Automated Physical Designers: What You See is (Not) What You Get

Renata Borovica Ioannis Alagiannis Anastasia Ailamaki
École Polytechnique Fédérale de Lausanne
1015 Lausanne, Switzerland
name.surname@epfl.ch

ABSTRACT

The explosion of available data in the last few years has increased the importance of physical database design, since the selection of proper physical structures (e.g. indices, partitions and materialized views) may improve query execution performance by several orders of magnitude. Commercial DBMS vendors have recognized this need and offered automated physical design tools as part of their products. These tools use what-if interfaces to simulate the presence of different physical structures and recommend physical designs that minimize the estimated execution time of a given workload. Along with the recommended design, they deliver an estimation of the expected improvement the new design brings.

In this paper, we examine the output of physical designers, i.e., whether what we see as a result of the tuning (the estimation of the improvement) is indeed what we may expect after applying the design (the actual improvement). We evaluate three commercial physical designers by varying their input parameters on real and synthetic data sets. Our results show that all three physical designers exhibit highly unpredictable behavior in certain cases, indicating that there is still significant room for improvement in terms of their predictability and consequently, their quality.

Categories and Subject Descriptors

H.2.2 [**Database Management**]: Physical Design

General Terms

Measurement, Predictability, Experimentation.

Keywords

Physical Design, Databases, Predictability, Evaluation.

1. INTRODUCTION

In recent years, the field of physical database design has become extremely popular and most commercial DBMS vendors nowadays offer physical designers in their products [2, 10, 19]. Physical designers significantly facilitate the tuning process and are an integral part of a broader effort toward fully automated database management systems which aims to: a) decrease the database administration cost and thus, the total cost of database ownership, b) help non-experts to use database systems and c) enable databases to move to a different environment, such as the cloud where database instances are offered as a service.

A typical physical designer tries to solve the following problem: Given a workload W and a set of constraints C (e.g. a storage budget, a time budget), find a set of physical structures or a *configuration* P that minimizes the execution cost for W and satisfies C. The output of a physical designer is a recommendation of auxiliary structures (e.g. indices) selected to boost performance for the given workload and an estimation of the expected performance improvement. The database administrator (DBA) examines the output of the physical design process, verifies the usefulness of the proposed configuration and decides what structures will be created inside the database.

The only insight regarding the usefulness of the proposed configuration is the expected improvement presented by the tool. Changing the physical design is a heavyweight operation, thus an inaccurate estimation regarding the configuration usefulness may lead to ineffective resource allocation; for instance, building an index that occupies large storage space but provides marginal performance improvement. From the DBA's perspective, not getting the anticipated performance improvement has a negative impact on the user experience and may cause loss of trust toward the effectiveness of the tool.

There is a plethora of published work on the topic of physical design [6]; however, it is mainly focused on the performance impact of the recommended solutions, neglecting to examine the accuracy of the designers' estimations. An accurate estimation implies that the recommendation can be adopted with a high degree of confidence, while an inaccurate estimation raises questions of the trustworthiness of physical designers. Therefore, in this paper, we study the predictability of physical designers in terms of how accurately they estimate the effectiveness of their proposed configurations. We base our evaluation on the output of three commercial physical database designers.

Our experiments show that:

- All three physical designers exhibit highly unpredictable behaviour, with discrepancies of up to 92% between the estimated improvement they deliver and the actual improvement databases gain.
- A tight interaction with the query optimizer's cost model and a strong reliance on statistics have a substantial influence on the accuracy of the designer's estimations.

DBTest'12, May 21, 2012 Scottsdale, AZ, U.S.A.

- The presence of update statements in the workload is especially challenging for the designers, making them incapable of accurately modeling the trade-off between the usefulness of proposed structures and the cost of maintaining them.

The rest of the paper is structured as follows. In Section 2, we briefly describe the interaction between physical designers and query optimizers. In Section 3, we present the metrics we use along with the experimental methodology, while we report experimental results in Section 4. Section 5 discusses our findings, and Section 6 offers our conclusions.

2. AUTOMATED PHYSICAL DESIGNERS

During the tuning process, physical designers rely heavily on the query optimizer and its cost model. Therefore, in this section, we discuss the details of this relationship and further show how statistics affect recommendations. Finally, we briefly summarize related work that evaluates the quality of physical designers.

