Access Path Selection in a Relational Database Management System

A short summary by Platon Karageorgis

This paper is focused on the analysis of access path selection and how is it performed by System R, an experimental database system developed by IBM San Jose research laboratory. The paper is divided in seven parts along with introduction and conclusion. To begin with, the author starts by explaining shortly the procedure of an SQL query, separating it in the four known parts, parsing, optimization, code generation, execution and how each one of them contributes. In a nutshell, a query is first and foremost parsed and if it does not contain errors, the optimizer after examining name variables and statistics, is responsible for choosing an access path that minimizes the final cost. After a plan is selected, the code generator converts it to executable code, which is finally executed by the RSS, the topic of the following part. The RSS is a component of System R's system and its duty is storaging. Relations are saved inside it as a group of tuples, whose columns are physically contiguous. Moreover, there are two ways that a tuple can be read from the system. The first is a RSS scan and the second one is an index scan. The index scan, needs an index generation by System R in order to function and there is a separate process, between clustered and non-clustered index. Although these methods are different, they both have the ability to fetch a set of arguments that are significant for a query called SARGS, in case that a set like that exists. A sargable argument, is one of the type "column comparison-operator value" and is illustrated as a Boolean expression of such arguments in disjunctive normal form. The author then moves on to the fourth part, which analyzes the cost of a single relation access path. The total cost is derived from a formula, executed by the optimizer that calculates the final I/O and CPU cost of a query. Furthermore, the optimizer examines the WHERE tree and searches for matches between the access path and the SARGS, since the tree is in conjuctive normal form. Also, the optimizer inspects statistic results stored into System R's catalogs, in order to choose a selectivity factor for each Boolean factor in the argument catalogue. Moreover, the terms of query cardinality and interesting order are introduced, in order to explain how the cheapest plan is generated. In a few words, the query cardinality refers to the uniqueness of data values contained in a particular attribute of a dataset table, while an interesting order is an order of attributes that is mentioned in a GROUP BY or ORDER BY clause in the query. The cheapest path is obtained, after the comparison of a generated unordered path cost, along with the cost of the sorting of its cardinality tuples and a generated path with an interesting order. In the fifth part, the author examines the access path of nested loop join and merge join. A nested loop join is an algorithm that joins two sets by using two nested loops and merge join is an algorithm that joins two sets that are sorted on the join column. The author explains that these algorithms are applied easily to n-way joins, the same way that they are executed with 2-way joins and underlines that the order of the operations make an immense cost difference. The plan is developed based on heuristics and there is an extensive analysis that will be briefly mentioned. Initially a tree of feasible plans is created and is followed by the computation of each relation's cardinality. A solution is consisted of a sorted list of the relations about to be joined, along with the join method for each case and a procedure showing the relations will be accessed. In case a GROUP BY or an ORDER BY exists, the rules of interesting order apply. The whole cost calculation is demonstrated by an example accompanied by illustrations. Finally, the author mentions nested queries, which are queries that appear as arguments of another query. If the interior query uses a variable from the outer query, then it is called correlation subquery. The order the system reads the queries shifts in an occasion like this one in order to avoid errors. Lastly, the author closes the paper by concluding the key points discussed and winds up that DBMS can keep up with non-procedural query languages with quite productively, in comparison with those supporting the modern more procedural languages.

Which questions aren't answered by this text?

Even though this paper is more advanced and could consider most of the terms known, it does mention even briefly most of them. The text is well-written and the fifth part which is more practical, is followed by an example that helps the reader. I did not feel that something is missing.

What has changed since this was written?

This paper was published in 1979 so probably something should be out of date. The storaging system must have changed with RAM memory and flash disks. On the other hand the query processing, the cost calculation and the rest of the operations discussed, are most likely still the same.