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 5. The integration of the previous pieces 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:

Pipe id for input from database file is is -2 and ignore Pipes with ID -1.

→ Note 3:

The video, *.bin files, *.tbl files and all test *.sql files along with the Report are present in proj5/tmp/.

The compilation process:

The compilation process is accomplished by a recursive call to Makefiles in each subdirectory by the root Makefile

The provided NamanArora_NikunjSarda_p5.zip.

```
$> unzip NamanArora_NikunjSarda_p5.zip
$> cd proj5
```

Now to build:

```
$> make
```

The above command will create a5.out

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
```

<u>Change Log</u>:

➔ Added catalog.cc
➔ Added query.h and query.cc
➔ Added ddl.h and ddl.cc
➔ Updated schema.cc
➔ Added enterypoint.h and enterypoint.cc
➔ Added operation.h and operation.cc
➔ Updated qp_tree.cc
➔ Updated qp_tree_helper.cc
➔ Updated statistics.cc
➔ Added tableinfo.cc
➔ Added ops.h and ops.cc

<u>catalog.cc</u>:

➔ void Catalog :: addRel(char *_rname, char *_fname, fType type, int n_tup): Method to add relations
➔ void Catalog :: addAtt(char *_rname, char *aname, int n_dis, Type type, int key): Method to add attributes
➔ void Catalog :: addRel(char *_rname, fType type): Method to add relation without tuples
➔ void Catalog :: addAtt(char *_rname, char *aname, Type type, int key): Method to add attributes with no distinct attributes
➔ void Catalog :: remRel(char *_rname): Method to remove relation
➔ Schema *Catalog :: snap(char *_rname): Method to check relation is present or not and return the schema for the same
➔ void Catalog :: write(char *fname): Method to write output to a file
➔ void Catalog :: read(char *fname): Method to read a file
➔ FILE *f_handle(char *fname, const char *perm): Method to handle files

<u>ddl.cc</u>:

➔ Ddl :: Ddl(struct query *q, char *cat_f): Constructor to class Ddl
➔ Ddl :: ~Ddl(): Destructor to class Ddl
➔ void Ddl :: process(): Method to process query
➔ void Ddl :: create(): Method to create table
➔ void Ddl :: drop(): Method to drop table

➔ void Ddl :: insert(): Method to insert files into table

query.cc:

➔ query :: query(struct FuncOperator *func, struct TableList *tbls, struct AndList *a_list, struct NameList *grp_atts, struct NameList *sel_atts, int dis_att, int dis_func): constructor to structure query, initializing initial values to variables

➔ query :: ~query(): destructor to structure query

➔ void init_stats(): Method to initialize initial state

➔ void query :: dispatcher(): Method to start process as per query

enterypoint.cc:

➔ int main (): Method to initiate the whole process

operation.cc:

➔ operation :: operation(type_flag type, double cost, vector<tableInfo *> &vec): Constructor to structure operation, initializing variables and table

➔ operation :: operation(type_flag type, Schema *sch): Constructor to structure operation, initializing variables

➔ operation :: ~operation(): Destructor to structure operation

➔ void operation :: add_pipe(pipe_type p_type, Pipe *pipe): Method to add pipes

➔ void operation :: append_sch(int indx, struct operation *child): Method to append schema

➔ void operation :: traverse(int indx): Method to traverse tree

ops.cc:

➔ self_op :: self_op(): Constructor to structure self_op

➔ self_op :: ~self_op(): Destructor to structure self_op

➔ void self_op :: traverse(int indx, struct operation *parent): Method to traverse function

➔ selp_op :: selp_op(): Constructor to structure selp_op

➔ selp_op :: ~selp_op(): Destructor to structure self_op

➔ void selp_op :: traverse(int indx, struct operation *parent): Method to traverse operation

➔ join_op :: join_op(): Constructor to structure join_op

➔ join_op :: ~join_op(): Destructor to structure join_op

➔ void join_op :: traverse(int indx, struct operation *parent): Method to traverse join

➔ grpby_op :: grpby_op(): Constructor to structure grpby_op

➔ grpby_op :: ~grpby_op(): Destructor to structure grpby_op

➔ void grpby_op :: traverse(int indx, struct operation *parent): Method to traverse groupby

➔ sum_op :: sum_op(): Constructor to structure sum_op

➔ sum_op :: ~sum_op(): Destructor to structure sum_op

➔ void sum_op :: traverse(int indx, struct operation *parent): Method to traverse sum

➔ dist_op :: dist_op(): Constructor to structure dist_op

➔ dist_op :: ~dist_op(): Destructor to structure dist_op

➔ void dist_op :: traverse(int indx, struct operation *op): Method to traverse distinct operation

➔ proj_op :: proj_op(): Constructor to structure proj_op

➔ proj_op :: ~proj_op(): Destructor to structure proj_op

➔ void proj_op :: traverse(int indx, struct operation *parent): Method to traverse project operation tableinfo.cc:

➔ bool op_comp_sel :: operator()(operation *l, operation *r): Method to compare left and right operation cost

➔ void tableInfo :: add_sel(struct AndList *alist, double cost): Method to add select operation

➔ struct operation *tableInfo :: dispense_sel(Qptree *qpt): Method to remove select operation selected to be executed

schema.cc:

➔ void splice(char *attName, char *placeholder):

➔ Attribute :: Attribute(): Constructor to structure Attribute, initializing varaible to default

➔ Attribute :: Attribute(char *name, Type type): Constructor to structure Attribute, initializing variable with provide values

➔ void Attribute :: update(char *name, Type type): Method to update attributes

➔ Attribute :: ~Attribute(): Destructor to structure attribute

➔ const Attribute &Attribute :: operator=(const Attribute &in): Method to overload operator "=" for attribute comparison

➔ Schema &Schema :: operator+(Schema &in): Method to overload operator "+" to add schema

➔ void Schema :: addAtt(char *aname, Type type, int n_dis, int key): Method to add attributes

qp_tree.cc:

➔ Pipe *Qptree :: dispense_pipe(int *pipe_id): Method to create new pipe

➔ void Qptree :: execute(int flag): Method to execute query parse tree

qp_tree_helper.cc:

➔ bool op_comp_join :: operator()(operation *l, operation *r): Method to compare left and right operation cost for compound join

➔ void Qptree :: mk_ops(struct AndList *alist): Method to traverse to right and make operator

➔ Schema *Qptree :: mk_agg_sch(): Method to make aggressive schema

statistics.cc:

➔ Statistics :: Statistics(Catalog *c): Constructor for class statistics

➔ Statistics :: ~Statistics(): Destructor for class statisitics

➔ double Statistics :: operate(struct OrList *olist): Method to calculate no of tuples for or list

➔ void Statistics :: get_rel(char *_aname, Schema **sch, Attribute **attr): Method to get realtion for given schema, attributes and name

➔ double Statistics :: traverse(struct AndList *alist, struct OrList *olist): Method to traverse and list and or list

➔ double Statistics :: Estimate(struct AndList *a_list): Method to estimate the and list