Query Optimizer Plan Diagrams: Production, Reduction and Applications

Jayant R. Haritsa

Database Systems Lab, Indian Institute of Science
Bangalore 560012, INDIA
haritsa@dsl.serc.iisc.ernet.in

***Abstract*—The automated optimization of declarative SQL queries is a classical problem that has been diligently addressed by the database community over several decades. However, due to its inherent complexities and challenges, the topic has largely remained a "black art", and the quality of the query optimizer continues to be a key differentiator between competing database products, with large technical teams involved in their design and implementation.**

Over the past few years, a fresh perspective on the behavior of modern query optimizers has arisen through the introduction and development of the "plan diagram" concept. A plan diagram is a visual representation of the plan choices made by the optimizer over a space of input parameters, such as relational selectivities. In this tutorial, we provide a detailed walk-through of plan diagrams, their processing, and their applications.

We begin by showcasing a variety of plan diagrams that provide intriguing insights into current query optimizer implementations. A suite of techniques for efficiently producing plan diagrams are then outlined. Subsequently, we present a suite of post-processing algorithms that take optimizer plan diagrams as input, and output new diagrams with demonstrably superior query processing characteristics, such as robustness to estimation errors. Following up, we explain how these offline characteristics can be internalized in the query optimizer, resulting in an intrinsically improved optimizer that directly produces high-quality plan diagrams. Finally, we enumerate a variety of open technical problems, and promising future research directions.

All the plan diagrams in the tutorial are sourced from popular industrial-strength query optimizers operating on benchmark decision-support environments, and will be graphically displayed on the Picasso visualization platform.

I. Introduction

Modern database systems employ a *query optimizer* module to automatically identify the most efficient strategy for executing the declarative SQL queries submitted by users. The efficiency of the strategies, called "plans", is usually measured in terms of query response times. Optimization is a mandatory exercise since the difference between the cost of the best execution plan and a random choice could be in orders of magnitude. The role of query optimizers has become especially critical during this decade due to the high degree of query complexity characterizing current data warehousing and mining applications, as exemplified by the TPC-H and TPC-DS decision support benchmarks [26], [27].

The deep complexities and challenges of database query optimization are well-documented [8], resulting in the area largely remaining a "black art". Consequently, the quality of the query optimizer continues to be a key differentiator between competing database products, with large R & D teams involved in their design and implementation.

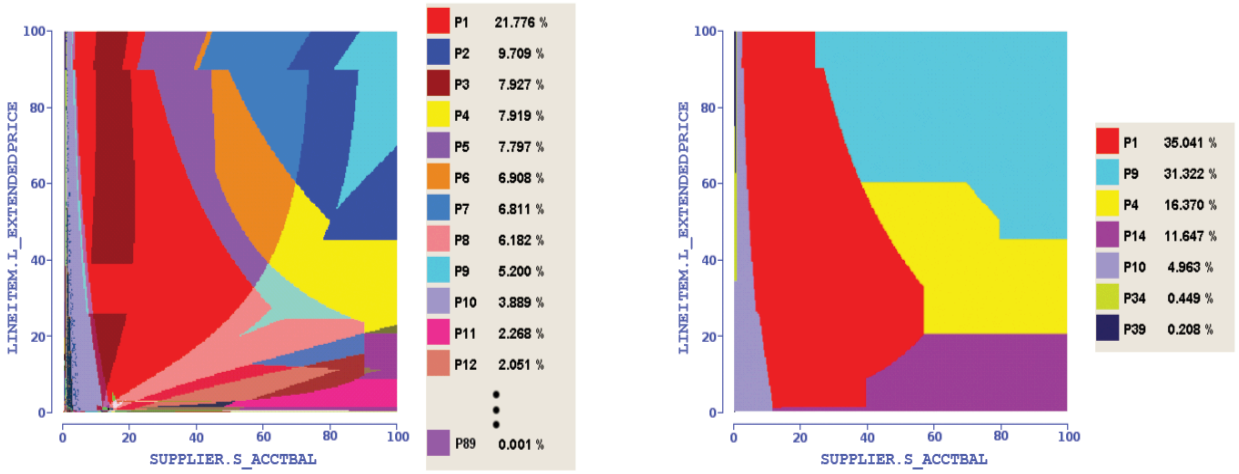
Plan Diagrams. Over the past five years, through a series of VLDB conference papers [23], [16], [17], [14], [1], [18], we have created a fresh perspective on the behavior of modern query optimizers by developing the notion of a *"plan diagram"*. Specifically, a plan diagram is a *visual* representation of the plan choices made by the optimizer over an input parameter space, whose dimensions may include database, query and system-related features. In a nutshell, plan diagrams pictorially capture the geometries of the optimality regions of the parametric optimal set of plans (POSP) [19].

