# 1

# Introduction

Once a stand alone application databases emerged as an integral part of today's operating systems, be it Linux that comes with the complete Postgres, or future releases of Microsoft Windows have been announced to include some native database support. Other vendors of operating systems announced similar developments. This strategy does not come as a surprise but reflects the enormous penetration of almost all fields of computing by database technology over the past years.

Practically all major applications that involve some kind of data management use database back-ends via application programming interfaces or query language interfaces using standard query languages like SQL. In such an architecture, applications determine and retrieve relevant data sets by posing queries that are ran against the database. Figure 1.1 shows a prototypical architecture of a query processor and its coupling. Let us briefly review the single steps it takes to process a query. The query is submitted in a declarative form in that properties and constraints the data must fulfill are specified but no information how to retrieve the data from the storage is suggested. First, the query is rewritten by a preprocessor, which simplifies the original SQL expression and transfers it into some sort of internal representation. This representation is passed on to the query optimizer who's task is to find a cost effective procedural execution plan for the query. Unlike the declarative query, the execution plan is a procedural description how to retrieve the data. It is composed of operators of the relational algebra which implement the standard set operators but also provide extended functionality like sorting etc. The resulting plan is evaluated by the execution engine which retrieves the data from the storage. Finally, the result data is returned to the client application.

Query optimization is a central task in the processing cycle—selecting an appropriate plan is immensely performance critical. However, the query optimization problem is known to belong to the class of NP-hard problems. Also often referred to as *intractable*, these problems defy any efficient algorithm for a solution. Its financial volume and the widespread use of databases render query optimization one of the most important and most

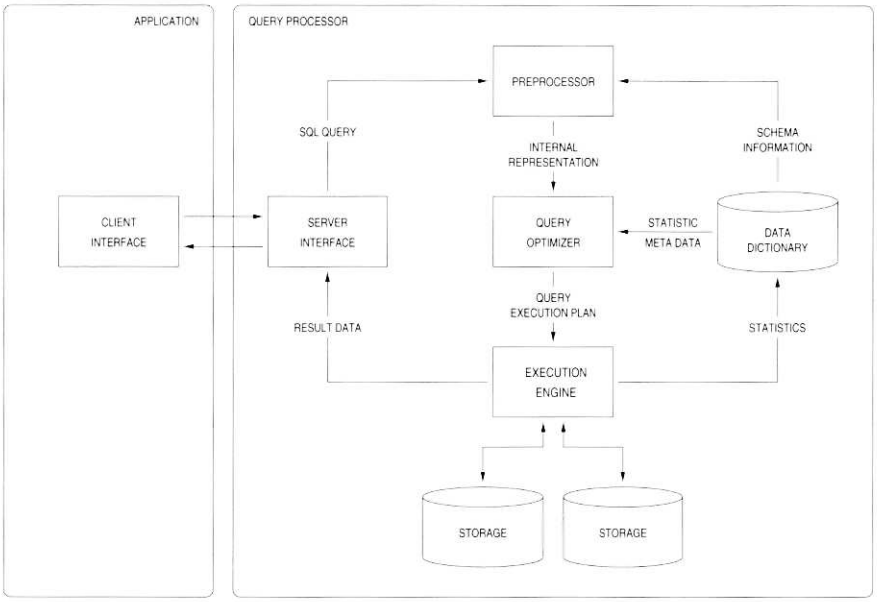


**Figure 1.1.** Outline of the query processing architecture

frequently tackled NP-hard problems in today's computing. Consequently, it received continuous attention ever since. The proposed techniques for approaching this problem reach from exact methods like dynamic programming [SAC+79] to genetic algorithms [BFI91, SMK97], from various heuristics [KBZ86, SI93] to randomized search strategies like Simulated Annealing [IK90] to mention just a few. Interestingly, only a few heuristics and the exact methods, both in form of dynamic programming and transformation based frameworks, made it all the way into commercial products.

