Project Report 3

Implementing Relational Operators

Arpan Banerjee

UFID: 9359-9083

[arpanbanerjee@ufl.edu](mailto:arpanbanerjee@ufl.edu)

Krutantak Patil

UFID: 5615-6343

Krutantakb.patil@ufl.edu

***1) Steps to compile and run - code, tests and gtests***

*(*Assuming .tbl files would be provided to us in same directory)

1. **make a2test.out** – Command to compile a2test.cc which creates bin files.
2. **./a2test.out** – Command to run a2test.cc.
3. **make test.out** – Command to build the tests.
4. **./test.out** – Command to run the tests.
5. **make gtest.out** – Command to build the gtests.
6. **./gtest.out** – Command to run the gtests.

***2) Implementations of Relational Operators***

**SelectFile**

1. **Data members**
   1. *pthread\_t thread* – Thread started by Run function.
   2. *Record \*buffer*
   3. *DBFile \*inFile* – Pointer to input DBFile
   4. *Pipe \*outPipe* – Pointer to output pipe
   5. *CNF \*selOp* – Pointer to CNF used to select record
   6. *Record \*literal*
2. **Run** (DBFile &inFile, Pipe &outPipe, CNF &selOp, Record &literal)
   1. Used to set the data member pointers and start a thread using pthread\_create for processing.
   2. This function does the same task for all the operations.
3. **WaitUntilDone** ()
   1. Used to wait for and join the background thread created in Run.
   2. This is a blocking function and performs the same task for all operations.
4. **Use\_n\_Pages** (int runlen)
   1. In Operations that use a BigQ instance, this function is used to initialize the run length. For other operations, it does nothing.
   2. For SelectFile, it does nothing.
5. **select\_file\_function** ()
   1. This function runs on the thread created by Run.
   2. It uses GetNext with the CNF argument to get records from the DBFile and inserts them into the output pipe.

**SelectPipe**

1. **Data members**
   1. *pthread\_t thread*
   2. *Record \*buffer*
   3. *Pipe \*inPipe*
   4. *Pipe \*outPipe*
   5. *CNF \*selOp*
   6. *Record \*literal*
2. **select\_pipe\_function** ()
   1. This function runs on the thread created by Run.
   2. It removes records one by one from the input pipe, compares them using the CNF and inserts it into the output pipe if it’s a match.

**Project**

1. **Data members**
   1. *pthread\_t thread*
   2. *Pipe \*inPipe*
   3. *Pipe \*outPipe*
   4. *int \*keepMe* – Integer array of indices of the record that we want to keep.
   5. *int numAttsInput* – number of attributes in input records
   6. *int numAttsOutput* – number of output attributes to project to.
   7. *Record \*buffer*
2. **project\_function** ()
   1. This function runs on the thread created by Run.
   2. It removes records one by one from the input pipe, uses Project() and then inserts it into the output pipe.

**WriteOut**

1. **Data members**
   1. *pthread\_t thread*
   2. *Record \*buffer*
   3. *Pipe \*inPipe*
   4. *FILE \*outFile* – File to write to.
   5. *Schema \*mySchema* – Schema of the records to write out.
2. **writeout\_function** ()
   1. This function runs on the thread created by Run.
   2. It removes records one by one from the input pipe, goes through all its attributes and keeps writing it to the file.

**DuplicateRemoval**

1. **Data members**
   1. *pthread\_t thread*
   2. *Pipe \*inPipe*
   3. *Pipe \*outPipe*
   4. *Schema \*mySchema*
   5. *int runlen*
2. **duplicate\_function** ()
   1. This function runs on the thread created by Run.
   2. First, the records are sorted (based on all its attributes) using a BigQ.
   3. Then we iterate through the sorted records and insert into the output pipe when the new record is not same as the previous record.

**Sum**

1. **Data members**
   1. *pthread\_t thread*
   2. *Pipe \*inPipe*
   3. *Pipe \*outPipe*
   4. *Function \*func* – Function to apply on the record to calculate sum.
2. **add\_result** (Type rtype, int &isum, int ires, double &dsum, double dres)
   1. This utility function adds the result into the int sum or the double sum depending on the return type.
3. **sum\_function** ()
   1. This function runs on the thread created by Run.
   2. Records are removed one by one from the input pipe and the function is applied. The results are accumulated using add\_result.
   3. Then we create a new schema with 1 attribute (sum) to put the result record into the output pipe.

