Assignment 3
Relational Operations

Group Members:

1. Dhiraj Mahesh Paryani, UFID: 1692 1261
2. Yogesh Laxman, UFID: 9451 2517

Instructions for compiling and running the code:

1. **Extracting folders:**
    a. Extract the contents of the folder. DhirajMaheshParyani_YogeshLaxman_p3.zip
    b. Open the terminal and navigate to the "Project" folder which is inside the extracted folder.

2. **Update test.cat:**
    a. Update catalog_path, dbfile_dir, & tpch_dir paths in test.cat file.
    b. The first line in test.cat should contain catalog_path.
    c. The second line in test.cat should contain dbfile_dir.
    d. The Third line in test.cat should contain tpch_dir.

3. **Generate DB files:**
    Generate DB (.bin) files using previous assignments' drivers to run this assignment. DB file can be a heap db file or sorted DB file.
    a. To Generate Heap DB file:
        i. Update catalog_path, dbfile_dir, & tpch_dir paths in a1-test.cc file.
        ii. Run "make a1test.out".
        iii. Run "./a1test.out" and give corresponding inputs to generate heap DB file.

    b. To Generate Sorted DB file:
        i. Update catalog_path, dbfile_dir, & tpch_dir paths in a2-test.h file.
        ii. Run "make a2test.out".
        iii. Run "./a2test.out" and give corresponding inputs to generate sorted DB file.

4. **Running the tests:**
    a. Run "make test.out".
    b. To run individual test case, run "./test.out {x}". Where {x} should be replaced by test case number i.e any number from 1 to 6.

c. Run "./runTestCases.sh" to run test cases from 1 to 5, this will generate output in a file named "output1.txt".

Brief explanation of implementation:

This assignment deals with the implementation of 8 relational operators. These operations are encapsulated within classes which are derived from the RelationalOp class.

Class RelationalOp:

- Data members:
  Following are protected data members, so that derived classes can use these members.
  1. **pthread_t thread:** This thread will be used by derived classes to do operation or apply relational operator on data asynchronously.
  2. **int runLength:** This stores how much internal memory the operation can use in a number of pages. The default is 16 pages.

- Methods:
  Following are the public methods, hence these methods are executed when corresponding methods are called with the derived class' object.
  1. **void WaitUntilDone():** This method will block the caller until thread has finished the execution. This means the caller will wait until current relational operation is done.
  2. **void Use_n_Pages(int n):** This updates the runLength member of the operator.

All derived relational operators have a Run method with different input parameters, which will be called to execute the corresponding relational operation. Basic flow of all the Run methods is same, it does the following:

1. Creates corresponding structure which will be used by a thread.
2. Loads that structure with the input of Run.
3. Spawns the thread with the corresponding function and a structure. Corresponding function will be called inside of spawned thread which will execute that relational operator with the help of passed structure.

Following is the brief details of implementation of each relational operator's corresponding function.

1. **SelectPipeThreadMethod:** This method extracts records from the input pipe in a loop and pushes the record into the output pipe if the record satisfies the cnf. It checks cnf by calling comparison engine's Compare method.

2. **SelectFileThreadMethod:** This method extracts records from the input file and pushes the record into the output pipe if the record satisfies the cnf. It checks cnf by calling the input file's GetNext method.

3. **ProjectThreadMethod:** This method extracts records from the input pipe. Each record is then passed to Record's Project method with required parameters, which will modify the record and keep only projected attributes. Finally, the projected record is pushed into the output pipe.

4. **JoinThreadMethod:** This method starts with constructing an order maker of left and right input pipe's records. If any of the order makers is empty, it calls NestedBlockJoin method, which performs Nested block join on records. Otherwise, it sorts left input pipe's records and right input pipe's records using BigQ. Then output of BigQs is passed to JoinUsingSortMerge method which applies a sort merge algorithm to join the records.

5. **DuplicateRemovalThreadMethod:** This method starts with calling order maker's constructor for constructing an order maker from the schema. Then BigQ is used to sort the records of the input pipe according to the constructed order maker. After which, it extracts sorted records from BigQ and pushes only unique records into the output pipe. It finds unique records by comparing each extracted record with the previously extracted record, if both the records are the same, it means we have duplicate and it doesn't push any record into the output pipe. If both the records are not the same, then the previous record is pushed into the output pipe. At the end it always pushes the last record into the output pipe.

