Project Report: Database System Implementation (COP6726)

Project 3: Relational Operators

Name: Vikas Chaubey**, UFID:** 3511 5826**, Email:** vikas.chaubey@ufl.edu

I have done this project alone without any project partner. Hence, I am submitting it on my behalf.

1) **Compile and Run Instructions:** This program has been compiled, run and tested on Ubuntu 18.04 operating system. Before running the below commands please update all the paths in "**test.cat**" file. In test.cat, the first line should be path for the catlogue file (catalog_path), the second line should be the path for heap files (dbfile_dir), The third line should be the path for the tpch-dbgen data tables (tpch_dir).

Step.1 – To run a2test.out:

In order to load all the tables and generate the heap files we need to run **a2test.out**, we need to make sure that in a2test.h file filepaths like dbfile_dir, tpch_dir and catalog_path is updated correctly with right locations. please use below commands sequentially to run **a2test.out** and follow onscreen instruction to load the table files. Conce the a2test.out is compiled, it has to be run 8 times to load all 8 table files and generate heap files.

```
> make clean
> make a2test.out
> ./a2test.out
```

Step.2 - To run test.out:

In order to run **test.out** first we need to make sure that in "test.cat" has right path to dbfiles, tpch_dbgen files and catalog file, updated correctly with right locations. please use below commands sequentially to run **test.out**.

```
> make test.out
>./test.out   (query number)      [Ex. ./test.out 1]
```

Step.3 - To run runTestCases.sh:

In order to run **runTestCases.sh** please use below commands sequentially.

```
> make test.out
> sh runTestCases.sh
```

Step.4 - To run gtest (google tests):

In order to run **gtest** please use below commands sequentially to run googletests.

```
> make gtest.out
>./gtest
```

2) Project files, structure, classes and functions: DBFile is extended and contains following classes.

1) **RelationOp Class:** RelationOp class is used as a base class to define all the relational operator in this project. All the operator classes inherit the RelationOp base class. This class has two functions, all the relational operator derived from this class inherit these methods. These methods are as below:

    a) **WaitUntilDone:** As the name suggests this function makes the caller wait until the current executing thread of relational operator finishes the execution and completely done. Then only the caller can start the execution. This method provides thread safety for the used resource by multiple threads.

    b) **Use_n_Pages :** Several relational operators derived in this project make use of BIgQ objects , these BigQ objects has run lengths , this function helps in deciding the run length for these BigQ objects.

```
/*=============================================================================================*/
class RelationalOp {

    public:
    // this function holds the calling entity till the executing relational operator has run to completion
    virtual void WaitUntilDone () = 0;

    // This function provides information regarding internal memory usage
    virtual void Use_n_Pages (int n) = 0;
};

/*=============================================================================================*/
```

2) **SelectFile Class:** This class has the RelationaOp class as baseclass. The selectFile class selects a DBFile and reads the records contained in it. It also has a CNF predicate which is applied on the records which are obtained from the DBFIle.Once the predicate is applied then the records are pushed in the output pipe. It has following methods.

```
/*=============================================================================================*/
class SelectFile : public RelationalOp {
    private:
    //Pthread object variable
    pthread_t pthreadObj;
    //Variable to hold record object
    Record *literal;
    //DBfile instance
    DBFile *inFile;
    //output Pipe Object
    Pipe *outPipe;
    //CNF class variable for CNF statement processing
    CNF *selOp;
    //all auxillary functions for this class
    public:
    void Run (DBFile &inFile, Pipe &outPipe, CNF &selOp, Record &literal);
    // this function holds the calling entity till the executing relational operator has run to completion
    void WaitUntilDone ();
    // This function provides information regarding internal memory usage
    void Use_n_Pages (int n);
    //this function is a static caller method
    static void* caller(void*);
    void *operation();
};

/*=============================================================================================*/
```

3) **SelectPipe Class:** The SelectPipe class is similar to the selectFile class, The SelectFile performs its operation in three steps, it reads a DBFile, applies the predicate on the fetched records and then afterwards pushes the records in the output pipe. The SelectPipe class does the same thing but it does not read records from the DBFile instead it reads records from an input Pipe and then it also has an CNF predicate attribute which is applied on the records and then records are pushed in the output pipe.

```
/*=============================================================================================*/
class SelectPipe : public RelationalOp {
    private:
    //Pthread object variable
    pthread_t pthreadObj;
    //input Pipe Object
    Pipe *inPipe;
    //output Pipe Object
    Pipe *outPipe;
    //CNF class variable for CNF statement processing
    CNF *selOp;
    //Variable to hold record object
    Record *literal;
    //all auxillary functions for this class
    public:
    void Run (Pipe &inPipe, Pipe &outPipe, CNF &selOp, Record &literal);
    // this function holds the calling entity till the executing relational operator has run to completion
    void WaitUntilDone ();
    // This function provides information regarding internal memory usage
    void Use_n_Pages (int n);
    //this function is a static caller method
    static void* caller(void*);
    void *operation();
};

/*=============================================================================================*/
```