**GroupBy**

1. **Data members**
   1. *pthread\_t thread*
   2. *int runlen*
   3. *Pipe \*inPipe*
   4. *Pipe \*outPipe*
   5. *OrderMaker \*groupAtts*
   6. *Function \*func*
2. **PerformGroupBy** ()
   1. This function runs on the thread created by Run.
   2. First, the records are sorted using a BigQ and the attributes that we need to group by.
   3. Then, until the value of the record changes (wrt group attributes) we keep adding the value of the function and insert it into the output pipe for each separate group.

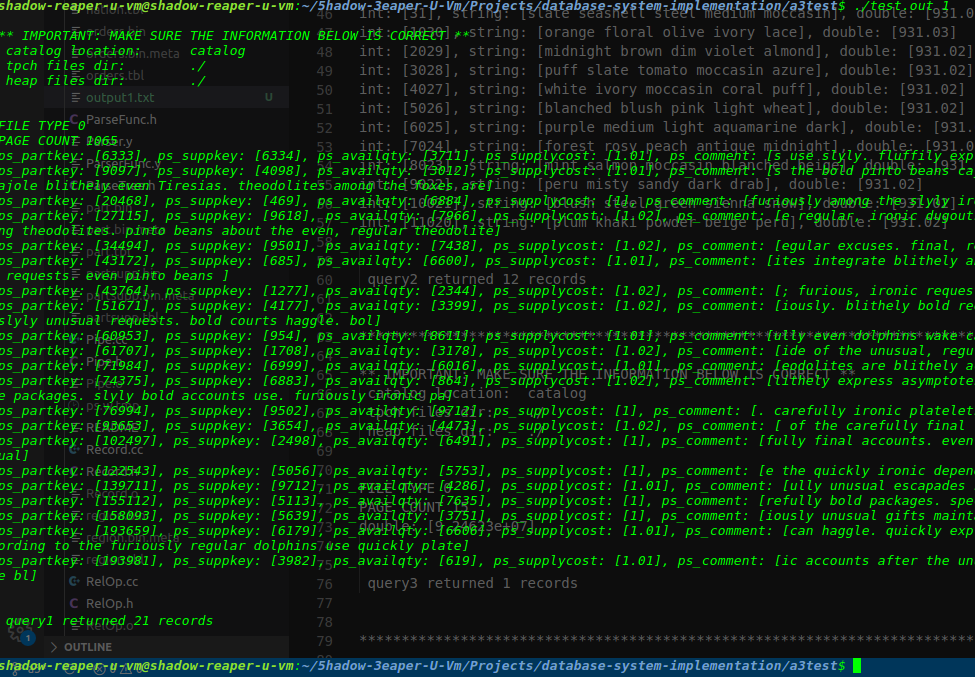
**Join**

1. **Data members**
   1. *CNF \*selOp*
   2. *int runLength*
   3. *Pipe \*leftInPipe, \*rightInPipe, \*outPipe*
   4. *pthread\_t joinThread*
   5. *Record \*literal*
2. **PerformJoin** ()
   1. This function runs on the thread created by Run.
   2. Two instances of BigQ are used, one for the left pipe and one for the right pipe.
   3. These are then merged using the algorithm described in the problem statement or a nested block loop.