6. **SumThreadMethod:** This method extracts records from the input pipe and applies Function on each extracted record and updates sum. After all the records are extracted, sum is ready. Finally, it pushes the sum record into the output pipe. It creates sum record by using ComposeRecord method of Record method by passing string equivalent of sum appended with pipe character ("|").

7. **GroupByThreadMethod:** This method is a combination of DuplicateRemoval and Sum Operator. This method starts with sorting the input records by using BigQ according to the group by order maker i.e attributes on which group by should be done. Then sorted records from BigQ's output pipe are grouped together and sum is updated for each group. Record for each group is createx with the sum value and values of the group by attributes.

8. **WriteOutThreadMethod:** This method converts each record of the input pipe and writes that string to the file. The ToString method of the Record is used to get a string equivalent of the record.

Results:

```
** IMPORTANT: MAKE SURE THE INFORMATION BELOW IS CORRECT **
 catalog location:      catalog
 tpch files dir:        /cise/homes/dparyani/tpch-dbgen/
 heap files dir:        db-files/


ps_partkey: [6333], ps_suppkey: [6334], ps_availqty: [3711], ps_supplycost: [1.01], ps_comment: [s use slyly. fluffily express requests wake carefully ironic packages]
ps_partkey: [9097], ps_suppkey: [4098], ps_availqty: [3012], ps_supplycost: [1.01], ps_comment: [s the bold pinto beans cajole carefully after the slyly unusual instructions. slyly special packages above the unusual, bold packages cajole blithely even Tiresias. theodolites among the foxes are]
ps_partkey: [20468], ps_suppkey: [469], ps_availqty: [6884], ps_supplycost: [1], ps_comment: [furiously among the slyly ironic instructions. final, unusual packages wake slyly. final accounts cajole. deposits above the i]
ps_partkey: [27115], ps_suppkey: [9618], ps_availqty: [7966], ps_supplycost: [1.02], ps_comment: [e regular, ironic dugouts. slyly special requests cajole quickly across the blithely express requests. deposits unwind carefully pending the odolites. pinto beans about the even, regular theodolite]
ps_partkey: [34494], ps_suppkey: [9501], ps_availqty: [7438], ps_supplycost: [1.02], ps_comment: [egular excuses. final, regular deposits wake. pinto beans according to th]
ps_partkey: [43172], ps_suppkey: [685], ps_availqty: [6600], ps_supplycost: [1.01], ps_comment: [ites integrate blithely above the slyly regular instructions. asymptotes besides the regular, even accounts haggle carefully slyly bold requests. even pinto beans ]
ps_partkey: [43764], ps_suppkey: [1277], ps_availqty: [2344], ps_supplycost: [1.02], ps_comment: [; furious, ironic requests nag furiously against the silent packages-- furiously pending pinto beans use blithely careful]
ps_partkey: [51671], ps_suppkey: [4177], ps_availqty: [3399], ps_supplycost: [1.02], ps_comment: [iously. blithely bold requests haggle furiously. slyly final requests sleep. final, final theodolites cajole. accounts play about the slyly unusual requests. bold courts haggle. bol]
ps_partkey: [60953], ps_suppkey: [954], ps_availqty: [8611], ps_supplycost: [1.01], ps_comment: [ully even dolphins wake carefully about the slyly final pinto beans]
ps_partkey: [61707], ps_suppkey: [1708], ps_availqty: [3178], ps_supplycost: [1.02], ps_comment: [ide of the unusual, regular excuses. unusual, special packages are carefully across the even theodolites: furi]
ps_partkey: [71984], ps_suppkey: [6999], ps_availqty: [6816], ps_supplycost: [1.01], ps_comment: [eodolites are blithely across the special requests. quickly regular excuses are furiously against the slyly final accou]
ps_partkey: [74375], ps_suppkey: [6883], ps_availqty: [864], ps_supplycost: [1.02], ps_comment: [lithely express asymptotes nag regular packages. special, ruthless instructions against the furiously ruthless packages boost around the packages. slyly bold accounts use. furiously ironic pa]
ps_partkey: [76994], ps_suppkey: [9502], ps_availqty: [9712], ps_supplycost: [1], ps_comment: [. carefully ironic platelets cajole furiously among the furiously regular asymptotes. furiously express asymptotes wake caref]
ps_partkey: [93653], ps_suppkey: [3654], ps_availqty: [4473], ps_supplycost: [1.02], ps_comment: [ of the carefully final requests. bold deposits are slyly. instructions nod furiously instructions. careful]
ps_partkey: [102497], ps_suppkey: [2498], ps_availqty: [6491], ps_supplycost: [1], ps_comment: [fully final accounts. even accounts after the carefully final accounts haggle according to the blithely special requests. carefully unusual]