This reluctance—partly based on the monopoly like position of some vendors in the field—has lead many researchers to consider the problem solved by agreeing that typical instances will not exceed the capabilities of exact methods. Especially since the projected query sizes that would require capabilities to optimize large and even very large join queries as predicted a decade ago in the wake of object-oriented databases and decision support systems failed to come.

Nevertheless, the requirements in terms of query size grew gradually but steadily and together with an ever increasing variety of operator implementations today's application often reach the limits of exact optimization methods indeed and even exceeded them. The techniques to face those

challenges reach from restricting the search space to certain kinds of processing plans at the expense of possibly failing to find the optimal and putting up with a sub-optimal yet acceptable plan, to ignoring the problem at all. Therefore, practitioners in the database development business consider the problem anything but solved, as I learned from my visits at Microsoft's SQL Server Group and IBM's Almaden Research Center.

But not only the technical requirements changed, the market situation did so too. With new fields of applications like the world wide web which opened a whole new market segment and new players in the field the competition was enlivened which in turn contributed to a higher readiness to adapt research results much quicker than in the past. Ideas of optimization techniques like randomized algorithms could now possibly fall on fertile ground as opposed to a few years ago.

However, practitioners are cautious with deploying non-exact methods mainly because the records of such algorithms are slightly tainted by inconsistent results published so far. Research papers in this field typically proposed a costing technique, and a framework of a randomized algorithm. The performance of the algorithm would be assessed by corroboration with an individually assembled set of queries—usually comparing the proposed method with other algorithms of this category or techniques used in previous work. However, different cost models and different queries as well as different parameters for the algorithms were used for the comparisons leading not seldom to results contradicting previous research, see e.g. [SG88, IK90]. Such proceeding rendered the superiority of the newly proposed techniques rather a matter of believe. Especially intuitive explanations seem to run the risk of fallacious conclusions.

With regard to a practical deployment of those techniques we need deeper insight in both the principles of the particular algorithms and their mode of action as well as—and this might be of even higher importance—into the basic properties of cost based query optimization and the possibilities of exploiting those features. To this end we develop a search space analysis based on the occurring cost values, that is, we essentially scrutinize the ratio of good to bad solutions to the problem. This approach has been inspired by two recent trends in the field of combinatorial optimization yet independent from query optimization. One is concerned with new ways to determine a problem's difficulty apart from its theoretic worst case complexity, the other deals with the applicability and fundamental properties of blind search algorithms. We briefly describe the highlights of both in the following.

Cheesman *et al.* in a noteworthy work detailed that the typical characteristic of a problem's difficulty by means of classic complexity theory often proofs too coarse a measure [CKT91]. In other words, typical instances of some NP-hard problems are fairly easy to solve despite their theoretical worst case complexity. Prior to this work a similar claim has been made by Turner concerning the very particular problem of $k$-colorability of graphs [Tur88]. The concept of phase transition, according to which a

problem is characterized by a parameter for which two disjoint ranges can be identified—for values within one of the ranges the problem is almost always easy to solve, for values of the other range, the problem is usually hard—has been applied to a variety of decision problems. As the most prominent example, satisfiability received special attention and the search for the precise numerical value where the phase transition and thus its difficulty toggles has meanwhile become a discipline on its own. While this concept takes the credit of having revived the discussion of a problem's "difficulty", its applicability to problems other than decision problems is unclear. Cheesman *et al.* present a shot at the Traveling Salesman Problem as one of the traditional and probably best understood optimization problems, however, the results are disputable. Though close relatives, decision problems differ from optimization problems when it comes to the notion of difficulty. In the former, there is only one correct answer an algorithm has to find—yes or no. In practice however, optimization problems are of an approximative nature. In most cases a solution close to the optimum is *good enough*. So the difficulty depends on the optimization goal, i.e., it is a function of the distance between the minimal quality required and the optimal solution.