2.1 Interaction with the Query Optimizer

Physical designers invoke the query optimizer using what-if interfaces [12] to simulate the presence of different design structures without materializing them [5]. Such an approach has several advantages: a) low overhead since the examined physical structures are not actually created; b) recommended structures, if implemented, are guaranteed to be used by the optimizer; c) enhancing the optimizer's cost model improves query optimization from which physical designers further benefit. The drawback is the reliance on a single source of truth, the predictions of the optimizer. Optimizers are known to be error prone due to a multitude of factors [8]. They rely on an embedded cost-model that does not entirely represent reality, and on various data statistics that might be unavailable or inaccurate.

2.2 Importance of Statistics

Statistics are an integral part of the optimization process since the optimizer uses them to estimate the number of output tuples of every operator in a query plan [1]. The result size of a query that involves predicates on multiple attributes depends on the joint data distribution of the attributes, i.e., the frequencies of all combinations of attribute values. Due to the large multidimensional nature of joint distributions and the high number of possible combinations, a direct approximation of the distributions is complex and expensive. To simplify the estimations, commercial systems assume attribute independence [3]. According to this assumption, the selectivity of an operator filtering data on several predicates is calculated as the product of the selectivities for each predicate. In practice, the assumption is often violated, which causes significant cardinality under-estimates. The latter generates gross errors in query runtime predictions, which leads to the choice of sub-optimal plans, and finally results in design proposals that hurt performance instead of improving it [13].

Considering the connection between the statistics, query optimizer, and physical designer, it is clear that we cannot simply isolate and assess the quality of physical designers, without taking into account the query optimizer's errors. In this work we explore to what extent the mentioned interactions affect the trustworthiness of physical designers.

2.3 Benchmarking Physical Design

Although several benchmarks for physical design already exist, they all examine designers in the light of performance, while no benchmark examines their predictability. Consenses et al. [9] introduce a framework to evaluate the quality of physical designers while looking at the number of tuned queries that are faster than a given threshold. Although insightful on a workload basis, the metric does not give any insight on query-by-query performance. Therefore, a complementary metric has been proposed, which measures the cost improvement on a query basis [4]. Schnaitter et al. [16] look at minimizing the combined cost of executing a query under a configuration C and the cost of changing the current configuration to C. The last metric is more meaningful in the context of online tuning, while in this work we consider off-line tuning and assume a representative training workload given a priori.

All metrics presented until now are focused on the performance, while we look at the predictability and trustworthiness of physical designers. Hence, our metric provides different insights regarding the quality of physical designers.

3. TESTING FOR PREDICTABILITY

In this section, we describe the evaluation metrics, the workloads and our experimental methodology employed to compare the physical designers.

3.1 Evaluation Metrics

We calculate the predictability of physical designers as the difference between the expected improvement they deliver and the actual improvement databases achieve after applying the proposed physical designs. We present the difference as a percentage over the whole workload. Table 1 summarizes the used metrics. We denote as T_O the workload execution time before the tuning phase, and as T_{AT} the workload execution time after the proposed design has been adopted. We use I_E to express the improvement estimated by the physical designer and I_A to show the actually achieved improvement. We use the metrics to calculate the following formulae:

$$I_A = 100 - \left(\frac{T_{AT}}{T_O}\right) \times 100$$

$$T_{ET} = T_O - \frac{(I_E \times T_O)}{100}$$

$$R_{EE} = \frac{|T_{ET} - T_{AT}|}{T_{AT}} \times 100$$

$$A_{EE} = |T_{ET} - T_{AT}|$$

The relative estimation error (R_{EE}) demonstrates the predictability of a system, meaning the most accurate system is the one having the lowest estimation error.

3.2 Workloads

In our experiments we study two workloads with different characteristics. The first one contains 18 queries from the TPC-H decision support benchmark [17] and we report results for scale factors (SF) 10 and 100 (data set size of 10GB and 100GB). Originally, the TPC-H benchmark consists of 22 queries, while we exclude Q17, and Q20 to Q22 due to their long execution in some of the DBMS. Additionally, to narrow the vast search space in the experiments performed on this benchmark, we restrict the designers to proposing only indices. To increase the confidence in our results, we

Table 1: Metrics descriptions

Metric name	Description
Original time (T_O)	Workload execution time before the tuning phase.
Estimated tuned time (T_{ET})	Physical designer's estimated execution time (with the new physical design).
Actual tuned time (T_{AT})	Actual workload execution time (with the new physical design).
Estimated improvement (I_E)	Physical designer's estimated improvement for the proposed design (%).
Actual improvement (I_A)	The actual improvement with the proposed design (%).
Relative estimation error (R_{EE})	The relative error between T_{ET} and T_{AT} (%).
Absolute estimation error (A_{EE})	The difference between T_{ET} and T_{AT}.

repeat the experiments with this workload 10 times using different predicate values in queries obtained by calling the QGen tool on the default TPC-H query templates. Every reported data point is the average of multiple executions with a standard deviation of less than 5%.