ps_partkey: [122543], ps_suppkey: [5056], ps_availqty: [5753], ps_supplycost: [1], ps_comment: [e the quickly ironic dependencies. slyly ironic accounts]
ps_partkey: [139711], ps_suppkey: [9712], ps_availqty: [4286], ps_supplycost: [1.01], ps_comment: [ully unusual escapades sleep along the special instructions. final, bold ideas across the slyly ironic ideas sleep dependenc]
ps_partkey: [155112], ps_suppkey: [5113], ps_availqty: [7635], ps_supplycost: [1], ps_comment: [refully bold packages. special somas cajole according to the foxes. furiously even accou]
ps_partkey: [158093], ps_suppkey: [5639], ps_availqty: [3751], ps_supplycost: [1], ps_comment: [iously unusual gifts maintain quickly according to the slyly pending deposits. quickly ]
ps_partkey: [193659], ps_suppkey: [6179], ps_availqty: [6606], ps_supplycost: [1.01], ps_comment: [can haggle. quickly express packages are blithely. even requests against the silent accounts sleep special packages. ironic ideas according to the furiously regular dolphins use quickly plate]
ps_partkey: [193981], ps_suppkey: [3982], ps_availqty: [619], ps_supplycost: [1.01], ps_comment: [ic accounts after the unusual, regular instructions grow carefully around the blithely unusual dependencies. pending accounts along the bl]


 query1 returned 21 records
```

```
**********************************************************************************************************************************************************************************

** IMPORTANT: MAKE SURE THE INFORMATION BELOW IS CORRECT **
 catalog location:      catalog
 tpch files dir:        /cise/homes/dparyani/tpch-dbgen/
 heap files dir:        db-files/


int: [31], string: [slate seashell steel medium moccasin], double: [931.03]
int: [1030], string: [orange floral olive ivory lace], double: [931.03]
int: [2029], string: [midnight brown dim violet almond], double: [931.02]
int: [3028], string: [puff slate tomato moccasin azure], double: [931.02]
int: [4027], string: [white ivory moccasin coral puff], double: [931.02]
int: [5026], string: [blanched blush pink light wheat], double: [931.02]
int: [6025], string: [purple medium light aquamarine dark], double: [931.02]
int: [7024], string: [forest rosy peach antique midnight], double: [931.02]
int: [8023], string: [mint salmon moccasin blanched beige], double: [931.02]
int: [9022], string: [peru misty sandy dark drab], double: [931.02]
int: [10021], string: [blush steel green sienna snow], double: [931.02]
int: [11020], string: [plum khaki powder beige peru], double: [931.02]


 query2 returned 12 records
```

```
**********************************************************************************************************************************************************************************

** IMPORTANT: MAKE SURE THE INFORMATION BELOW IS CORRECT **
 catalog location:      catalog
 tpch files dir:        /cise/homes/dparyani/tpch-dbgen/
 heap files dir:        db-files/


double: [9.24623e+07]


 query3 returned 1 records


**********************************************************************************************************************************************************************************

** IMPORTANT: MAKE SURE THE INFORMATION BELOW IS CORRECT **
 catalog location:      catalog
 tpch files dir:        /cise/homes/dparyani/tpch-dbgen/
 heap files dir:        db-files/


 query4
double: [4.00421e+08]
 query4 returned 1 recs


**********************************************************************************************************************************************************************************

** IMPORTANT: MAKE SURE THE INFORMATION BELOW IS CORRECT **
 catalog location:      catalog
 tpch files dir:        /cise/homes/dparyani/tpch-dbgen/
 heap files dir:        db-files/


 query5 finished..output written to file ps.w.tmp
```

GTests:

Instructions to run:

1. Generate 1 Gb of tpch-dbgen data.
2. Update catalog_path, dbfile_dir, & tpch_dir paths in “gtests/RelOpGTests.cc”.
3. Run “make gTestRelOp.out”.
4. Run “./gTestRelOp.out”.

Details:

Following are the details of 4 written GTests:

1. TestLoadVectorFromBlock: This tests the LoadVectorFromBlock method of RelOp. LoadVectorFromBlock method converts gets records from the block of pages and puts that records into the vector. It tests this by adding the nation table’s records into the block of pages. After that it calls the LoadVectorFromBlock method for testing and finally it checks if the number of records in the vector are equal to the number of records pushed into the block of pages.

