Query Optimization by R. Elmasri, S. Navathe

A short summary by Platon Karageorgis

The following summary focuses on the 19th chapter of the book "Fundamentals of Database Systems" and it is about -as the title states- query optimization. The chapter dives deep into how exactly optimization works and describes specifically the rules and the algorithms that come with it. Due to the trickiness of the subject, there are extensive examples in almost every segment of the chapter. The prologue reminds the reader the definition of query optimization and unfolds the plan of the chapter. The author then continues with the first part of the chapter, which is the heuristic rules that are applied in query optimization. A heuristic's rule's primary concern is to alter the interior representation of a query without changing the result. This is a very significant topic in query optimization, because a small change in the order of actions that need to take place can make a big difference in the total cost. For example, as the author outlines, if there is a Cartesian product calculation inside the query, and the query also has a selection that causes a crucial reduction to the total rows, then it would be a waste if the Cartesian calculation took place first. In general, the SELECT and PROJECT operations are critical when it comes to the downsizing of a table or a file. Moreover, the author introduces the definition of a query tree, a data structure that comes along with many relational algebra expressions. In a query tree, the calculations start from the leaf nodes of the tree, and the products of the computations are extracted from the fathers of the nodes. The query tree finishes when the root node is executed. When a query starts being processed, a temporary query tree is created, which is afterwards altered many times until the cost has been significantly reduced. It is important to mention that the tree that the optimizer chooses is not the optimal since that would require a lot of time, so the optimizer is mainly looking for a solid and efficient solution. The author along with the definitions, includes a variety of transformation examples and figures, which assist the reader to comprehend the applied version of query optimization. The key point from the examples is the transformations that take place and some will be discussed shortly. Besides the example of the Cartesian product mentioned before, we achieve cost minimization by prioritizing selections that focus on primary key columns, because the result will not be more than one tuple. In addition, we can replace Cartesian products with join operations. The author, continues the analysis, by revealing the rules that all these transformations have to comply with, in order to preserve the result of the initial query. Some examples would be simple cascades, commutative methods or De Morgan laws. Subsequently, he connects these rules with the previous example by examining thoroughly how those rules would apply in that case. Furthermore, there is a small reference to materialization, pipelining and

their influence in the query execution plans. Additionally, the author brings up the topic of cost optimization. The query optimizer, before deciding on a query execution plan, besides applying heuristic rules, has the task to evaluate and compare the cost of a query, by summoning a variety of algorithms. Someone should not forget though, the query plans have a limit because this process takes time. This technique that was just discussed is called cost-based query optimization and due to its challenging nature in order to apply, rules and methods have been constructed. Moreover, there is a discussion on the different types of costs in a query, such as the input-output or IO's and the CPU costs. It is important to mention the difference in large databases with smaller ones. In large databases, the primary goal is the reduction of the access cost secondary storage -the IO's- and in smaller ones, the target is to minimize the CPU cost since they have a small amount of data and they do not need secondary storage. The author also states the importance of histograms in databases. In the next paragraph, the algorithms that are used broadly in query optimization are analyzed, such as brute force or binary search and they are accompanied by examples that implement them. The different outcomes are compared, in order to choose the optimal algorithm and the reader gets an idea, on how to put each algorithm into effect. Furthermore, the term of join selectivity is introduced. Join selectivity, is the ratio of the amount of tuples, divided by the Cartesian product. This is used by algorithms like nested loop join, in order to get more precise cost functions. After discussing more examples about cost minimization, the author returns to the topic of query trees and analyzes the difference between left-deep trees, right-deep trees and bushy trees. The left-deep are strongly preferred by the public, since they fit in greatly with algorithms and they can produce fully pipelined plans. Along with the query trees, the term of physical optimization is examined. That is the need of an operation to get more information at a physical level, in order to make a decision. There are two ways to approach this issue. The top-down and the bottom-down approach. The only difference is, that the iteration is the opposite and both solutions can offer in cost minimization. Moreover, the definition of dynamic programming is presented, which is an optimization method that focuses on problems that can be divided to sub-problems and calculates the results recursively. Last but not least, the author provides an extensive example of cost-based query optimization.

Which questions aren't answered by this text?

All in all, this chapter was very clear. Every part of the theory was accompanied with examples that clarified almost everything that the author introduced.

What has changed since this was written?

This chapter is consisted by theory and applied examples based on the theory so there are no changes since this was written, the chapter is very relevant to this day.