Looking Deeply into the Magic Mirror: An Interactive Analysis of Database Index Selection Approaches

Stefan Halfpap
BIFOLD, TU Berlin
Berlin, Germany
halfpap@tu-berlin.de

Jan Kossmann
Snowflake Inc.
Berlin, Germany
jan.kossmann@snowflake.com

Rainer Schlosser
Hasso Plattner Institute
Potsdam, Germany
rainer.schlosser@hpi.de

Volker Markl
BIFOLD, TU Berlin, DFKI
Berlin, Germany
volker.markl@tu-berlin.de

ABSTRACT

Indexes are important data structures for database tuning. However, finding the best indexes for a given workload is challenging. In this demonstration, we present our extensible open-source index selection evaluation platform and the corresponding interactive result analysis tool. The platform provides an automatic setup of the database, workload, and cost evaluation, which is otherwise often tedious work when evaluating index selection approaches. Users can also connect the platform to their own existing database and evaluate indexes for custom workloads. Our platform comprises multiple state-of-the-art index selection approaches, which can be used as baselines for new index selection proposals. Further, we present an application for thoroughly analyzing the selected database indexes. One can observe which indexes are used for which queries and their effect on processing costs. Also, it is possible to adapt the resulting index selections (i.e., add, remove, or change an index) and observe the impact. In this process, the application helps to understand the effects of indexes, improve index selections, and craft new index selection approaches.

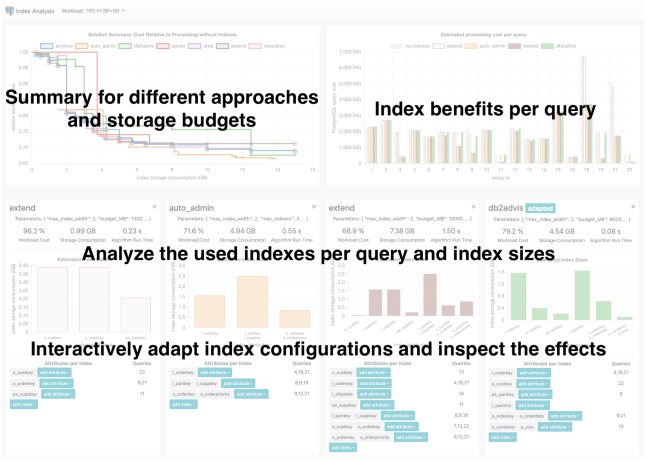

Figure 1: Demo summary: Evaluating database index selection approaches and analyzing the chosen indexes.

PVLDB Reference Format:
Stefan Halfpap, Jan Kossmann, Rainer Schlosser, and Volker Markl. Looking Deeply into the Magic Mirror: An Interactive Analysis of Database Index Selection Approaches. PVLDB, 17(12): 4301 - 4304, 2024.
doi:10.14778/3685800.3685860

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/klauck/index_analysis.

1 INTRODUCTION

Indexes are commonly used to speed up database workloads. However, indexes consume storage and cause (usually) maintenance overhead when the underlying data changes due to inserts, updates, or deletes. Thus, selecting the best database indexes, i.e., the indexes that minimize the workload cost under given constraints, is essential for database systems running on-premise or in the cloud.

Challenges of the index selection problem. Complex workloads on large schemas make choosing suitable indexes challenging for various, well-known reasons [8]: First, the set of relevant index candidates is usually huge because indexes can be defined on arbitrary subsets of table attributes. Further, indexes interact with each other, i.e., the benefit of an index can depend on the existence of other indexes. Thus, we cannot determine the benefit of an index independently.

Moreover, efficiently determining workload costs under various index subsets (given the previously described interaction) is challenging. The naive solution of physically creating indexes and executing queries is usually impractical because it is too time-consuming and costly. Instead, we usually estimate workload cost under a given index set. Naturally, such estimations come with potential inaccuracies [1]. When conducting cost estimations, state-of-the-art approaches include the database optimizer because the theoretically best index is worth nothing if it is not used during query processing. However, because optimizer invocations are relatively costly, this approach limits the number of tractable cost estimations and, thus, index candidates, which can be considered during the selection process.

Besides, dynamic environments (e.g., changing data and workloads) and selection robustness (i.e., the workload cost under the given index configuration does not degrade much if the workload changes) pose further challenges for the index selection problem.

Index Selection Approaches. Diverse index selection approaches, which automatically choose indexes, have been proposed since the 1970s. These various approaches differ in underlying solution approaches and complexity. Conceptually, *imperative* approaches

Proceedings of the VLDB Endowment, Vol. 17, No. 12 ISSN 2150-8097.
doi:10.14778/3685800.3685860

either start with an empty index set and extend their selection successively or start with a large candidate set that is reduced continuously. But there are also *declarative* approaches, for example, using integer linear programming, which formalizes the problem with an optimization function and a set of (in)equalities as constraints. There are also *machine learning-powered* [12] and specifically reinforcement learning (RL)-based approaches, which define the index selection problem as Markov decision problem (MDP) via reward functions, states (chosen indexes), and admissible actions.