4) **Project Class:** This project Class manipulates the number of attributes that the records have in a relation. Project class has RelationOp class as base class.

```
/*=============================================================================================*/
class Project : public RelationalOp {
    private:
    //Pthread object variable
    pthread_t pthreadObj;
    //input Pipe Object
    Pipe *inPipe;
    //output Pipe Object
    Pipe *outPipe;
    //variable to keep counter
    int *keepMe;
    //variable to hold integer number at TS input
    int numAttsInput;
    //variable to hold integer number at TS output
    int numAttsOutput;
    //all auxillary functions for this class
    public:
    void Run (Pipe &inPipe, Pipe &outPipe, int *keepMe, int numAttsInput, int numAttsOutput);
    // this function holds the calling entity till the executing relational operator has run to completion
    void WaitUntilDone ();
    // This function provides information regarding internal memory usage
    void Use_n_Pages (int n);
    //this function is a static caller method
    static void* caller(void*);
    void *operation();
};

/*=============================================================================================*/
```

5) **Join Class:** Join class has RelationOp class as base class. Apart from all the inherited methods it has got from the base class, it implements two join algorithms. The first algorithm is Sorted-Merge Join Algorithm, this algorithm is used when a after using a given CNF the OrderMaker object is not formed properly. In this case this algorithm is used. The second Algorithm is Block-Nested Join algorithm, this algorithm applied to all other cases than previous one, it helps applying the CNF predicate without using any OrderMaker object.

```
/*=============================================================================================*/
class Join : public RelationalOp {
    private:
    //Pthread object variable
    pthread_t pthreadObj;
    //input Pipe Object left
    Pipe *inPipeL;
    //input Pipe Object right
    Pipe *inPipeR;
    //output Pipe Object
    Pipe *outPipe;
    //CNF class variable for CNF statement processing
    CNF *selOp;
    //Variable to hold record object
    Record *literal;
    int rl, mc=0, lrc=0, rrc=0;
    //all auxillary functions for this class
    public:
    void Run (Pipe &inPipeL, Pipe &inPipeR, Pipe &outPipe, CNF &selOp, Record &literal);
    // this function holds the calling entity till the executing relational operator has run to completion
    void WaitUntilDone ();
    void Use_n_Pages (int n);
    static void* caller(void*);
    //main join method
    void *operation();
    //method to merge record pages
    void MergePages(vector<Record*> lrvec, Page *rp, OrderMaker &lom, OrderMaker &rom);
    //method to merge records
    void MergeRecord(Record *lr, Record *rr);
    //method which defines  sited merge join : This algorithm is used in case a proper OrderMaker object is not formed using given CNF
    void sortMergeJoin(Record lr,Record rr, Record m, OrderMaker &lom, OrderMaker &rom);
    //method which defines block nested join : This algorithm is run otherwise and helps apply the CNF without using any OrderMaker
    //objects
    void blockNestedJoin(Record lr,Record rr, Record m, OrderMaker &lom, OrderMaker &rom);
};

/*=============================================================================================*/
```

6) **DuplicateRemoval Class: In** SQL data manipulation language, the DISTINCT statement is used to return only distinct records or values from fetched records of a table. This Class implement the DISTINCT keyword functionality, it sorts the fetched records and keeps on eliminating the duplicates in a given schema.

```
/*=============================================================================================*/
class DuplicateRemoval : public RelationalOp {
    private:
    //Pthread object variable
    pthread_t pthreadObj;
    // input Pipe Object variable
    Pipe *inPipe;
    // output Pipe Object variable
    Pipe *outPipe;
    // Variable to hold object of Schema Class
    Schema *mySchema;
    int rl;
    // all auxillary functions for this class
    public:
    void Run (Pipe &inPipe, Pipe &outPipe, Schema &mySchema);
    // this function holds the calling entity till the executing relational operator has run to completion
    void WaitUntilDone ();
    // This function provides information regarding internal memory usage
    void Use_n_Pages (int n);
    //this function is a static caller method
    static void* caller(void*);
    void *operation();
};

/*=============================================================================================*/
```

7) **Sum Class:** In SQL data manipulation language, the SUM aggregate function is used to calculate the sum of all or distinct values in an expression. This class implements the sum aggregation function. This implementation is done using function object. when the rows are selected from the relation then this function is applied on the valid selected rows and the sum is obtained. The output is then pushed to the output pipe as a record.

```
/*================================================================================================*/
class Sum : public RelationalOp {
    private:
    //Pthread object variable
    pthread_t pthreadObj;
    // input Pipe Object variable
    Pipe *inPipe;
    // output Pipe Object variable
    Pipe *outPipe;
    // Variable to hold Function class object
    Function *computeMe;
    public:
    // all auxillary functions for this class
    void Run (Pipe &inPipe, Pipe &outPipe, Function &computeMe);
    // this function holds the calling entity till the executing relational operator has run to completion
    void WaitUntilDone ();
    // This function provides information regarding internal memory usage
    void Use_n_Pages (int n);
    //this function is a static caller method
    static void* caller(void*);
    void *operation();
};

/*================================================================================================*/
```