2. TestNestedBlockJoin: This tests the NestedBlockJoin method of RelOp. NestedBlockJoin method joins records from two input pipes and pushes joined records into the output pipe. Hence this test, creates two pipes with the records of the nation table i.e each pipe contains 25 records. Then it calls NestedBlockJoin for testing and expects the number of records in the output pipe should be equal to 625 i.e 25*25.

3. TestJoinTableBlocks: This tests the JoinTableBlocks method of RelOp. JoinTableBlocks method joins records from two vectors and pushes joined records into the output pipe. Hence this first creates two vectors with the records of the nation table. Then it calls JoinTableBlocks for testing and expects the number of records in the output pipe to be equal to 625.

4. TestAddGroupByRecordToPipe: This tests the AddGroupByRecordToPipe of RelOp. GroupByRecordToPipe creates a group-by record from the sum value, table record and group by order maker. This test checks group-by record formed AddGroupByRecordToPipe method by comparing it with the expected string form the record.

Output:

```
g++ -O2 -Wno-deprecated -o gTestRelOp.out Record.o Comparison.o Compa
x.yyfunc.o -ll -lpthread -lgtest
[==========] Running 4 tests from 1 test case.
[----------] Global test environment set-up.
[----------] 4 tests from RelOpGTests
[ RUN      ] RelOpGTests.TestLoadVectorFromBlock
[       OK ] RelOpGTests.TestLoadVectorFromBlock (1 ms)
[ RUN      ] RelOpGTests.TestNestedBlockJoin
[       OK ] RelOpGTests.TestNestedBlockJoin (1 ms)
[ RUN      ] RelOpGTests.TestJoinTableBlocks
[       OK ] RelOpGTests.TestJoinTableBlocks (1 ms)
[ RUN      ] RelOpGTests.TestAddGroupByRecordToPipe
[       OK ] RelOpGTests.TestAddGroupByRecordToPipe (0 ms)
[----------] 4 tests from RelOpGTests (3 ms total)

[----------] Global test environment tear-down
[==========] 4 tests from 1 test case ran. (3 ms total)
[  PASSED  ] 4 tests.
dhirajmaheshparyani@Dhirajs-MBP Project %
```

Bugs:

1. **No way to fetch number of attributes of a record.**
   Description: Join and GroupBy operation doesn't provide a number of attributes of the records. This makes it difficult to merge or project the records. As MergeRecords and Project methods of Record class require a number of attributes in the record.

   Implemented solution:
   - Get the start location of the 1st attribute. Let's call it x.
   - Calculate number of attributes as x/size(int).
   - The above method will work only if the 1st attribute of record is int. Because Record aligns double and string attributes.
   - Hence if the 1st attribute of schema is anything except int, the above method won't work.

   Possible solutions:
   1) Number of attributes of the record should be passed to Join and GroupBy operations. That is change Join and GroupBy definitions.
   2) Store number of attributes in the bits array of the record. So that it can be fetched directly when required.
      Note: This change will require lots of refactoring

2. **Wrong pipe and schema used in q2 of test.cc::132**
   Description and solution: Call to clear_pipe should be clear_pipe (_out, &out_sch, true) instead of clear_pipe (_p, p->schema (), true).

3. **Incorrect pipe buffer size in q6 of test.cc::283**
   Description and solution: size of output pipe should be at least 25, as we are expecting 25 grouped-by records in the output. Hence it should be Pipe _out (25) instead of Pipe _out (1);

4. **Incorrect order maker passed to group by in q6 test.cc::293**
   Description: q6 represents following query:
   " select sum (ps_supplycost) from supplier, partsupp
   where s_suppkey = ps_suppkey groupby s_nationkey "

   While calling groupBy in Q6 it is passing the entire join schema as order maker for the group by attributes. It should pass an order maker with an attribute containing s_nationkey.

   Implemented solution:
   Added following code to generate correct groupBy order maker:
   "char *group_cnf_str = "(s_nationkey)";

get_cnf (group_cnf_str, &join_sch, cnf_c, lit_c);
OrderMaker groupOrderMaker, dummyOrderMaker;
cnf_c.GetSortOrders(groupOrderMaker, dummyOrderMaker);"

Then created "groupOrderMaker" should be used while calling GroupBy. That is line 293 will change to "G.Run (_s_ps, _out, groupOrderMaker, func);".

5. **Wrong expected output given in the comments in test.cc.**