Database System Implementation

Project 3 - Relational Operations

Developed by –

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

Aim of this project is to extend our existing database system with Relational Operators.

We have implemented Relational Operations by completing the required functions. Completed code is present in RelOp.cc and RelOp.h, has been tested and results have been summarized.

Implementation Details:
We have created the following structure to hold the arguments needed for each thread to perform their respective task.

```
struct ThreadArgs
{

public:
    Pipe *inputPipe, *outputPipe;
    CNF *cnf;
    DBFile *dbFile;
    Record *recordliteral;
    int *keepMe;
    int numAttsInput;
    int numAttsOutput;
    Function *computeMe;
    Schema *mySchema;
    FILE *writeOutFile;
    OrderMaker *groupAtts;
};

typedef struct ThreadArgs workerArgs_t;
```

Relational Operations:

1) SelectFile:

The required member functions for SelectFile operation are:

void Run(DBFile &inFile, Pipe &outPipe, CNF &selOp, Record &literal)

This function packs the arguments that are required into the ThreadArgs structure and passes on to the SelectFile worker function called selectFileWorker. This worker function performs a scan of the input DBFile and for every tuple accepted by the input CNF, it pushes the tuple into the output Pipe as output.

2) SelectPipe:

The required member functions for the SelectPipe operation are:

void Run (Pipe &inPipe, Pipe &outPipe, CNF &selOp, Record &literal)

This takes 2 pipes an input and output pipe and a CNF, it packs all these arguments and sends it to the Select pipe worker function called as SelectPipeWorker which does the work of applying the CNF to every tuple on the input pipe and if a tuple satisfies the CNF it is accepted and pushed to the output pipe.

3) **Project**:

The required member functions for the Project operation are:

void Run (Pipe &inPipe, Pipe &outPipe, int *keepMe, int numAttsInput, int numAttsOutput)

This takes all the arguments, binds it and sends to the Project worker function called ProjectOperation. This function takes an input pipe and an output pipe. It also takes an array of integers needed for output, keepMe as well as the number of attributes for the records coming through the input pipe and the number of attributes to keep from those input records. The array of integers tells the Project which attributes to keep from the input records and which order to Put them in.

4) **Join**:

The required member function for the Join operation are:

void Run (Pipe &inPipeL, Pipe &inPipeR, Pipe &outPipe, CNF &selOp, Record &literal)

This takes all the arguments and packs it and sends to the Join worker function called JoinOperation. We create ordermakers for left and right tables using the input CNF. If the ordermakers are compatible we perform a sorted merge join else, we do a blocked nested loop join. For sorted merge join we initially sort the left and right pipes using the order makers created and then we merge the records from these sorted pipes into one. For blocked-nested loop we first stored the left pipe records into temporary file and then compare then against the right pipe and push records to output pipe.

5) **DuplicateRemoval**:

The required member function for the DuplicateRemoval operation is:

void Run (Pipe &inPipe, Pipe &outPipe, Schema &mySchema)

This function packs all the arguments and sends it to the Duplicate Removal worker function called as DuplicateRemoveOperation. An ordermaker is build using the input schema, post that records are sorted based on this ordermaker. For this sorted pipe of records the consecutive records are compared to check for duplicate records and only distinct records are pushed to the output pipe.

6) <u>Sum</u>:

The required member function for the Sum operation is:

void Run (Pipe &inPipe, Pipe &outPipe, Function &computeMe)

This function packs all its arguments and sends it to the Sum Operation Worker function called SumOperation. We have maintained two types double and integer to aggregate the sum over the records in the input pipe. For all the records a cumulative sum is calculated over the input pipe and once it is done a new record is created, pushed to output pipe.

7) <u>GroupBy</u>:

The required member function for the GroupBy operation is:

void Run (Pipe &inPipe, Pipe &outPipe, OrderMaker &groupAtts, Function &computeMe)

This function packs all its arguments and sends it to the GroupByOperation worker function. Initially we sort records in the input pipe using the ordermaker we got from the input. The records are removed from this sorted pipe and grouped cumulatively, aggregate sum is calculated and new record for each group is created and pushed into the output pipe. Later a new group is created and again sum is calculated cumulatively till the records exist.