The second workload contains exploratory queries on the NREF database [18]. The NREF database provides a collection of protein sequence data from several genome sequencing projects. It contains 6 tables that together occupy 7GB. The query set consists of 400 combined SELECT and UPDATE statements, from which the select-only workload contains 200 statements.

3.3 Experimental Methodology

All experiments are conducted following the same algorithm. First, we execute the queries in the original database (before any tuning) and measure the workload execution time, which we use as the baseline of our evaluation. Then, we call the physical designers to suggest a new physical design. We call all physical designers with the same input for every series of our experiments. In order to obtain more accurate results from the query optimizer, we manually update statistics after loading the data and applying referential integrity constraints, as well as after applying the designer's recommendations. Once the proposed designs are built in the DBMS, all the queries are re-run and the improvement and the predictability are calculated.

4. EXPERIMENTAL RESULTS

We conduct several experiments to evaluate the predictability of designers and identify to what extent different parameters affect the estimates. In the experiments, we vary the space budget for recommendations, the size of the input database, and the number of queries in the workload (considering the effect of updates as well).

4.1 Experimental Setup

All experiments are conducted using a 2.70GHz AMD Opteron 2384 with two Quad-Core CPUs with 32GB of RAM running Windows Server 2008 R2 (64bit). For storage, we use two 750GB 7200rpm SATA hard disks, configured as a RAID 0 with no caching allowed, and with an average I/O data transfer rate of 90MB/s. Due to legal restrictions the names of the used database systems are not disclosed here and will be referred as *System-A*, *System-B*, and *System-C*. All databases use 2GB of RAM unless stated otherwise and in all our experiments we report cold runs with the caches being emptied between the experiments.

4.2 The Designers' Running Time

In this section, we report the time needed for the physical designers to propose a new set of design structures.

TPC-H: In the set of experiments performed using the TPC-H benchmark, the tuning time is not limited since our goal is to achieve the best possible results. System-A and System-B run for less than 3 min in all the experiments. System-C has been the fastest in getting the response from the designer, but also as we will see in the following subsections the least accurate. Despite the fact that the tuning time is not limited, it only takes 3 sec to complete.

NREF: In the experiments performed using the NREF data set, we set the time budget to 30 min, since the workload comprises 400 statements. The designers of System-A and System-B exploit all the given time, while the one of System-C again returns results in seconds. In this set of experiments, we notice that System-C exceeds the available space budget[1], leading us to believe that it might skip the merge phase in which designers merge design structures together to fulfil space constraints.

Typically, physical designers stop when the solution cannot be further improved or if they exhaust the time budget. In the latter case, it would be useful that physical designers report the distance between the proposed and optimal solution, providing thereby the DBA with the feedback on the quality of delivered solutions [11].

4.3 Impact of Space Budget

In this experiment, we examine the impact of the space budget on the predictability of physical designers. We use a TPC-H database of size 10GB and vary the space budget from 5GB, and 15GB to unlimited space.

We observe that both System-A and System-B exploit almost the whole available space for recommendations. On the other hand, System-C uses only 3.9GB and returns the same proposal regardless of the space budget; thus, we report results only for this proposal. With the unlimited space budget, System-A exploits 23GB, while System-B uses 17GB.

We do not observe any correlation between the space budget, and the predictability of improvement the designers deliver. Table 2 shows the results for the three systems. System-A returns the most accurate estimations of the improvement. For the space budget of 5GB, it estimates an improvement of 46%, while the database achieves an improvement of 57%. An R_{EE} is 26% over the whole workload. By increasing the space budget, System-A becomes more accurate making an R_{EE} only of 1.7% with 15GB, and 3.7% with unlimited space. On the other hand, System-B exhibits completely unstable behavior; an acceptable R_{EE} for the space budget of 5GB, a completely wrong estimation of the improvement for the space budget of 15GB that caused performance degradation of 776% with an R_{EE} of 92%, and an R_{EE} of 67% with unlimited space.

