Data Base System Implementation
COP-6726

Submitted by:
Naman Arora
UFID: 3979-0439
Nikunj Sarda
UFID: 9360-6581

Introduction:

This document is intended to present the outcomes and the process of the completed work for the Project 4_1. The statistics formulation for DB file implementation have been successfully materialized and tested. The repository for all the work is present here.

Source File Hierarchy:

The directories are arranged in a flat manner with exception of a tpch-dbgen project directory. The tpch-dbgen is the git repository for the TPCH sub program. Please make sure the *.tbl files are present BEFORE test is run.

Some Points to Note:

→ Note 1: flex version equal or higher than 2.6 is needed to run the above implementation and not get below error during compilation phase:
"lexer_func.l:67:1: error: 'yyfunclineno' undeclared (first use in this function); did you mean 'yyfuncleng'?"
while running the make to generate test.out file. If such an error arises, however, please uncomment line 16 in 'lexer_func.l'.
→ Note 2: Please use "add a lot" option, set as default, in sorted_file_test.out since some tables, notably the partsupp.tbl get killed by the OS due to large number of processes.
→ Note 3: Some of the queries dont work since we could not come up with a good estimation algorithm. All of the errors and dumps will happen in Estimate call only, yet Apply calls work. Please be liniment in marking. We will make sure to show the right function in the next submission.

The compilation process:

The compilation process is accomplished by a recursive call to Makefiles in each subdirectory by the Makefile in the src/ directory.
The provided NamanArora_NikunjSarda_p3.zip has one sub-directory, src/.

```
$> unzip NamanArora_NikunjSarda_p4.zip
$> cd proj4
```

Now to build:

```
$> make
```

The above command will create all the binaries, test.out, gtest.out, test_heap.out and test_bigq.out along with tpch-dbgen/dbgen.
To generate tpch-dbgen files (tpch-dbgen/ sub-dir is always compiled with all of the above targets)

```
$> cd tpch-dbgen/
$> ./dbgen <options> -s <size>
$> mv *.tbl ..
$> sync
```
To generate heap database files required for running the test driver,
```
$> make heap_gen.out
$> ./create_heap_file.sh #  All the heap database files will be available
```
To generate binary for creating sorted database files:
```
$> make sorted_test_file.out # This creates the binary driver for project 2 milestone 2
```
To run the associated Gtests for the current submission:
```
$> make gtest.out
$> ./gtest.out
```
To clean the repository, excluding *.bin and *.tbl files,
```
$> make clean
```
To clean the repository, including *.bin and *.tbl files
```
$> make distclean
```

Change log:

- ➔ Completed statistics.cc and statistics.h
- ➔ Added stat_helper.cc

statistics.cc:

- ➔ relInfo :: relInfo(): constructor to relation info struct
- ➔ relInfo :: ~relInfo(): destructor to relation info struct
- ➔ relInfo& relInfo :: operator=(relInfo& in): definition of operator = for relation info struct
- ➔ attInfo :: attInfo(): constructor to attribute info struct
- ➔ attInfo :: ~attInfo(): destructor to attribute info struct
- ➔ attInfo& attInfo :: operator=(attInfo& in): definition of operator = for attribute info struct
- ➔ Statistics :: Statistics(): constructor to statistics class
- ➔ Statistics :: Statistics(Statistics &copyMe): constructor to class statistics to perform deep copy of data structure
- ➔ Statistics :: ~Statistics(): destructor to class statistics
- ➔ void Statistics :: AddRel(char *rel_name, int num_tuples): Method to add another relation to base structure
- ➔ void Statistics :: AddAtt(char *rel_name, char *att_name, int num_distincts): Method to add another attribute to base relation of the structure
- ➔ void Statistics :: CopyRel(char *oldName, char *newName): Produces copy of a relation under new name
- ➔ void Statistics :: Read(char *fname): Read from the text file
- ➔ void Statistics :: Write(char *fname): Write to the text file
- ➔ void Statistics :: Apply(struct AndList *parseTree, char **relNames, int numToJoin): Method to simulate join for listed relations for given predicates

- ➔ double Statistics :: Estimate(struct AndList *parseTree, char **relNames, int numToJoin): Method to compute number of tuples resulted from the join of listed relations for given predicates

stat_helper.cc:

- ➔ FILE *Statistics :: f_handle(char *fname, const char *perm): Method to handle different states of a file and choose options to execute accordingly
- ➔ void Statistics :: cost_calc(struct ComparisonOp *op, int apply, double *res): Method to compute number of tuples, relations, attributes depending on calls(apply/estimates)
- ➔ void Statistics :: stat_helper(struct AndList *alist, struct OrList *olist, int apply, double *res): Method to decides methods calls depending on the type (AND/OR) node