The second trend is the currently vigorously fought debate about the efficacy of blind search algorithms in general. The label blind search covers all those optimization algorithms that do not take advantage from any knowledge about the problem itself but rather require an abstract framework of manipulators like transformation rules and a mechanism to assess the quality of a single solution. The algorithm then tries to find acceptable solutions only by using the manipulators controlled by the feedback from the quality function. Genetic Algorithms and Simulated Annealing are typical representatives of blind search techniques. The No-Free-Lunch theorems by Wolpert and Macready proof that we cannot expect any such algorithm to outperform any other algorithm of this class on average. In other words, if genetic algorithms are the best optimization technique for problem A, then there exists a problem B where another algorithm performs better. While a powerful construct by themselves—provoking an enormous paper trail—those theorems unfortunately do not provide any clues as to what algorithm should be used for a particular problem.

Clearly, query optimization is tightly intertwined with both those fields, the difficulty of NP-hard problems on the one hand, and the foundations of the applicability of blind search algorithms on the other. Thus, this work is concerned with a synthesis of the different concepts not only with respect to its application to query optimization but also providing deeper insight in previous work, explaining the discrepancies observed.

## 1.1 Research Objectives

Combinatorial optimization problems in practice usually differ from their synthetic theoretical counterparts as e.g. described in [GJ79] in that their objective function is much more complex. In the case of query optimization, a cost function of industrial standard covers a multitude of parameters that are intended to describe both the state of the database and the hardware the query is processed on. The accuracy of the cost model is obviously vital to the successful functioning and therefore belongs to the very core of proprietary code in commercial databases. The more complex the cost model gets, the more difficult its analysis becomes. Commercial cost functions cannot possibly be captured by practical and useful mathematical models and so a simplified, tangible version is needed. We will present several models and discuss the question as to whether the analysis of a simplified model provides insights that are also valid for its complex counterparts. This question is not only of importance for our further considerations but overshadowed most of previous research in this field. We will see that cost distributions, that is, the statistical distribution of cost values in the search space, are characteristic for an optimization problem. Not only do those distributions provide an intuitive measure of difficulty by detailing the ratio of good to bad solutions, they are also the key to an analysis of applicability of certain algorithmic principles and techniques. A large part of this work is concerned with techniques to obtain cost distributions from both the simple models and a commercial database system. Contrasting these results with each other provides also means to validate the simplifications.

Once obtained, we will scrutinize the distributions in order to identify the major features and their generality, i.e., we address the question as to whether the features found depend on the particular instance and if so, to what degree? The question central to this work is: How difficult is query optimization from a practical point of view? We will try to answer this question with respect to different optimization algorithms and the algorithmic principles deployed. Stated differently, we investigate which methods are the most promising ones.

Since we base our research on cost distributions only, the results are not necessarily restricted to the problem at hand but offer the possibility of transfer to other combinatorial optimization problems.

## 1.2 Organization of this Thesis

This work consists of two major parts. First, after introducing a basic model for the problem in Chapter 2, we review costing techniques and present a brief overview on related work. To obtain cost distributions by enumeration and sampling we develop different techniques that reach from enumeration methods for labeled binary trees and non-isomorphic process-

ing trees to techniques for transformation-based optimizers in Chapter 3. The latter has been implemented in Microsoft SQL Server which allows the analysis of cost distribution in a holistic query optimization context rather than in restricted scenarios which capture for example join ordering only. We conclude the first part with a discussion in Chapter 4 of the cost distributions found and the question of the difficulty of the problem in Chapter 5.

The second part is devoted to the consequences which arise from the preceding results. In Chapter 7 we present a close look at random join ordering and a similar analysis for evolutionary algorithms is given in Section 6. Both Chapters also re-evaluate previous work on the respective topics. The second part is completed with a look at biased sampling of join orders, uniform sampling in the context of unrestricted query optimization, and a practical take on random query optimization in Chapter 8.

Chapter 9 finally summarizes this thesis and contains suggestions for further research.