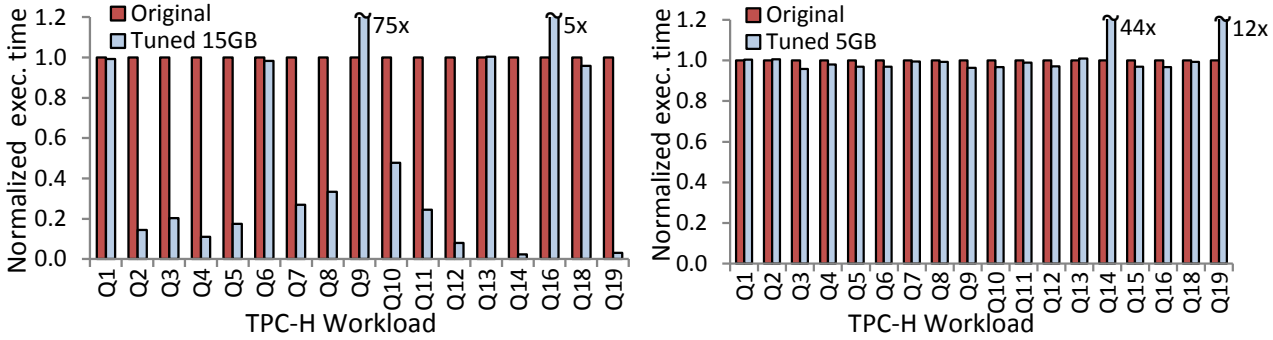
After applying the proposed designs, System-B with 15 GB space budget and System-C with 5GB space budget encounter severe performance degradation. For System-B, Q9 and Q16 run 75 and 5 times slower, while for System-C Q14 and Q19 are 44 and 12 times slower. Figure 1 plots the nor-

[1]20GB is the allowed space budget, while System-C uses up to 32GB of disk space

Table 2: Predictability when using the TPC-H 10GB

	System-A			System-B			System-C
Metrics	5GB space	15GB space	Unbounded space	5GB space	15GB space*	Unbounded space	5GB space
I_E (%)	45.94	63.46	73.32	21.16	37.29	39.9	10.62
I_A (%)	57.13	64.09	74.27	30.96	-776.3	64.1	-219.23
R_{EE} (%)	**26.09**	**1.74**	**3.7**	**14.2**	**92.84**	**67.37**	**72**

* An error in the optimizer's estimates results in 75 times longer execution time for Q9. For the rest of the workload, an I_A is 60%, which gives us an R_{EE} of 57%.



(a) System B (b) System C

Figure 1: Relative improvement with TPC-H SF10

malized execution time for System-B and System-C in these cases. For System-B, we observe that all others queries in the workload benefit from the new physical design. Nevertheless, the longer execution times of Q9 and Q16 prolonged the overall execution 8 times (from half an hour to 4 hours).

The problem appears due to errors in the optimizer's cardinality estimates that favor an index seek over a full table scan. The optimizer opts for such a plan because, by employing the attribute value independence assumption, it underestimates the size of intermediate results. The effect of the assumption is presented in more detail in Section 4.7. We are certainly not the first to identify the problem of cardinality errors and various techniques have already been proposed in literature [15, 14, 7]. Nevertheless, database systems still use the assumption to simplify cardinality computation, and as this heuristic is often violated in practice, it results in significant query optimizers' errors [3].

Discussion. If we exclude the mentioned extreme cases, we notice that the designers' predictions are quite conservative in comparison with the actual improvement databases achieve. One might claim that this is not a problem, as long as new designs improve performance. We do not agree, since an inaccurate estimation regarding the usefulness of a proposed design might mislead the DBA and discourage them completely from implementing such a design.

4.4 Impact of Database Size

To examine the influence of the database size on the predictability of physical designers we conduct additional experiments in which we increase the size of the TPC-H data set from 10GB to 100GB. Proportionally, we increase the bufferpool size from 2GB to 20GB, and vary the space budget from 50GB, and 150GB to unlimited space.

