64843e+07]
double: [1.59514e+07]
double: [1.60954e+07]
double: [1.55616e+07]
double: [1.57613e+07]
query6 returned sum for 25 groups (expected 25 groups)
gloria@gloria-Inspiron-5593:/media/gloria/Work1/UniversityOfFlorida/Study/SemII/DBI/Projects/Project3/a3test$