To make these notions concrete, consider the parametrized SQL query template, QT8, shown in Figure I, which is based on TPC-H Query 8. The template defines a relational *selectivity space* on the SUPPLIER and LINEITEM relations, with the selectivity variations specified through the s_acctbal :varies and l_ extendedprice :varies predicates, respectively.

```
select o_year, sum(case when nation = 'BRAZIL' then volume
       else 0 end) / sum(volume)
from (select YEAR(o_orderdate) as o_year, l_extendedprice
       * (1 - l_discount) as volume, n2.n_name as nation
   from part, supplier, lineitem, orders, customer,
        nation n1, nation n2, region
   where p_partkey = l_partkey and s_suppkey = l_suppkey
   and l_orderkey = o_orderkey and o_custkey = c_custkey
   and c_nationkey = n1.n_nationkey and n1.n_regionkey =
   r_regionkey and s_nationkey = n2.n_nationkey and r_name
   = 'AMERICA' and p_type = 'ECONOMY ANODIZED
   STEEL' and
   s_acctbal :varies and l_extendedprice :varies
   ) as all_nations
group by o_year
order by o_year
```

Fig. 1. Example Query Template: QT8

The associated plan diagram for QT8, produced on a popular commercial database engine, is shown in Figure 2(a). In this picture, each colored region represents a specific plan, and a set of 89 different optimal plans, P1 through P89, cover the selectivity space. The value associated with each plan in the legend indicates the percentage area covered by that plan in the diagram – the biggest, P1, for example, covers about 22% of the space, whereas the smallest, P89, is chosen in only 0.001% of the space.



(a) **Plan Diagram** (b) **Reduced Diagram (Threshold λ = 10%)**

Fig. 2. Sample Plan Diagram and Reduced Plan Diagram (QT8)

As is evident in Figure 2(a), plan diagrams can be surprisingly complex and dense, with a large number of plans covering the space – several such instances, spanning a representative set of benchmark-based query templates on current optimizers, are available at [28].

Applications. Plan diagrams are currently in vogue at a host of industrial and academic sites world-wide. They are employed in a diverse set of applications, including characterizing the behavior of optimizer designs; visually analyzing regression test results; investigating structural differences between neighboring plans in the parameter space; debugging new query processing features; evaluating the variations in the plan choices made by competing optimizers; etc. As a case in point, vivid examples of *non-monotonic* cost behavior in commercial optimizers, perhaps indicative of modeling errors, were highlighted in [23].

Apart from aiding optimizer design, plan diagrams can also be used in operational settings. Specifically, since they identify the optimal set of compile-time plans, they can be accessed at run-time to immediately identify the best plan for the current query without going through the time-consuming optimization exercise. Further, they can prove useful to adaptive plan selection techniques (e.g. [12], [13], [22]) which, based on run-time observations, may dynamically choose to re-optimize the query and switch plans mid-way through the processing. In this context, plan diagrams can help reduce the overheads incurred in determining the substitute plan choices.

The most compelling use of plan diagrams lies, however, in their serving as inputs to *plan-replacement algorithms*, which output new plan diagrams with demonstrably superior query processing characteristics. The novel aspect of these replacement schemes is that they do not entail returning to the exponentially large plan search space – instead, they restrict their attention to the comparatively miniscule parametric-optimal space, thereby ensuring computational efficiency.

For example, given a cost-increase-threshold (λ), the original plan diagram can be "reduced" to a simpler picture that features only a subset of the original plans while ensuring that the cost of any query point does not go up by more than λ percent, relative to its original cost. That is, some of the original plans are completely "swallowed" by their siblings, leading to a reduced number of plans in the diagram. It has been empirically shown in [16] that with a threshold of λ = 20%, the absolute number of plans in the reduced picture invariably comes down to *within or around ten*. In short, complex plan diagrams can be made "anorexic" while retaining acceptable query processing performance. As a case in point, the QT8 plan diagram (Figure 2(a)) can be reduced with just λ =10% to the diagram shown in Figure 2(b), where only 7 of the original 89 plans are retained.