Similar to the previous experiment, we do not observe a trend in predicting improvement. Figure 2 shows how the R_{EE} changes as we increase the recommendation space budget. System-A starts with an R_{EE} of 46% and further improves its predictions with an R_{EE} of 23% in the second and 20% in the third experiment. System-A is the only system that actually achieves performance improvement after implementing the proposed design; an improvement of 61% in the first, 71% in the second and 75% in the third experiment. System-B, due to several long running queries in each experiment, ends up with 15% worse performance in the first case, and 25% and 6% in the second and third case. The reason for performance degradation again lies in the optimizer's cardinality errors that favor index usage over full table scans. Due to the same reason, System-C finishes its execution in twice the time in comparison with the baseline in all three experiments, causing an R_{EE} of 68%.

We additionally note that despite the fact physical designers can substantially boost performance, after some point they can further improve performance only to a marginal extent in comparison with the space they need to use or the time that takes to create all proposed structures[2]. Figure 3 shows the percentage of improvement System-A achieves when increasing the space budget for the TPC-H workload. With the initial budget, the designer achieves an improvement of 57% and 61%, for SF 10 and 100 respectively. When adding the additional space budget equal to the database size, the designer proposes recommendations that further improve performance for another 7% in the former and 10% in the latter case. For improving the design for another 10% in the case of SF 10 and 5% in the case of SF 100, the designer needs additional 8GB and 68GB respectively. Clearly, there is a threshold after which the trade-off between achieving further improvement at the expense of using much more space is not worth. Thus, the feedback on how far the current solution is from the optimal or the information about how many resources the designer needs for achieving additional improvement are attributes that commercial systems vendors should consider including in their tools.

4.5 Impact of Workload Size

In this experiment, we examine the physical designers' behaviour when we increase the size of the workload. We use a select-only synthetic workload, based on exploratory queries on the NREF data set [18]. The workload comprises a set of two, three and four-table joins, in addition to simple

[2]It takes nearly 16 hours to create a configuration proposed by System-B for SF100, when the space budget is unlimited.

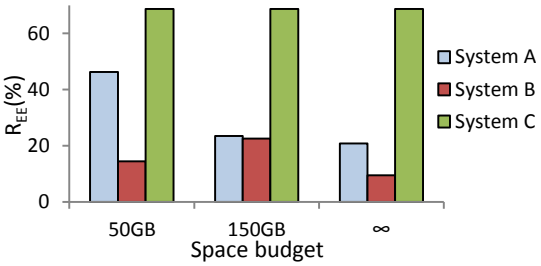


Figure 2: R_{EE} with TPC-H SF100

range queries that read data from just one table and filter it by several predicates. Throughout the experiments we progressively increase the workload size using 20, 50, 100 and 200 queries between different runs. The time budget for designers is set to 30 minutes, and the space budget to 20GB. In the round of experiments conducted on the NREF data set, we do not set any restrictions on the possible physical design structures, i.e., in addition to indices, partitioning and views may also be considered.

Table 3 summarizes the results for the current and the following section. System-A improves performance after applying the proposed designs, making a relative error between 20% and 45% throughout the experiments. For the same setting, the proposed designs bring improvement to System-C with a relative error between 42% and 87%. System-B degrades performance in the majority of experiments, again because it proposes indices whose usefulness is overstated.

Discussion. Unlike the experiment described in Section 4.3, in this experiment we notice exactly the opposite behavior. While the tools are more conservative in the case of the TPC-H benchmark, they are too optimistic in this experiment, since they estimate higher improvements in comparison with what they achieve. Thus, in addition to being inaccurate in their estimations, the tools are also inconsistent across different runs, making it harder for us to anticipate the potential performance improvement we may gain.

4.6 Impact of Updates

In this experiment we augment the select-only workload with a set of update statements on *protein* and *neighboring_seq* tables. Our goal is to exercise the cost model of design tools, since now they have to consider the trade-off between proposing indices that improve performance and maintaining these structures. This can be considered as the hardest case for physical designers.

The results are presented in Table 3 (column 400). System-B is not able to find a design that will improve performance, therefore its error is 0%. System-A and System-C propose a set of structures with an estimated improvement of 58% in the first, and 2% in the second case. Nevertheless, both systems actually deteriorate performance after applying the designs. The reason lies in the long running update operations, since now systems have to reorganize indices created on both aforementioned tables whenever an update operation occurs. System-A in this experiment is the least accurate with an R_{EE} of 65%. System-C is more careful and proposes fewer indices, while it concentrates more on other design structures (e.g. views), which results in an R_{EE} of 9%. Nevertheless, we notice that System-C does not actually respect the space constraints. In this experiment it uses 32GB of space, while the limit is set to 20GB.