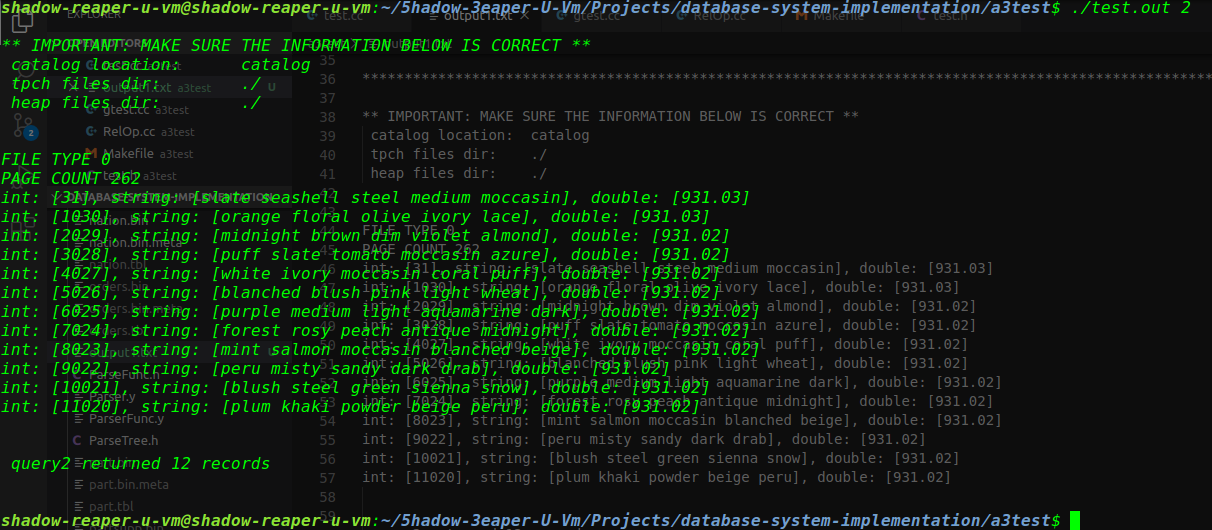
***3) ./test.out results for 1GB data***

1. **Q1**

select \* from partsupp where ps\_supplycost <1.03

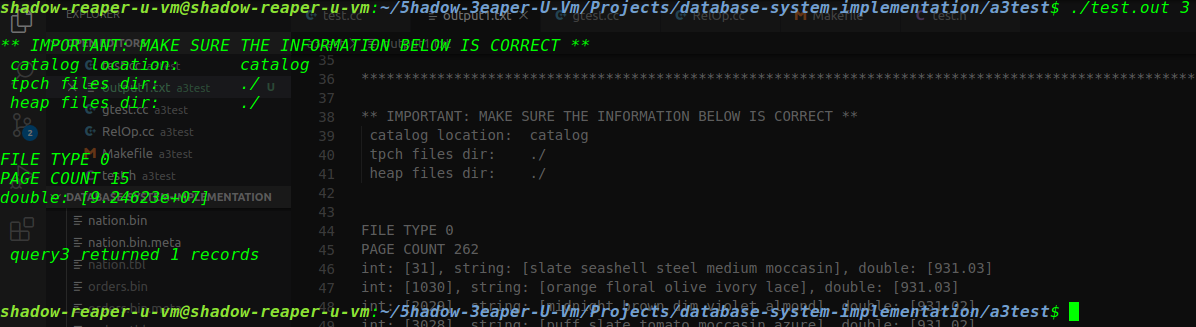


1. **Q2**

select p\_partkey(0), p\_name(1), p\_retailprice(7) from part where (p\_retailprice > 931.01) AND (p\_retailprice < 931.3);

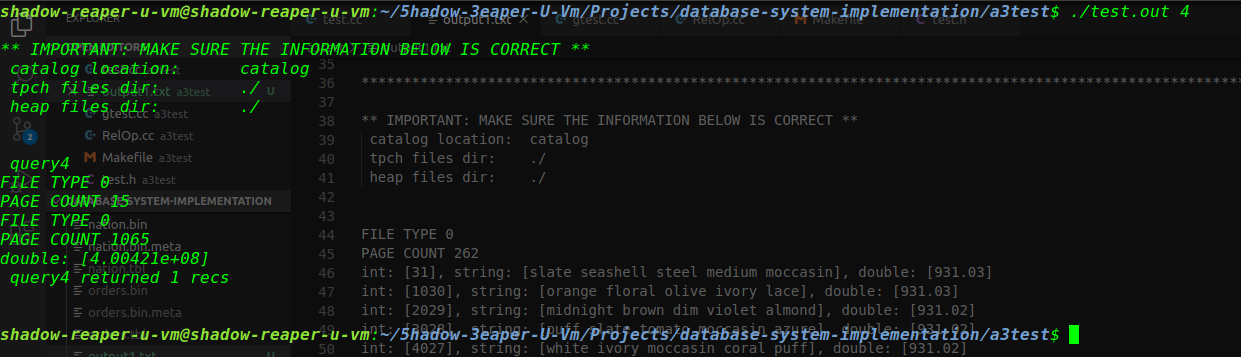
1. **Q3**

select sum (s\_acctbal + (s\_acctbal \* 1.05)) from supplier;

