**FACULTY OF INFORMATION TECHNOLOGY**

**BACHELOR OF BUSINESS INFORMATION TECHNOLOGY**

**Writing Essay**

**BBT 3104: ADVANCED DATABASE SYSTEMS**

**Admission Number: *102856***

**Name: *Nabeel Rafique***

**Group: C: *(Exempt)***

**QUESTION**

*Write a 700 – 2,000-word essay on how each of the 8 cost estimation techniques (refer to slides 75 to 89) can be combined with various heuristic rules during query optimization by the DBMS and the effect this has on reducing the workload of a DBA*

**Introduction**

The cost estimation techniques refer to how the cost-based strategies relate to the business decision to base the price of a product on the costs of production rather than external factors such as competition or the economic environment, this is however from a business perspective but is relevant for a Database Administrator. On the other hand, queryprocessing is the translation of high-level queries into low-level expressions that can be used at the physical level of a file system while query optimization is the actual execution of the query to get the result.

The Cost-based query optimization compares different strategies based on relative costs (amount of time that the query needs to run) and selects and executes one that minimizes the cost. In this case the expense is in the sense of time taken for the query to run.

However from the Artificial Intelligence perspective, a **heuristic** is a technique designed for solving a problem more quickly when classic methods are too slow, or for finding an approximate solution when classic methods fail to find any exact solution. As a result the main focus is how we can combine both the Cost-based query optimization and heuristic rules to reduce the workload.

Firstly our main focus is how this objective can be achieved, one has to understand how the process works. Firstly, from my understanding Cost-based query optimization is an overall process of choosing the most efficient means of executing a SQL statement based on the overall cost of the query. The efficient execution is the execution with minimum cost. To find the cost of query execution plan, the optimization technique uses database statistics.

In this optimization technique, all of the possible ways or scenarios in which a query can be executed will be assigned a ‘cost’, which indicates how efficiently that query can be run. Then, the optimizer will pick the scenario that has the least cost and execute the query using that scenario, because the query with least cost is the most efficient way to run the query. The dominant cost in query processing for centralized relational databases is disk access because disk access is slower than memory access (Disk Access refers to Reading and writing the physical part of the disk. It generally refers to the most time-consuming part of the operation, which is moving the read/write head.) The estimation of the cost is based on:

1. Statistical information about relations. An example would be the number of tuples and the number of distinct values for an attribute
2. Statistics estimation for intermediate results to compute cost of complex expressions
3. Cost formulae for algorithms which is computed using statistics

The cost estimation approach to query optimization compares different strategies based on their relative costs and selects the one that minimizes resource usage, the given cost-based approaches in the slides given to refer to include using Search Keywords and Indices. The Types of searches include;

1. A linear search looks down a list, one item at a time, without jumping.
2. A binary search is when you start with the middle of a sorted list, and see whether that's greater than or less than the value you're looking for, which determines whether the value is in the first or second half of the list
3. An equality search is a type of search filter that can be used to identify entries that contain a specific value for a given attribute. The server will use an equality matching rule to make the determination

The given cost-based approach techniques for query optimization include:

* 1. Linear search on a heap file that has no index
  2. Binary search on a sorted file that has no index
  3. Equality search on an attribute that has a hash index
  4. Equality search on an indexed primary key
  5. Inequality search on an indexed primary key
  6. Equality search on an indexed (clustered) non-primary key
  7. Equality search on an indexed (nonclustered) non-primary key
  8. Inequality search on an indexed

While on the other hand, the heuristic approach to query optimization orders the operations in a query using transformation rules that are known to generate good execution strategies, so here the main focus is on good quality regardless of the cost.

While on the other hand, the heuristic approach to query optimization orders the operations in a query using transformation rules that are known to generate good execution strategies. The rules include:

1. Cascading selections. A list of conjunctive conditions can be broken up into separate individual conditions.
2. Commutativity of the selection operation.
3. Cascading projections. All but the last projection can be ignored.
4. Commuting selection and projection. If a selection condition only involves attributes contained in a projection clause, the two can be commuted.
5. Commutativity of Join and Cross Product.
6. Commuting selection with Join.
7. Commuting projection with Join.
8. Commutativity of set operations. Union and Intersection are commutative.
9. Associativity of Union, Intersection, Join and Cross Product.
10. Commuting selection with set operations.
11. Commuting projection with set operations.
12. Logical transformation of selection conditions. For example, using DeMorgan’s law.
13. Combine Selection and Cartesian product to form Joins

**Objective**

The objective is to merge the concept of the low cost and maintain good quality of queries that are executed. To achieve this, we can look at the current procedure of how queries are executed and dig in deep. We can have a look at Oracle, Oracle currently uses cost based optimization, and rule based optimization although rule based optimization was removed later. The Oracle database provides query optimization. A query optimizer is a component in a database management system (DBMS) that analyzes Structured Query Language (SQL) queries and determines efficient execution mechanisms. A query optimizer generates one or more query plans for each query, each of which may be a mechanism used to run a query. You can influence the optimizer's choices by setting the optimizer goal, and by gathering representative statistics for the query optimizer. The optimizer’s goal is either throughput or response time.

**Approach**

To achieve the given objective, I propose the following action. As we know, the order of execution of the steps changes the cost of the execution. In query optimizer, a binary tree is obtained. In the proposed idea, all the dependent variables are set to one side of the branch of the tree. Each variable is assigned some weight in our proposed algorithm and for simulation purpose, I calculated the cost on the basis of the total weight. Weight is assigned according to how much time that variable or operation takes.

**ALGORITHM 1**

We can take an example of a function which creates “magic tree”

1. Parsing of the query.
2. Building the tree.
3. Selection operation moves at the head node of the tree.
4. All subsequent selections are removed.
5. Projection operation moves next to the selection.
6. All subsequent projections are removed.
7. All dependent groups are formed and will branch to the same side of the tree.
8. Leaf node is a relation, so once we reach leaf, operation is terminated.
9. Search query is initiated.
10. As soon as the target is found, it moves to the projection and appropriate functions are performed.

Heuristics has always proved to be a useful tool. Sometimes the result may not show an improvement in early stage, but after sometime it will show an improvement as soon as its memory is filled with the usage information. In any organization or in any system of a database, generally the same query is executed after certain time. So every-time a tree must be built and then a new search query is executed. So, if we know in advance where the search is, then one can directly go there and it will save the time and hence decreases the cost. Computational time of the execution of the operation, higher will be the weight of that operation.

**ALGORITHM 2**

When a search query is triggered, it initiates the search of the requested  
item. All the items will be at leaf. As we have discussed earlier, all dependent variables will be on same side of the branch of the tree obtained from the query from algorithm In the proposed idea I have reordered certain variables and eliminated certain variables.

Function: heuristics query search.

1. Once a query is run, a storage file is created.
2. Counter is set to company usage factor for each storage file.
3. “Magic tree” is stored in the storage where the number of storage files to be created is equal to company usage factor (c.u.f).
4. When the next query is run, it first checks the equivalence of the tree with any tree in any storage file.
5. If the trees are equal, then it will go to the path of search to the branch directly as stored in the storage file.
6. If the search is successful than the appropriate actions are performed.
7. If the search fails, it will search in the magic tree.
8. If the trees are not equivalent, then it will form its own magic tree as described in algorithm 1 and the counter is increased.

The effect of this approach would be:

1. Rather than considering time constraints it would adapt to client requirements
2. Speed of query retrieval increase.
3. The query-tree would perform selection early which would reduce the number of tuples
4. It would perform projection early (reduces the number of attributes)
5. It would perform most restrictive selection and join operations (i.e. with smallest result size) before other similar operations.