Project Report 4.2

Query Compilation and Optimization

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 a42.out** – Command to compile a42.cc into executable.
2. **./a42.out** – Command to run a42.out.
3. **make gtest.out** – Command to build the gtests.
4. **./gtest.out** – Command to run the gtests.

***2) Implementation of Query Planner***

1. **class Optimizer**
   1. This is the main class of the optimizer for which the constructor is called from a42.cc.
   2. Stores the statistics and processes all the operations evaluating the best order.
   3. Constructor - *Optimizer(Statistics\* stats)* - The constructor performs the following tasks in this order -
      1. *constructLeafNodes()*
      2. *processJoins()*
      3. *ProcessSums()*
      4. *processProjects()*
      5. *processDistinct()*
      6. *processWrite()*
      7. *printNodes()*
2. **class OptimizerNode**
   1. Encapsulates a node (operation) in the query planning tree.
   2. This is the base class that all nodes inherit from.
   3. Provides a skeleton infrastructure including pipes, schema and print functionality.
   4. Constructors -
      1. OptimizerNode(const string& op, Schema\* outSchema, Statistics\* stats);
      2. OptimizerNode(const string& op, Schema\* outSchema, char\* relation, Statistics\* stats);
      3. OptimizerNode(const string& op, Schema\* outSchema, char\* relations[], size\_t num, Statistics\* stats);
   5. Member functions
      1. virtual void print(ostream& os = cout) const;
      2. virtual void printAnnot(ostream& os = cout) const = 0;
      3. virtual void printPipe(ostream& os) const = 0;
      4. virtual void printChildren(ostream& os) const = 0;
3. **class LeafNode**
   1. Inherits OptimizerNode to represent leaf nodes in the tree.
   2. These are effectively SelectFile operations, also storing a CNF for selection.
   3. *hasCNF()* - Returns true if a CNF is used for selection.
4. **class UnaryNode**
   1. Inherits Optimizer Node and is used to represent nodes with one children such as Project, Dedup etc.
   2. Contains a pointer to its child node.
   3. Nodes that inherit from UnaryNode -
      1. ProjectNode
      2. DedupNode
      3. GroupByNode
      4. SelectPipeNode
      5. SumNode
      6. WriteNode
5. **class** BinaryNode
   1. Inherits Optimizer Node and represents nodes with two children.
   2. Stores pipe ids and pointers for both children.
   3. Nodes that inherit from BinaryNode -
      1. JoinNode

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

1. **Q1**

select \* from partsupp where ps\_supplycost <1.03

***3) GTests and results***

1. **TEST (OPTIMIZER\_TEST, CHECK\_JOIN\_QUERY\_COUNT)** : We are running the query in tc6.sql and checking the results. This test checks that the number of joins calculated is correct.
2. **TEST (OPTIMIZER\_TEST, CHECK\_SELECT\_QUERY\_COUNT) :** This test checks the number of select nodes or leaf nodes created.





