Strategies for Query Processing by R. Elmasri, S. Navathe

A short summary by Platon Karageorgis

The following summary focuses on the 18th chapter of the book "Fundamentals of Database Systems" and it is about -as the title states- query processing. The chapter explains the whole procedure and occasionally elaborates some topics that have a deeper theory behind them. Query processing, begins with the scanning, which is responsible of detecting the main attributes of the query, and is followed by parsing and validating. Parsing is no more than the syntax check, while validation is responsible for locating the variables being used in the query. After all is said and done, the query reaches the stage where the query optimizer is in charge. The optimizer should now choose one of the query plans that have been created by the system's algorithms, and there are two conventional ways of doing that. The first approach, is based on heuristic rules that usually arrange the tasks in the execution plan, and the latter is based on cost-estimation and targets on cost minimization. This is an ideal point to mention that the query optimizer, does not have the luxury of spotting the optimal solution every time, because that would significantly affect the total time of the calculations. The most common case, is to find a plan that is productive but might not be the best one. Before moving on to tougher topics, the author outlines the standard SELECT-FROM-WHERE query shape and the function of the join operation followed by examples and special cases like the semi-join. This is commonly found on nested queries and has a specific goal which is to unnest subqueries that have operators like EXISTS, ALL, IN etc. Furthermore, there is an extensive analysis of the SELECT operation. Initially the difference of table or file scan and index scan is reminded to the reader, and then the author mentions every possible plan for selecting. Summarily, it starts with simple techniques such as the standard brute-force method and binary search, which are followed by different hypothetical plans that require different types of indices, such as the clustered index on a key or a non-key attribute. Moreover, it tries to include a bit more complex cases like the disjunctive selection, which is more or less a logical OR operation, that requires every segment of the statement to have an index or any kind of access path, in order to be able to avoid a brute-force algorithm, and therefore a more expensive solution. In the next paragraph, the author moves on to a more compelling subject that uses the previous information accompanied by some definitions. He attempts to enumerate and explain the join techniques. To be more specific, the block nested loop join along with the indexed nested loop join, hash join and sort-merge join are examined. The analysis is followed by the code of each algorithm and also includes several important aspects of the each algorithm that make a difference in the total cost. For example, it justifies along with an example, why it is necessary for the table with the

least rows to be on the outer loop in the nested-loop join and how the sort-merge join algorithm can deliver with just one table read. In addition, it explains why there is a need of M+1 buffers where M is the number of partitions and reveals the difference if the smallest table does not fit in memory. In the next paragraph, the author sets a different course, as he introduces the definition of pipelining, which is about merging operations in order to reduce the cost by eluding the writing of temporary results. The main upsides of pipelining is, as stated already, the reduction of the cost followed by the faster return of the tuples to the user while the rest of the calculations are still underway. In the final part, the author focuses on the contribution of parallelization in the topic of query processing. Common methods like shared memory, shared-disk and most importantly shared nothing are introduced to the reader and are partly compared. Subsequently, he tries to link them to the algorithms that were discussed earlier and presents how they can work with parallelization. The join operation is again the main protagonist. Join is split into n joins, which run in parallel in n processors and ultimately merge into the result, and this can happen with numerous algorithms like equality-based partition join or parallel partitioned hash join. Last but not least, the definitions of intraquery and interquery feature, as the author attempts to link the pipelining part with parallelism.

Which questions aren't answered by this text?

In general the analysis is adequate as it fits every part of query processing in a relatively short description. I feel that I need to state though that the author could have included more information in some specific points. For example, I might be familiar with relational algebra, but if I weren't I would not be able to comprehend fully the examples given at the beginning of the chapter. Moreover, I think there should be more examples on the parallelization paragraph, as it is more advanced than the rest, but the explanation is not particularly extensive.

What has changed since this was written?

I do not have the applied expertise to know if in modern databases these techniques have changed or have been improved, but I have the intuition that they should be more or less the same. The chapter focuses mainly on theory that surely can be extended as the years go by, but I think the case of a theory method being debunked should be quite rare.