8) **GroupBy Class:** In SQL data manipulation language, The GroupBy statement is used with aggregate functions, when used with GroupBy clause the aggregate functions are applied on the selected groups made by the GroupBy statement. This class implements the GroupBy functionality, It applies the aggregate functions in group of records with the same order.

```
/*================================================================================================*/
class GroupBy : public RelationalOp {
    private:
    //Pthread object variable
    pthread_t pthreadObj;
    // input Pipe Object variable
    Pipe *inPipe;
    // output Pipe Object variable
    Pipe *outPipe;
    // variable to hold object of ordermaker clas providing group attribiutes
    OrderMaker *groupAtts;
    //Function class object variable
    Function *computeMe;
    int rl;
    public:
    // all auxillary functions for this class
    void Run (Pipe &inPipe, Pipe &outPipe, OrderMaker &groupAtts, Function &computeMe);
    // this function holds the calling entity till the executing relational operator has run to completion
    void WaitUntilDone ();
    // This function provides information regarding internal memory usage
    void Use_n_Pages (int n);
    //this function is a static caller method
    static void* caller(void*);
    void *operation();
};

/*================================================================================================*/
```

9) WriteOut Class: This class works in similar way as the print function works. Print function prints the records on the screen. on the other hand, WriteOut class works little differently, it reads records from input pipe and write those records into a text file as raw records.

```
/*=============================================================================================*/
class WriteOut : public RelationalOp {
    private:
    //Pthread object variable
    pthread_t pthreadObj;
    // input Pipe Object variable
    Pipe *inPipe;
    // output File Object variable
    FILE *outFile;
    //variable to store schema class instance
    Schema *mySchema;
    public:
    // all auxillary functions for this class
    void Run (Pipe &inPipe, FILE *outFile, Schema &mySchema);
    // this function holds the calling entity till the executing relational operator has run to completion
    void WaitUntilDone ();
    // This function provides information regarding internal memory usage
    void Use_n_Pages (int n);
    //this function is a static caller method
    static void* caller(void*);
    void *operation();
};
#endif
/*=============================================================================================*/
```

3) Program Run Results: All the screenshots are added in screenshots folder of this submission**.**

a) Results of "runTestCases.sh", Screenshot of Output1.txt:

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


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. theodolite
s 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 theodolites. pinto beans about the ev
en, 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 hagg
le. 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: [6016], 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. fur
iously 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:        tpch-dbgen/
 heap files dir:        dbfiles/


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:        tpch-dbgen/
 heap files dir:        dbfiles/


double: [9.24623e+07]
                                                                                                                                                                1,1           Top
```

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

** IMPORTANT: MAKE SURE THE INFORMATION BELOW IS CORRECT **
 catalog location:      catalog
 tpch files dir:        tpch-dbgen/
 heap files dir:        dbfiles/


 query4
double: [5.26696e+07]
 query4 returned 1 recs


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

** IMPORTANT: MAKE SURE THE INFORMATION BELOW IS CORRECT **
 catalog location:      catalog
 tpch files dir:        tpch-dbgen/
 heap files dir:        dbfiles/



 Number of records written to output file : 9996
 query5 finished..output written to file ps.w.tmp

```

b) Gtest Results:

```
azureuser113@VM-02:~/a3test$ ./gtest
[==========] Running 4 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 4 tests from QueryTesting
[ RUN      ] QueryTesting.GettingUniqueFilePath
[       OK ] QueryTesting.GettingUniqueFilePath (0 ms)
[ RUN      ] QueryTesting.sum

** IMPORTANT: MAKE SURE THE INFORMATION BELOW IS CORRECT **
 catalog location:      catalog
 tpch files dir:        tpch-dbgen/
 heap files dir:        dbfiles/


[       OK ] QueryTesting.sum (10 ms)
[ RUN      ] QueryTesting.WriteOutTesting

 Number of records written to output file : 10000
[       OK ] QueryTesting.WriteOutTesting (35 ms)
[ RUN      ] QueryTesting.DuplicateRemovalTesting
[       OK ] QueryTesting.DuplicateRemovalTesting (26 ms)
[----------] 4 tests from QueryTesting (71 ms total)

[----------] Global test environment tear-down
[==========] 4 tests from 1 test suite ran. (71 ms total)
[  PASSED  ] 4 tests.
azureuser113@VM-02:~/a3test$
```

4) Observed Bugs:

1) The expected output given for query 2 and query 4 in test.cc file is wrong. On running the query, we get different outputs, it might be because of the change in database data**.**
2) In the file test.cc, query 2, line number - 133: we have to change clear_pipe (_p, p->schema (), true) to clear_pipe (_out, &out_sch, true).
3) In the file test.cc, query 6, line number - 283: we have to change Pipe _out (1) to Pipe _out (pipesz) as it is expecting 25 records.
4) Number of attributes for Merging and Projecting of the Record in the Join and GroupBy queries are not provided which can be passed as a parameter or can be calculated by the following formula. Offset to first attribute- sizeof(int)/sizeof(int)where Offset to first attribute is present in the record.