Anorexic plan diagram reduction has significant practical benefits, as described in detail in [16], including quantifying the redundancy in the plan search space, enhancing the applicability of parametric query optimization (PQO) techniques [19], [20], identifying error-resistant and least-expected-cost plans [10], [11], and minimizing the overheads of multi-plan approaches [2], [21].

A particularly potent use of plan replacement schemes is in addressing the long-standing problem of *selectivity estimation errors* [24], which result in poor plan choices at run-time. Our study [17] indicates that a substantial fraction of optimizer choices may be improved by identifying substitute POSP plans that are comparatively *robust* to such errors, without materially sacrificing query performance in the absence of errors.

Finally, in our most recent work [1], we have shown that it is indeed feasible to successfully internalize, in the query optimizer core, the offline plan-replacement notions discussed above. The end result is an intrinsically improved optimizer that *directly* produces high-quality plan diagrams in an online fashion, rather than as a post-processing step.

II. Tutorial Stages

In this tutorial, we aim to provide the audience with a detailed walk-through of plan diagrams, their processing, and their applications. The presentation is organized in six stages, described below.

A. Stage I: Plan Diagram Analysis

We begin by showcasing the intriguing plan diagrams obtained with popular industrial-strength optimizers in database environments based on the TPC-H and TPC-DS benchmarks. In particular, we provide compelling evidence that plan diagrams often appear similar to *cubist paintings*, with a large number of plans covering the space and possessing optimality regions characterized by highly intricate patterns and irregular boundaries. These tessellated patterns include speckles, stripes, blinds, mosaics and bands, while the irregular boundaries suggest the presence of strongly non-linear and discretized cost models. The diagrams also highlight that the fundamental assumptions underlying the PQO research literature rarely hold in practice. Even worse, our study has thrown up examples of non-monotonic cost behavior where increasing result cardinalities decrease the estimated processing cost, perhaps indicative of modeling errors.

B. Stage II: Plan Diagram Production

We then explain how plan diagrams can be directly produced using the features generically available in the APIs of query optimizers. While these techniques are satisfactory for low-dimension and low-resolution diagrams, they become computationally infeasible when scaled to higher dimensions and resolutions. To address this issue, we present powerful approximation techniques that provide highly accurate plan diagrams while incurring only minor overheads compared to the exhaustive approach. We also prove that if optimizer APIs were extended to provide, in addition to the cheapest plan, the *second-best* plan also, then *zero-error* diagrams can be produced with only around 10% overheads.

C. Stage III: Anorexic Plan Diagrams

Next, we show how complex plan diagrams can almost always be reduced to much simpler "anorexic" pictures, featuring only a few plans from the POSP set, without materially affecting the query processing quality. Our first reduction technique is based on a conservative upper-bounding of plan costs and can be applied to generic optimizers. The second, and more powerful, option leverages a "foreign plan costing" (FPC) feature now available in high-end optimizers, wherein plans can be costed *outside* of their native optimality regions.

We investigate the plan diagram reduction issue from theoretical, statistical and empirical perspectives. The analysis shows that reduction is an NP-hard problem in general, and remains so even for interesting constrained variations. We then design an online greedy reduction algorithm with tight and optimal performance guarantees, whose complexity scales linearly with the number of plans in the diagram. Finally, statistical estimators that accurately predict the best tradeoff between the query processing quality and the reduction in plan diagram cardinality are constructed.

D. Stage IV: Robust Plan Diagrams

We then turn our attention to the chronic problem of selectivity estimation errors faced by database systems, and explain how the plan diagram reduction scheme can be extended to identify plans that are comparatively robust to such errors. The extension is based on a generalized mathematical characterization of plan cost behavior over the parameter space, which lends itself to efficiently establishing guarantees on the behavior of the substitute plans as compared to the optimizer's standard choices. In particular, we prove the powerful result that the behavior on the *corners* of the parameter space can be used to deterministically predict the behavior *throughout the rest of the space*, resulting in efficient replacement strategies.

A particularly attractive feature of the robust plan diagrams is that the replacements are chosen such that they "never materially harm, but often significantly help", with respect to the original choices of the optimizer. That is, the replacements are always *safe* and often robust, guaranteeing that the behavior of the replacement plan diagram never compares unfavorably with that of the optimizer's own plan diagram.

