**Assignment 4.2: Query Compilation and Optimization**

Himavanth Boddu 32451847

Akib Maredia 38856489

**Steps to run:**

* Unzip the compressed file using akibMaredia\_himavanthBoddu\_p42.zip
* cd akibMaredia\_himavanthBoddu\_p42
* make main for generating dependencies and run ./main.
* make a42.out to compile a42.out(“./a42.out” to run the compiled binary and provide your query as input)
* ./runTestCases42.sh to test the output
* make gtest to compile Gtest.cc
* ./gtest to run test cases

**Implementation:**

* We have defined a recursive routine for enumerating all possible query plans.
* We will be using the stats code that we implemented in the previous project 4.1 to roughly estimate the size of the intermediate relations that our query will produce.
* For using the stats code, we first initialize our statistics object with all of our statistics that we were provided in the first part of project 4.
* After this we will be able to produce a query plan with the help of the recursive optimization routine which will be represented as a tree-based data structure.
* There will be seven different node types being associated with each relational operation from project 3. Each node should contain the output schema for records that are being piped out of the operation as well as any other operation-specific information which will be needed to actually invoke or call the relation operation.
* Such as in case of a relational projection node, we will store the last three parameters to the Project operation in the node. In the case of relational join node, we will store the last two parameters to the Join operation in the node.
* We will not create or store any pipes as they will be implied by the structure of the tree. In short, there will be a pipe from a child node to the corresponding parent node (the child and parent being different operations like Join and Project).
* These pipes will be created during the execution of the query plan.
* The main program will print the in-order traversal of the plan as the output.

**We implemented code in a new files named MainScheme.cc and MainScheme.h**

**We implemented Print() function in all classes to show respective print pattern**

Function operator structure :checks if Aggregate function present in the query - NULL if there's no aggregate. Tables list has List of tables and aliases in the query. boolean andlist has Predicate in the WHERE clause

groupingAtts have Attributes on which grouping is to be done - NULL if there's no grouping and attsToSelect have attributes for select query.

distinctAttrs variable checks in case of distinct present in a non-aggregate query similarly distinctFunc checks if

distinct present in aggregate query.

Stores the scheme in map namedSchemeStore and tablename aliases in AliasMap;

InitializeStatistics and initializeschemaStore are gtest functions which check if the assertion of given schema and table us true or not.

SumTreeNode handles the sum function in the query and it takes all the filtered records and prints the sum of the output obtained.

SelectPipeTreeNode will filter elements in pipeline whenever we need to read data from a new table.

JoinTreeNode works for join of the table using specific criterion mentioned, the records are filtered in such manner.

GroupByTreeNode works for group by aggregate function on a given criterion by filtering the records in the inpipe and send to outpipe.

ProjectTreeNode is invoked whenever a query is being processed and done with.

DistinctTreeNode discards the duplicate elements gives the count of the records satisfying the query.

**Output:**

**A screenshot of a computer screen

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer screen

Description automatically generatedA screenshot of a computer screen

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer screen

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer screen

Description automatically generatedA screenshot of a computer screen

Description automatically generated**

* **Gtest**

**A screenshot of a cell phone

Description automatically generated**