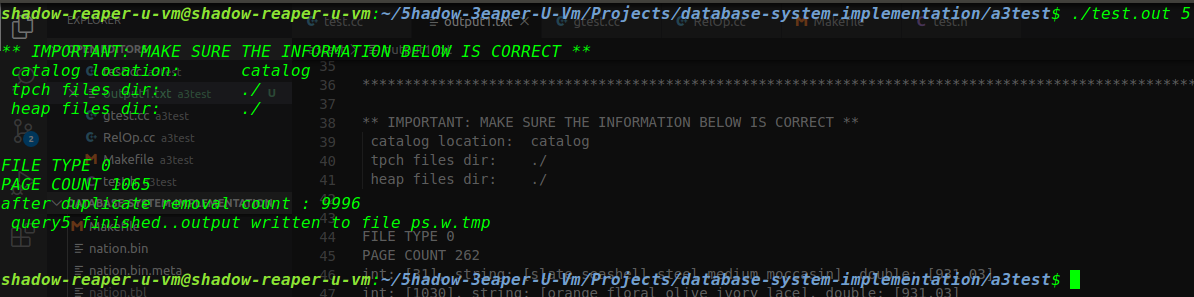


1. **Q4**

select sum (ps\_supplycost) from supplier, partsupp

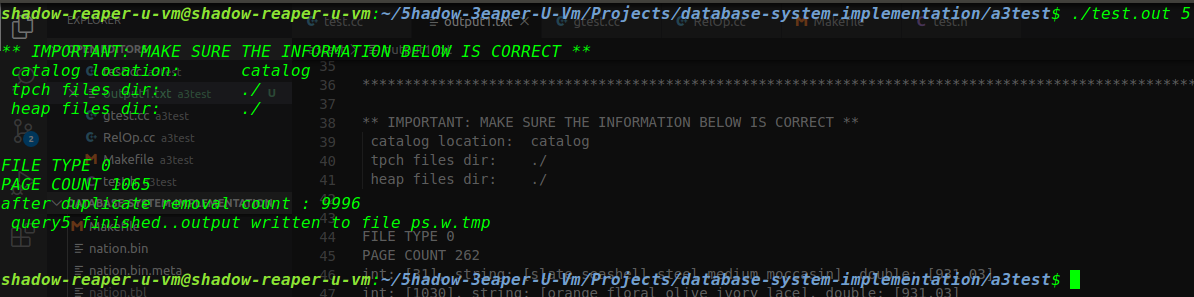
where s\_suppkey = ps\_suppkey;

1. **Q5**

select distinct ps\_suppkey from partsupp where ps\_supplycost < 100.11;

1. **Q6**

select sum (ps\_supplycost) from supplier, partsupp

where s\_suppkey = ps\_suppkey groupby s\_nationkey;

*In q6, we have modified the test to initialize the ordermaker as it wasn’t working initially.*

***3) GTests and results***

1. **SELECT\_GROUP\_BY** : We join the suppliers and partsupp tables using “s\_suppkey = ps\_suppkey”. Then we group it by ps\_supplycost. Finally we assert our output has 25 records.
2. **SELECT\_PIPE\_1:** We select from the partsupp table “ps\_suppcost < 1.03” and verify that we get 21 records.
3. **SELECT\_PIPE\_2:** We select from the partsupp table “ps\_suppcost < 1.04” and verify that we get 31 records.
4. **PROJECT:** We select “(p\_retailprice > 931.00) AND (p\_retailprice < 931.4)” from parts table, then project it to keep attributes 0, 1 and 7.
5. **SUM\_TEST:** From nation table, select using “n\_regionkey > 3” and sum n\_regionkey.