E. Stage V: Optimizer Integration

We then present a modified query optimization algorithm that, by judiciously expanding the set of candidate plans retained during the optimization procedure, directly incorporates the offline diagram post-processing techniques within the standard optimization framework. The end result is a new query optimizer design that delivers a small and select set of robust plans to execute user queries, and does so in spite of completely lacking the global behavioral information available to the offline algorithms. A prototype implementation has been successfully carried out on the PostgreSQL [29] system.

The concepts and algorithms presented in the above five stages will be pictorially demonstrated on the Picasso query optimizer visualization tool [28]. Given a parametrized query template such as QT8, Picasso automatically generates a suite of diagrams, including plan diagrams and reduced plan diagrams, that comprehensively capture the plan choices made by the optimizer over the parameter space. Our examples will cover a representative set of popular commercial and public-domain query optimizers, operating on the TPC-H and TPC-DS benchmark environments.

F. Stage VI: Future Research Directions

In the concluding part of the tutorial, we will outline a set of open technical problems and future research directions. Sample problems include the following:

a) Plan Diagram Density Classifier: Currently, only after a plan diagram is produced do we know whether it is *sparse* (few plans) or *dense* (several plans). It would be extremely useful to develop a predictor for the density of the diagram *prior* to production. This objective could be treated as a data mining problem involving classification, and the associated

feature vector is likely to have to include aspects of the query template, the database engine, and the schematic/statistical meta-data. If the boolean (sparse or dense) predictor works out successfully, a follow-up challenge would be to extend it to explicitly quantify the expected density.

b) Plan Diagram Coloring Mechanism: Currently, unique colors are assigned at random to the various plans featuring in the plan diagram. A semantically richer option would be to color plans in a manner that also reflects the extent of their *structural differences*. For instance, if a pair of plans happen to have the same join order, they should be assigned close shades of a common color. With this new approach to coloring, the plan diagram itself provides a first-cut reflection of the differences between plans as we traverse the selectivity space. To achieve this objective, a semantically consistent plan distance metric has to be first defined, after which an efficient coloring scheme that reflects these differences as closely as possible has to be designed.

c) Query Execution Visualization: While plan diagrams capture the "compile-time" behavior of query optimizers, it would be instructive to also visualize the *run-time* behavior in a similar manner. A first step in this important direction has been taken in [15], and there is ample scope for pursuing these ideas further.

III. Target Audience and Outcomes

The target audience of the tutorial includes researchers, developers and students with an interest in the internals of database engines. The background expected is that of an introductory database systems course covering relational data models, declarative query languages, and query processing techniques. The primary source material for the tutorial is available in [23], [16], [17], [14], [1], [18], complemented by inputs from the rich corpus of literature on query optimization and processing. A sampling of relevant papers is given in the accompanying list of references.

Database researchers can expect to find the tutorial providing a fresh perspective on a classical problem, and serving as a stimulus to work on the development of stable and efficient database engines. This gains significance in light of the recent observation that *lack of robustness can contribute as much as a third to the total cost of ownership for a database system* [25]. From the perspective of system developers and practitioners, the concepts and diagrams presented in the tutorial can serve as potent mechanisms for the analysis, testing and redesign of their systems. Finally, for database instructors and students, plan diagrams are a powerful pedagogical support to help comprehend and appreciate the complexities and subtleties of industrial-strength query optimization, going far beyond the toy examples typically covered in a classroom setting.

Overall, the primary message of this tutorial is that it is indeed feasible to *efficiently produce plan diagrams that simultaneously possess the desirable properties of being online, anorexic, safe and robust*. We are optimistic that this result could play a meaningful role in designing the next generation of database query optimizers.

References

[1] M. Abhirama, S. Bhaumik, A. Dey, H. Shrimal and J. Haritsa, "On the Stability of Plan Costs and the Costs of Plan Stability", *Proc. of 36th Intl. Conf. on Very Large Data Bases (VLDB)*, September 2010.

[2] G. Antoshenkov, "Dynamic Query Optimization in Rdb/VMS", *Proc. of 9th IEEE Intl. Conf. on Data Engineering (ICDE)*, April 1993.

[3] B. Babcock and S. Chaudhuri, "Towards a Robust Query Optimizer: A Principled and Practical Approach", *Proc. of ACM SIGMOD Conf. on Management of Data*, June 2005.

[4] S. Babu, P. Bizarro and D. DeWitt, "Proactive Re-Optimization", *Proc. of ACM SIGMOD Conf. on Management of Data*, June 2005.

