Project Report: Database System Implementation (COP6726)

Project 4: Query Compilation and Optimization

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: (Skip this step if the bin files are already created and present)
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 a42.out: In order to make a42.out please run below commands.

```
> make clean
> make main.out
> ./a42.out  <  [query]        (Ex. ./a42.out < tc1.sql)
```

Step.3 - To run runTestCases42.sh:
In order to run **runTestCases42.sh** please use below commands sequentially.

```
> make clean
> make main.out
> sh runTestCases42.sh
```

Step.4 - To run gtest (google tests):
In order to run **gtest** please use below commands sequentially to run googletests.

```
> make clean
> make gtest.out
> ./gtest < tc6.sql
```

2) Project files, structure, classes and functions:

1) **QueryTreeNode.h File:** Execution tree node is represented by this file , different classes present in this file provide the basic structure for the execution tree node. The class defines various nodes in form of classes which are explained below:
    a) **The Base Node (RelOpNode Class):** RelOpNode class represents the base node. This base node is inherited by all other nodes in the file.The RelOpNode implements a basic structure from which all other nodes are derived.Below is the basic structure of the RelOpNode class.

```
class RelOpNode {

public:

    int pid;
    RelOpType t;
    Schema schema;
    RelOpNode ();
    RelOpNode (RelOpType type) : t (type) {}
    ~RelOpNode () {}
    virtual void PrintNode () {};

};
```

    b) **Class JoinOpNode (The Join Node):** JoinOpNode class represents the join node. It inherits the basic node structure defined in the RelOpNode Class.

```
class JoinOpNode : public RelOpNode {

public:

    RelOpNode *left;
    RelOpNode *right;
    CNF cnf;
    Record literal;
    JoinOpNode () : RelOpNode (J) {} ;
    ~JoinOpNode () ;
    void PrintNode ();
};
```

c) **Class ProjectOpNode (The Join Node): ProjectOpNode** class represents the join node. It inherits the basic node structure defined in the RelOpNode Class.

```
class ProjectOpNode : public RelOpNode {

public:

    int numIn;
    int numOut;
    int *attsToKeep;
    RelOpNode *from;
    ProjectOpNode () : RelOpNode (P) {}
    ~ProjectOpNode ();
    void PrintNode ();
};
```

d) **Class SelectFileOpNode (The Join Node): SelectFileOpNode** class represents the join node. It inherits the basic node structure defined in the RelOpNode Class.

```
class SelectFileOpNode : public RelOpNode {

public:

    bool opened;

    CNF cnf;
    DBFile file;
    Record literal;
    SelectFileOpNode () : RelOpNode (SF) {}
    ~SelectFileOpNode ();
    void PrintNode ();
};
```

e) **Class SelectPipeOpNode (The Join Node): SelectPipeOpNode** class represents the join node. It inherits the basic node structure defined in the RelOpNode Class.

```
class SelectPipeOpNode : public RelOpNode {

public:

    CNF cnf;
    Record literal;
    RelOpNode *from;
    SelectPipeOpNode () : RelOpNode (SP) {}
    ~SelectPipeOpNode ();
    void PrintNode ();
};
```

f) **Class SumOpNode (The Join Node): SumOpNode** class represents the join node. It inherits the basic node structure defined in the RelOpNode Class.

```
class SumOpNode : public RelOpNode {

public:

    Function compute;
    RelOpNode *from;
    SumOpNode () : RelOpNode (S) {}
    ~SumOpNode ();
    void PrintNode ();
};
```

g) **Class DistinctOpNode (The Join Node): DistinctOpNode** class represents the join node. It inherits the basic node structure defined in the RelOpNode Class.

```
class DistinctOpNode : public RelOpNode {

public:

    RelOpNode *from;
    DistinctOpNode () : RelOpNode (D) {}
    ~DistinctOpNode ();
    void PrintNode ();
};
```

h) **Class GroupByOpNode (The Join Node): GroupByOpNode** class represents the join node. It inherits the basic node structure defined in the RelOpNode Class.

```
class GroupByOpNode : public RelOpNode {

public:

    RelOpNode *from;
    Function compute;
    OrderMaker group;
    bool distinctFuncFlag;
    GroupByOpNode () : RelOpNode (GB) {}
    ~GroupByOpNode ();
    void PrintNode ();
};
```

i) **Class WriteOutOpNode (The Join Node): WriteOutOpNode** class represents the join node. It inherits the basic node structure defined in the RelOpNode Class.

```
class WriteOutOpNode : public RelOpNode {

public:

    RelOpNode *from;
    FILE *output;
    WriteOutOpNode () : RelOpNode (W) {}
    ~WriteOutOpNode ();
    void PrintNode ();
};
#endif
```