Discussion. From this and the set of experiments performed using the TPC-H benchmark, we notice that indices

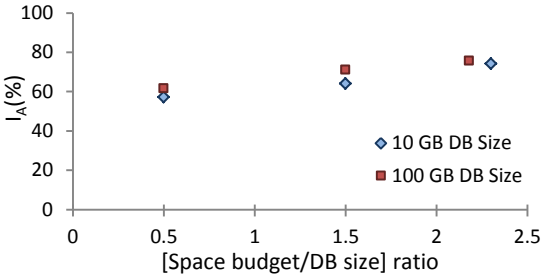


Figure 3: System-A - Actual improvement when increasing the space budget

can have both positive and negative impact on performance. They can boost performance (up to 75% of improvement in our experiments), but can also significantly degrade performance if the overhead of maintaining them is not modeled accurately, or if the optimizer under-estimates the size of intermediate results and hence decides to use indices in queries that are low selective, leading to substantial overheads that random I/O accesses bring.

4.7 Impact of Statistics

A harmful effect of the attribute value independence assumption on the quality of proposed designs is already mentioned in Sections 4.3 and 4.7. The assumption prolongs execution time of the majority of experiments conducted on the select-only NREF workload on System-B. A typical query from the workload is shown below:

```
SELECT p_name FROM protein
  WHERE seq_length BETWEEN 121 AND 3932
    AND table1.last_updated
BETWEEN '12/21/2001' AND '02/11/2002';
```

Even with this simple query we can see a detrimental effect of the attribute value independence assumption. Estimated cardinality of this query is 20.595, while the actual cardinality is 179.763, an order of magnitude more. The wrong estimate mislead the optimizer that an index seek followed by a table lookup for the rest of the columns (not covered by the index) by RowIDs is the cheapest solution, while in reality a full table scan would be much faster. The same situation appears in Q19 of the TPC-H benchmark performed on System-C, where the query optimizer, due to the same reason, makes the cardinality error underestimating the size of intermediate results by three orders of magnitude. As a consequence it decides to use a nested-loop join between tables LINEITEM and PART with three orders of magnitude more tuples than estimated, which finally results in 12 times longer execution time.

Another surprising observation is that for System-B and C performance-wise it is better not to have statistical information at all in some cases, than to have it with the independence assumption. Without statistics on indices, the optimizer chooses the safe option which is a full table scan, and hence it chooses more efficient execution plans. In addition, we notice that the reason why System-A does not fall into this trap is because it proposes creation of statistics on all joint columns from the workload. We tried to manually perform the same task on System-B, unfortunately without success, since the cardinality errors remained. System-C to our knowledge does not support such a command.

Discussion. From everything mentioned so far, it can be concluded that statistics have a major impact on the quality of execution plans and consequently on the predictability of physical designers.

Table 3: Predictability when increasing workload size

	System A					System B					System C				
Metrics	20	50	100	200	400*	20	50	100	200	400	20	50	100	200	400
I_E (%)	94.11	90.29	92.3	81.62	58.62	73.66	50.55	37.39	35.75	0	95.75	90.12	92.35	68.8	2.23
I_A (%)	91.16	87.26	85.7	77.16	-18.3	18.15	-69.6	-60.69	-91.02	0	64.75	53.36	66.62	45.28	-8.13
R_{EE} (%)	**33.35**	**23.75**	**46.15**	**19.53**	**65.02**	**67.82**	**70.83**	**61.03**	**66.36**	**0**	**87.93**	**78.81**	**77.1**	**42.98**	**9.58**

* The number of statements in the workload. 400 represents the update-intensive workload.

5. DISCUSSION

We have seen the substantial influence of the query optimizer's cardinality errors on the quality of proposed designs. Nevertheless, cardinality errors are not only attributed to the presence of joint data distributions. We additionally notice the cases when cardinality errors are high without joint distributions, for instance in queries with the "LIKE" predicate that filters character values. Furthermore, even with accurate cardinality estimates, the cost model sometimes favors an index seek and a table lookup, which is much more expensive than a simple full table scan (e.g. 44 times more in the case of the TPC-H Q14 executed on System-C). The latter implies that the cost model itself has to be refined. For instance, the ratio between random and sequential reads has to be modeled more accurately.

Another concern relates to the user satisfaction when using physical designers. For instance, we have noticed cases when designs are proposed with syntax errors, which caused compilation errors. In the case of System-C, we have even seen how proposed designs violate space constraints, which all can deter users from using the tools.