8) <u>WriteOut</u>:

The required member function for the WriteOut operation is:

*void Run (Pipe &inPipe, FILE *outFile, Schema &mySchema)*

This function packs all the arguments and sends it to the WriteOut worker function called WriteOutOperation. The records from input pipe are removed and pushed to the file using the input schema.

9) gtest.cc:

This file implements google tests for 4 relational operations named SelectFile, SelectPipe, Project and Sum.

Instructions to Run the Code:

Ensure the directory contains the heap files generated from the Project1 or sorted files from Project2 so that code has access to Bin files or you can even use the a2test.cc file to compile it and run to generate the sorted bin files.

If you want to generate sorted files in this same directory:

Steps:

1) make a2test.out
2) ./a2test.out

Follow the options to generate the sorted bin files.

To run the overall project follow the below steps:

Compile and run test.cc:
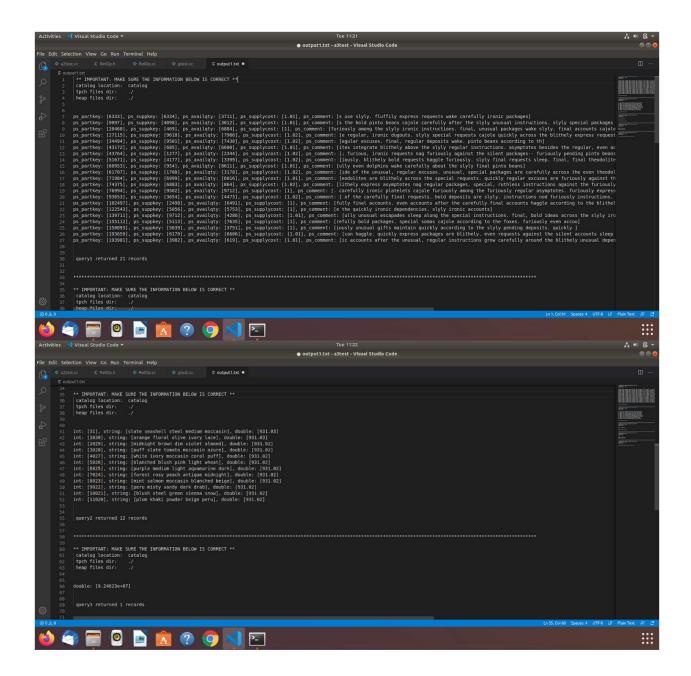
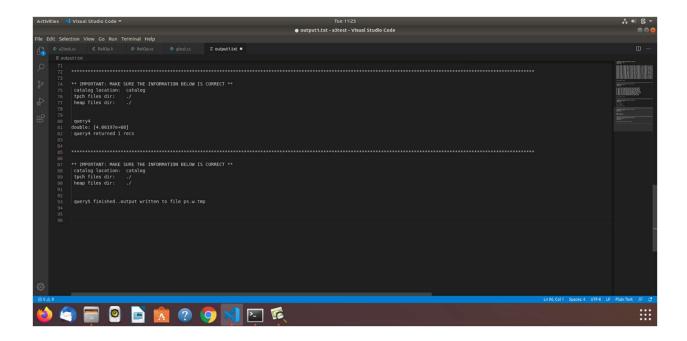Ensure there is Catalog file in the root folder as well

1. make
2. ./test.out [1-8]

To run the test cases

1. make
2. ./runTestCases.sh

File output1.txt is generated

Screen Shot of the output1.txt is below (continued images (3 in total) as entire ouput1.txt cannot be shown in one picture.)

catalog location: catalog
tpch files dir: ./
heap files dir: ./

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
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
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 request
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 ac
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
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 theodolite
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 theodol
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 th
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
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
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.
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 blithel
ps_partkey: [122543], ps_suppkey: [5056], ps_availqty: [5753], ps_supplycost: [1], ps_comment: [e the quickly ironic dependencies. slyly ironic accounts]
ps_partkey: [139713], 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 iro
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
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 depen

   query1 returned 21 records

****

catalog location: catalog
tpch files dir: ./
heap files dir: ./

---

catalog location: catalog
tpch files dir: ./
heap files dir: ./

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

****

catalog location: catalog
tpch files dir: ./
heap files dir: ./

double: [9.24623e+07]

   query3 returned 1 records

Instructions to run gtests:

Compile and run code gtest.cc:

Ensure the directory contains the bin files generated already to run the gtests.

1. make gtest
2. ./gtest

The screenshot of ./gtest is below