2) **QueryPlanner.h File:** This file contains the Query Planner class. The task of query planner is to take the SQL statements passed to it and to parse these SQL statements for execution tree. These execution tree may perform different RelOp operations. These are used to process the statement, but query planner parses the raw SQL statements for them to be processed by the execution tree. The basic structure , variables , functions of the QueryPlanner class are as follow.

```
class QueryPlanner{
public:
    vector<char *> tableNames;
    vector<char *> TableOrderForJoin;
    vector<char *> memoryBuffer;
    TableAliasMap aliasMap;
    SchMap map;
    Statistics s;
    RelOpNode *root;

    QueryPlanner();
    void PopulateSchemaMap ();
    void PopulateStatistics ();
    void PopulateAliasMapAndCopyStatistics ();
    void CopyNameList(NameList *nameList, vector<string> &names) ;
    void Compile();
    void Optimise();
    void BuildExecutionTree();
    void Print();
    booleanMap GetMapFromBoolean(AndList *boolean);
};

QueryPlanner::QueryPlanner():memoryBuffer(2){
    PopulateSchemaMap ();
    PopulateStatistics ();
    yyparse ();
};
```

Function Description of QueryPlanner Class:

a) QueryPlanner ();

Description: This is the Query Planner class constructor which initiates the class object. Constructor are responsible to initialize the necessary variables for query parsing and optimization plan. Initially yyparse() function is called by constructor , this method initiates all the extern data variables used in the program present in this file. Once the parsing is completed then the schema maps are populated from the string and attribute statistics for the tables present in the catalog

b) void PopulateSchemaMap ();

Description: When the Query Planner class is invoked then the constructor executes. This method is called when the constructor is invoked. When the PopulateSchemaMap executes then it populates the SchemaMap variable present in the class. Each of the string names of schemas are mapped to the corresponding schema objects.

c) void PopulateStatistics ();

Description: This method is used to populate statistics object. When the Query Planner class is invoked then the constructor executes. This method is called when the constructor is invoked. To get the statistics for each of the attributes which are related to the relations “AddRel” method is invoked in the process to populate complete statistics of attributes.

d) void PopulateAliasMapAndCopyStatistics ();

Description: if a query contains alias names. Aliases are the different name references for the tables. when query is parsed then all the alias names are saved with the corresponding table in the alias maps for the uniformity in table names. The statistics of the original name is copied to alias name.

e) void CopyNameList (NameList *nameList, vector<string> &names) ;

Description: This function is utilized by several other methods present in this class. This method converts the NameList data structure type to a vector of strings. The reference of the NameList data structure is passed its completely read till end to populate the vector of strings.

f) void Compile ();

Description: This function is responsible to be the starting point of the implemented logic. It is main component in logic execution because it calls function like compile() and BuildExecutiontree() which mainly implement the logic for processing.

g) void Optimise ();

Description: when there are more than 1 join then in this case optimize function makes use of estimate () method to come up with the best ordering of joins which has the minimum estimated execution cost. This join ordering is saved in a variable which could be utilized later during the node creation.

h) void BuildExecutionTree ();

Description: BuildExecutionTree method is responsible for building a query plan tree with the available processed information. Different type of query nodes corresponding to 7 relational operator implementations like selectfile and select pipe etc are built and then they are appended to the corresponding pointer. Unique pipe ids are assigned and then in last updated schema is passed down to the child.

i) void Print ();

Description: this function prints the tree branch with minimum cost. This function is used to populate the ouput schemas with other useful information such attributes etc.