[5] S. Babu, P. Bizarro and D. DeWitt, "Proactive Re-Optimization with Rio", *Proc. of ACM SIGMOD Conf. on Management of Data*, June 2005.

[6] P. Bizarro, N. Bruno and D. DeWitt, "Progressive Parametric Query Optimization", *IEEE Trans. on Knowledge & Data Engineering*, 21(4), April 2009.

[7] N. Bruno, S. Chaudhuri and R. Ramamurthy, "Power Hints for Query Optimization", *Proc. of 25th IEEE Intl. Conf. on Data Engineering (ICDE)*, March 2009.

[8] S. Chaudhuri, "An Overview of Query Optimization in Relational Systems", *Proc. of ACM Symp. on Principles of Database Systems (PODS)*, June 1998.

[9] S. Chaudhuri, "Query Optimizers: Time to Rethink the Contract?", *Proc. of ACM SIGMOD Conf. on Management of Data*, June 2009.

[10] F. Chu, J. Halpern and P. Seshadri, "Least Expected Cost Query Optimization: An Exercise in Utility", *Proc. of ACM Symp. on Principles of Database Systems (PODS)*, May 1999.

[11] F. Chu, J. Halpern and J. Gehrke, "Least Expected Cost Query Optimization: What Can We Expect", *Proc. of ACM Symp. on Principles of Database Systems (PODS)*, May 2002.

[12] R. Cole and G. Graefe, "Optimization of Dynamic Query Evaluation Plans", *Proc. of ACM SIGMOD Intl. Conf. on Management of Data*, May 1994.

[13] A. Deshpande, Z. Ives and V. Raman, "Adaptive Query Processing", *Foundations and Trends in Databases*, Now Publishers, 1(1), 2007.

[14] A. Dey, S. Bhaumik, Harish D. and J. Haritsa, "Efficiently Approximating Query Optimizer Plan Diagrams", *Proc. of 34th Intl. Conf. on Very Large Data Bases (VLDB)*, August 2008.

[15] G. Graefe, H. Kuno and J. Wiener, "Visualizing the robustness of query execution", *Proc. of Conf. on Innovative Data Systems Research (CIDR)*, January 2009.

[16] Harish D., P. Darera and J. Haritsa, "On the Production of Anorexic Plan Diagrams", *Proc. of 33th Intl. Conf. on Very Large Data Bases (VLDB)*, September 2007.

[17] Harish D., P. Darera and J. Haritsa, "Identifying Robust Plans through Plan Diagram Reduction", *Proc. of 34th Intl. Conf. on Very Large Data Bases (VLDB)*, August 2008.

[18] J. Haritsa, "The Picasso Database Query Optimizer Visualizer", *Proc. of 36th Intl. Conf. on Very Large Data Bases (VLDB)*, September 2010.

[19] A. Hulgeri and S. Sudarshan, "Parametric Query Optimization for Linear and Piecewise Linear Cost Functions", *Proc. of 28th Intl. Conf. on Very Large Data Bases (VLDB)*, August 2002.

[20] A. Hulgeri and S. Sudarshan, "AniPQO: Almost Non-intrusive Parametric Query Optimization for Nonlinear Cost Functions", *Proc. of 29th Intl. Conf. on Very Large Data Bases (VLDB)*, August 2003.

[21] N. Kabra and D. DeWitt, "Efficient Mid-Query Re-Optimization of Sub-Optimal Query Execution Plans", *Proc. of ACM SIGMOD Intl. Conf. on Management of Data*, May 1998.

[22] V. Markl, V. Raman, D. Simmen, G. Lohman, H. Pirahesh and M. Cilimdzic, "Robust Query Processing through Progressive Optimization", *Proc. of ACM SIGMOD Intl. Conf. on Management of Data*, May 2004.

[23] N. Reddy and J. Haritsa, "Analyzing Plan Diagrams of Database Query Optimizers", *Proc. of 31st Intl. Conf. on Very Large Data Bases (VLDB)*, August 2005.

[24] M. Stillger, G. Lohman, V. Markl and M. Kandil, "LEO, DB2's LEarning Optimizer", *Proc. of 27th Intl. Conf. on Very Large Data Bases (VLDB)*, August 2001.

[25] www.dagstuhl.de/no_cache/en/program/calendar/semhp/?semnr=10381

[26] www.tpc.org/tpch

[27] www.tpc.org/tpcds

[28] dsl.serc.iisc.ernet.in/projects/PICASSO/

[29] www.postgresql.org