6. CONCLUSION

Since databases are usually part of larger systems, the predictability in their behaviour is an important feature. Changing the physical design is a heavyweight operation, thus some level of guarantee is certainly needed. Promising improvement that eventually will not be obtained may cause users frustration and ultimately discourage them from using the tools. Therefore, designers have to deliver solutions with a high level of certainty, being hence trustworthy to implement.

In this paper, we evaluate the predictability of physical designers of three commercial database systems. We explore whether what we receive as the output of the tuning process corresponds to the improvement we gain after applying the proposed configuration. Our results show that the systems are not only inaccurate in their estimates, but are also inconsistent and hence even more unpredictable across different experiments raising questions regarding their trustworthiness. Furthermore, even with these rather simplistic workloads, we see a detrimental effect of the query optimizer's cardinality errors on the quality of proposed designs. In addition, update-intensive workloads seem to be a stepping-stone of physical designers that could not model accurately the trade-off between the improvement that proposed structures bring and the cost of maintaining them. Hence, we argue that in the light of the predictability of those tools there is still significant room for improvement, and a plethora of already published academic work gives some directions worth following and integrating in the physical designers.

7. ACKNOWLEDGMENTS

We would like to thank the DIAS laboratory members and the anonymous reviewers for their valuable feedback on the previous version of this paper, which substantially improved the current presentation.

8. REFERENCES

[1] A. Aboulnaga, P. Haas, M. Kandil, S. Lightstone, G. Lohman, V. Markl, I. Popivanov, and V. Raman. Automated statistics collection in DB2 UDB. In *VLDB*, 2004.

[2] S. Agrawal, S. Chaudhuri, L. Kollar, A. Marathe, V. Narasayya, and M. Syamala. Database Tuning Advisor for Microsoft SQL Server 2005. In *VLDB*, 2004.

[3] B. Babcock and S. Chaudhuri. Towards a Robust Query Optimizer: A Principled and Practical Approach. In *SIGMOD*, 2005.

[4] N. Bruno. A critical look at the TAB benchmark for physical design tools. *SIGMOD Rec.*, 36:7–12, 2007.

[5] S. Chaudhuri and V. Narasayya. AutoAdmin "what-if" index analysis utility. In *SIGMOD*, 1998.

[6] S. Chaudhuri and V. Narasayya. Self-tuning database systems: a decade of progress. In *VLDB*, 2007.

[7] C. M. Chen and N. Roussopoulos. Adaptive selectivity estimation using query feedback. In *SIGMOD*, 1994.

[8] S. Christodoulakis. Implications of certain assumptions in database performance evaluation. *ACM Trans. Database Syst.*, 9:163–186, 1984.

[9] M. P. Consens, D. Barbosa, A. Teisanu, and L. Mignet. Goals and benchmarks for autonomic configuration recommenders. In *SIGMOD*, 2005.

[10] B. Dageville, D. Das, K. Dias, K. Yagoub, M. Zait, and M. Ziauddin. Automatic SQL tuning in Oracle 10g. In *VLDB*, 2004.

[11] D. Dash, N. Polyzotis, and A. Ailamaki. CoPhy: a scalable, portable, and interactive index advisor for large workloads. *PVLDB*, 4:362–372, 2011.

[12] S. Finkelstein, M. Schkolnick, and P. Tiberio. Physical database design for relational databases. *ACM Trans. Database Syst.*, 13:91–128, 1988.

[13] K. E. Gebaly and A. Aboulnaga. Robustness in automatic physical database design. In *EDBT*, 2008.

[14] P. J. Haas and A. N. Swami. Sequential sampling procedures for query size estimation. In *SIGMOD*, 1992.

[15] Y. E. Ioannidis. The History of Histograms (abridged). In *VLDB*, 2003.

[16] K. Schnaitter and N. Polyzotis. A Benchmark for Online Index Selection. In *ICDE*, 2009.

[17] TPC. Tpc-h benchmark. `http://www.tpc.org/tpch/`.

[18] C. H. Wu and et al. The Protein Information Resource: an integrated public resource of functional annotation of proteins. *Nucleic Acids Research*, 2002.

[19] D. C. Zilio, J. Rao, S. Lightstone, G. M. Lohman, A. J. Storm, C. Garcia-Arellano, and S. Fadden. DB2 Design Advisor: Integrated Automatic Physical Database Design. In *VLDB*, 2004.