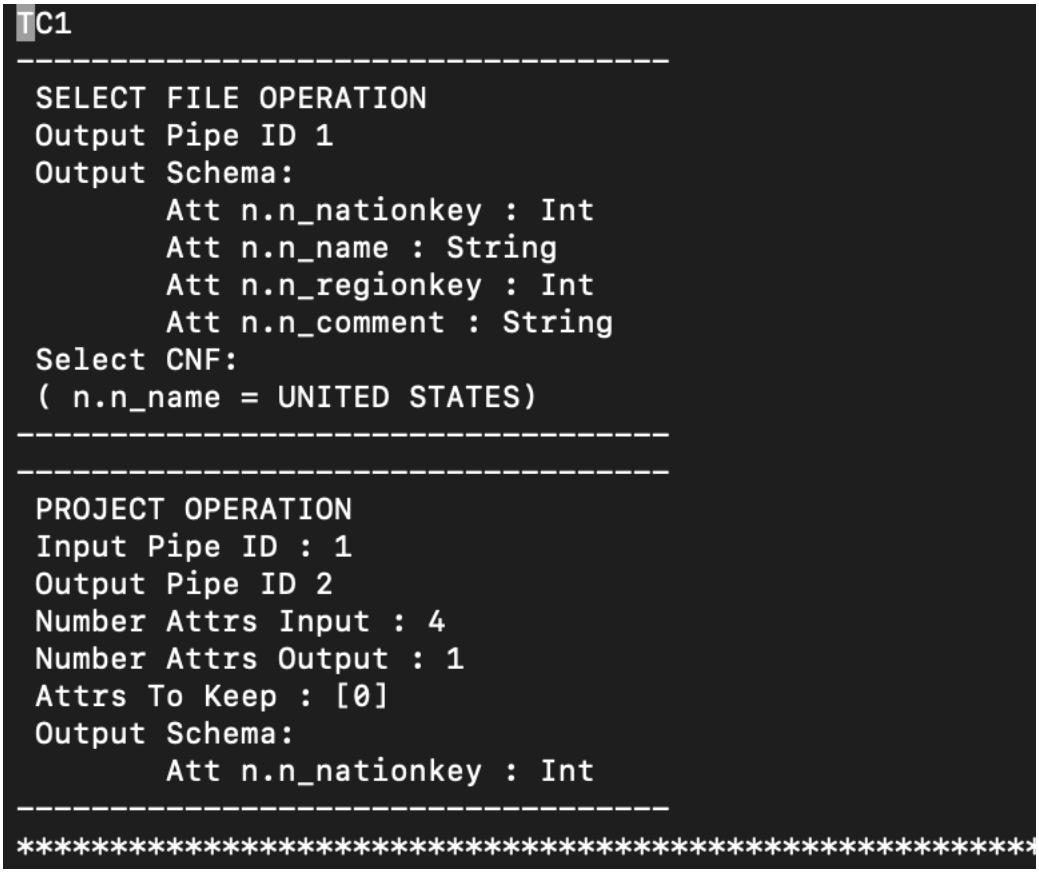
j) booleanMap GetMapFromBoolean(AndList *boolean);

Description: Given AndList this function identifies the condition which are responsible for the joins and create a map which contains all the possible combination of the joins.

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

a) Results of "runTestCases42.sh" - Screenshots from output42.txt file: (output for each query)

Query 1



Query 2

```
****************************************************
TC2
----------------------------------
 SELECT FILE OPERATION
 Output Pipe ID 1
 Output Schema:
        Att n.n_nationkey : Int
        Att n.n_name : String
        Att n.n_regionkey : Int
        Att n.n_comment : String
 Select CNF:
 ( n.n_nationkey > 5)
----------------------------------
----------------------------------
 SELECT FILE OPERATION
 Output Pipe ID 3
 Output Schema:
        Att r.r_regionkey : Int
        Att r.r_name : String
        Att r.r_comment : String
 Select CNF:
----------------------------------
----------------------------------
 JOIN OPERATION
 Left Input Pipe ID : 1
 Right Input Pipe ID : 3
 Output Pipe ID : 2
 Output Schema :
        Att n.n_nationkey : Int
        Att n.n_name : String
        Att n.n_regionkey : Int
        Att n.n_comment : String
        Att r.r_regionkey : Int
        Att r.r_name : String
        Att r.r_comment : String
 Join CNF :
 ( n.n_regionkey = r.r_regionkey)
----------------------------------
----------------------------------
 PROJECT OPERATION
 Input Pipe ID : 2
 Output Pipe ID 4
 Number Attrs Input : 7
 Number Attrs Output : 1
 Attrs To Keep : [0]
 Output Schema:
        Att n.n_name : String
----------------------------------
****************************************************
```

Query 3

```
----------------------------------
**********************************************************************
TC3
----------------------------------
SELECT FILE OPERATION
 Output Pipe ID 1
 Output Schema:
        Att n.n_nationkey : Int
        Att n.n_name : String
        Att n.n_regionkey : Int
        Att n.n_comment : String
 Select CNF:
 ( n.n_name = UNITED STATES)
----------------------------------
----------------------------------
 SELECT FILE OPERATION
 Output Pipe ID 3
 Output Schema:
        Att r.r_regionkey : Int
        Att r.r_name : String
        Att r.r_comment : String
 Select CNF:
----------------------------------
----------------------------------
 JOIN OPERATION
 Left Input Pipe ID : 1
 Right Input Pipe ID : 3
 Output Pipe ID : 2
 Output Schema :
        Att n.n_nationkey : Int
        Att n.n_name : String
        Att n.n_regionkey : Int
        Att n.n_comment : String
        Att r.r_regionkey : Int
        Att r.r_name : String
        Att r.r_comment : String
 Join CNF :
 ( n.n_regionkey = r.r_regionkey)
----------------------------------
----------------------------------
 SUM OPERATION
 Input Pipe ID : 2
 Output Pipe ID : 4
 Function :
 (n.n_nationkey)
 Output Schema:
        Att sum : Int
----------------------------------
**********************************************************************
```

