Summary of Query Optimization

By S. Xiao Fernández Marín

1 Summary

The query optimization is an activity performed by an query optimizer- Its job is to select the best available strategy for executing a query.

One way of optimization is to apply heuristic rules (apply SELECT and PROJECT before JOIN) for the internal representation of a query as a form of a query tree and query graph data. It starts with an initial query optimization (according to heuristic rules) and leads to an optimized query representation.

A query tree is a data structure corresponding with an extended relational algebra expression. The leafs of the tree (first thing to be executed) represent the input relations. The internal nodes represent the relational algebra expression and the root the SELECT. The execution would be: 1. Executing internal node, whenever its operands are available. 2. Replacing *1* by the relation that results from the execution of the operation.

The heuristic optimization of the query trees starts with the initial query tree, without any optimization. That canonical query tree is very inefficient. That is why it needs to be optimized to get the final query tree.

For doing that we have to take into account some general transformation rules:

Cascade of selection: One conjunctive selection is the same as individual selections.
Commutativity of selection: The order of selection does not matter.
Cascade of projection: All the projections can be removed except the last one.
Commutativity of selection and projection: Selection and projection are commutative, if and only if the projection list have the selection parameter.
Commutativity of cross product and inner join: The join order does not matter.

Outline of a heuristic algebraic optimization algorithm:

1. Break up SELECT operations into a sequence of selections
2. Use commutativity of selection to move SELECT operations as far down as possible
3. Use binary operator commutativity and associativity to (prority 1) avoid Cartesian products and (priority 2) execute the most restrictive selections first
4. Combine CARTESIAN PRODUCT and SELECT into inner join
5. Break up and move down the projection attributes
6. Identify subtrees that can be executed by a single algorithm

The cost-based optimization is also a very important part when optimizing a query. This estimates and compares the costs of executing a query using different strategies and algorithms. It is better to use it with compiled queries. When a interpreted query occurs at runtime, a more elaborate optimization is indicated for compared queries.

The cost of executing a query include: Cost to access a secondary or disk storage, a computation and memory usage cost and the communication one.

Some cost functions for different algorithms: **SELECT:** *Linear search:* all the file blocks retrieve all records satisfying the condition. *Binary search. Primary index to retrieve a single record:* the cost is one more disk block than the number of index levels. **JOIN:** *Nested-loop. Index-based nested-loop. Sort-merge.*

With some rules of the join operation, many equivalent expressions can be produced. Hence some query trees like the **left-deep join tree**, a binary tree in which the right child of each non-leaf

node is a base relation. We can also find the **right-deep join tree**, the same as the left-deep join tree but the left child. The number of permutations is given by $P(n) = n!$

A **bushy join tree** is a binary tree where the left or right child of an internal node may be an internal node. The number of permutations is given by $P(n) = (2n-2)/(n-1)$. The number of shapes are given by

$$\sum_{i=1}^{n-1} S(i) * S(n-1)$$

There are also **physical optimization** like using index scans when possible for selections, using the selection that results in the smallest cardinality first if the select condition is conjunctive or choosing sort-merge over other join methods when relations are already sorted in the attributes being matched as join.

2 Questions not answered by this text

The heuristic optimization and rewriting is not the same, the former gets the fastest execution plan and the latter is a canonical form and makes optimizer job easier, sometimes considered as a step in the optimization process

3 What has changed since this was written

This text is still valid in the year 2022 as just 6 years has passed since it was written.