Query 4

```
----------------------------------
**********************************************************************
TC4
----------------------------------
 SELECT FILE OPERATION
 Output Pipe ID 1
 Output Schema:
        Att n.n_nationkey : Int
        Att n.n_name : String
        Att n.n_regionkey : Int
        Att n.n_comment : String
 Select CNF:
 ( n.n_name = UNITED STATES)
----------------------------------
----------------------------------
 SELECT FILE OPERATION
 Output Pipe ID 3
 Output Schema:
        Att r.r_regionkey : Int
        Att r.r_name : String
        Att r.r_comment : String
 Select CNF:
----------------------------------
----------------------------------
 JOIN OPERATION
 Left Input Pipe ID : 1
 Right Input Pipe ID : 3
 Output Pipe ID : 2
 Output Schema :
        Att n.n_nationkey : Int
        Att n.n_name : String
        Att n.n_regionkey : Int
        Att n.n_comment : String
        Att r.r_regionkey : Int
        Att r.r_name : String
        Att r.r_comment : String
 Join CNF :
 ( n.n_regionkey = r.r_regionkey)
----------------------------------
----------------------------------
 GROUP OPERATION
 Input Pipe ID : 2
 Output Pipe ID : 4
 Output Schema :
        Att sum : Int
        Att n.n_regionkey : Int
 Function :
 (n.n_regionkey)
 Grouping On :
        NumAtts =  1
        n.n_regionkey : Int
----------------------------------
**********************************************************************
```

Query 5

```
**********************************************************************
TC5
-----------------------------------
 SELECT FILE OPERATION
 Output Pipe ID 1
 Output Schema:
        Att n.n_nationkey : Int
        Att n.n_name : String
        Att n.n_regionkey : Int
        Att n.n_comment : String
 Select CNF:
 ( n.n_nationkey > 10)
-----------------------------------
-----------------------------------
 SELECT FILE OPERATION
 Output Pipe ID 3
 Output Schema:
        Att r.r_regionkey : Int
        Att r.r_name : String
        Att r.r_comment : String
 Select CNF:
-----------------------------------
-----------------------------------
 JOIN OPERATION
 Left Input Pipe ID : 1
 Right Input Pipe ID : 3
 Output Pipe ID : 2
 Output Schema :
        Att n.n_nationkey : Int
        Att n.n_name : String
        Att n.n_regionkey : Int
        Att n.n_comment : String
        Att r.r_regionkey : Int
        Att r.r_name : String
        Att r.r_comment : String
 Join CNF :
 ( n.n_regionkey = r.r_regionkey)
-----------------------------------
-----------------------------------
 SELECT FILE OPERATION
 Output Pipe ID 4
 Output Schema:
        Att c.c_custkey : Int
        Att c.c_name : String
        Att c.c_address : String
        Att c.c_nationkey : Int
        Att c.c_phone : String
        Att c.c_acctbal : Double
        Att c.c_mktsegment : String
        Att c.c_comment : String
 Select CNF:
-----------------------------------
-----------------------------------
 JOIN OPERATION
 Left Input Pipe ID : 2
 Right Input Pipe ID : 4
 Output Pipe ID : 5
 Output Schema :
        Att n.n_nationkey : Int
        Att n.n_name : String
        Att n.n_regionkey : Int
        Att n.n_comment : String
        Att r.r_regionkey : Int
        Att r.r_name : String
        Att r.r_comment : String
        Att c.c_custkey : Int
        Att c.c_name : String
        Att c.c_address : String
        Att c.c_nationkey : Int
        Att c.c_phone : String
        Att c.c_acctbal : Double
        Att c.c_mktsegment : String
```

```
        Att c.c_comment : String
 Join CNF :
 ( n.n_nationkey = c.c_nationkey)
---------------------------------
---------------------------------
 GROUP OPERATION
 Input Pipe ID : 5
 Output Pipe ID : 6
 Output Schema :
        Att sum : Int
        Att r.r_regionkey : Int
 Function :
 ( (n.n_nationkey + r.r_regionkey))
 Distinct Function :  True
 Grouping On :
        NumAtts =  1
        r.r_regionkey : Int
---------------------------------
*********************************************
```

b) Gtest Results:

```
azureuser113@VM-01:~/t2$ ./gtest< tc6.sql
[==========] Running 2 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 2 tests from QueryTesting
[ RUN      ] QueryTesting.CheckQueryAlias
[       OK ] QueryTesting.CheckQueryAlias (0 ms)
[ RUN      ] QueryTesting.TestJoinOptimization
[       OK ] QueryTesting.TestJoinOptimization (0 ms)
[----------] 2 tests from QueryTesting (0 ms total)

[----------] Global test environment tear-down
[==========] 2 tests from 1 test suite ran. (0 ms total)
[  PASSED  ] 2 tests.
azureuser113@VM-01:~/t2$
```