Demonstration Proposal. In this demonstration, we present our extensible open-source index selection evaluation platform that can evaluate various index selection approaches for different workloads. The platform can be connected to an existing database and used on custom workloads, but it also provides an automatic setup of the database, pre-defined workloads, and cost evaluation, which is otherwise often tedious work when running index selection approaches. Further, it comprises multiple index selection approaches, which can be used as baselines for new index selection proposals. The platform currently supports running eight different selection algorithms, namely the Drop Heuristic [14], AutoAdmin's index selection [4], DB2 Advisor [13], Relaxation [2], CoPhy [5], Dexter [7], Extend [11], and DTA Anytime [3]. All algorithms can be run with different parameter settings using the standardized TPC-H and TPC-DS workloads, the Join Order Benchmark, and user-defined workloads. In this demonstration, we let users (1) conduct a fully automated setup and evaluation of index selection approaches and (2) evaluate indexes for custom workloads on an existing database.

To analyze the results, we built an interactive web application on top of it (see Figure 1). The application summarizes the benchmark results for index selection approaches depending on the storage consumption of the deployed indexes as a starting point for a deeper analysis. For every approach and the indexes' overall storage consumption, the application enables analyzing the selected indexes and the index benefit per query. Finally, we allow the user to adapt the index selection and investigate the impact on processing costs and index sizes.

This application enables comparing and evaluating diverse index selection approaches for various settings. By doing so, we can derive the strengths and weaknesses of individual approaches. Further, we can understand how specific indexes improve the estimated query costs. This knowledge helps in understanding the usage of indexes in general and developing new approaches [6, 9] that combine the most valuable concepts and also consider robustness, e.g., concerning workload uncertainty [10, 15].

Contributions. Our demonstration can be summarized as follows:

- We evaluate index selection approaches for pre-defined and custom workloads while the setup is fully automated.
- We analyze high-level results (i.e., which approach performs best for which workload and parameters) and details (i.e., the benefit and applied indexes per query).
- We observe the effects when manually modifying the current index selection.
- Our platform and application are open source and can be extended for further selection approaches and functionality.

2 INDEX SELECTION EVALUATION

Evaluating new or comparing existing index selection approaches from scratch is tedious because it does not only require implementing the selection algorithms. It also includes setting up the database and loading the data. Further, cost evaluation capabilities must be integrated into the selection algorithms so that they can, for example, use the optimizer's cost estimates or actual processing times. Moreover, it is desirable to automate the running of approaches for various parameter settings (e.g., storage budget or number of attributes per index) and different workloads. Finally, benchmark results need to be measured, collected, and evaluated. In the end, the evaluation work often takes more time than implementing the actual index selection approach.

To facilitate the development and evaluation of index selection approaches, we developed an evaluation platform [8]. The platform supports the automatic setup of databases and evaluation of approaches for different workloads, including the standardized benchmarks TPC-H, TPC-DS, and the Join Order Benchmark. Users can also connect the platform to an existing database and evaluate indexes for custom workloads. Further, the platform includes implementations of eight state-of-the-art approaches for comparison. The platform is open source[1] and can be extended for additional workloads, approaches, and database systems (see GitHub page).

To run an evaluation, we only have to specify a single (JSON) configuration file. Listing 1 shows a valid configuration file for conducting an evaluation with our platform.

```
{   "database_system": "postgres",
    "benchmark_name": "tpch",
    "scale_factor": 10,
    "queries": [1, 4, 5, 7, 9, 12, 13, 14, 17],
    "algorithms": [
        {   "name": "auto_admin",
            "parameters": {
                "max_indexes": [3, 4, 5],
                "max_index_width": 2
            },
        },
        {   "name": "db2advis",
            "parameters": {
                "max_index_width": 2,
                "budget_MB": [2500, 5000],
            },
        },
    ]
}
```

Listing 1: Example configuration for running an index selection evaluation in PostgreSQL.

In this case, we would use PostgreSQL with the TPC-H dataset (scale factor 10) and a workload consisting of nine TPC-H queries. Two different parametrized approaches, namely AutoAdmin and the DB2 Advisor, would create index configurations. Overall, there would be six index selection runs: Three for AutoAdmin, which select 3, 4, and 5 indexes (parameter `max_indexes`), each with a maximum of two attributes (`max_index_width = 2`); two for DB2Advis, which select indexes with an overall maximum storage consumption of 2.5 and 5 GB (parameter `budget_MB`) and the same index width as above.

[1] http://git.io/index_selection_evaluation

Figure 2: Result analysis. Top left: summary graph. Top right: cost per query. Bottom: selected index configuration information.

The evaluation is started with a single command, e.g., "`python -m selection config.json`". The platform first sets up the database, e.g., generating and loading TPC-H data, if not existing yet. It then runs the configured approaches for the specified workload. The result data of an evaluation run is stored in CSV files per approach and is the basis for the analysis. The CSV files contain one line per index selection run and consist of, e.g., the determined index configuration, the runtime for determining it, and the cost of every query given the configuration.

3 RESULT ANALYSIS

Figure 2 shows a screenshot of our result analysis application[2], in this case for comparing index selection approaches for a TPC-H workload. We can choose between analyzing different workload results via the navigation bar at the top.

Summary Graph. The graph in the top left part is visible right from the application start and displays a result summary for the selected workload. It shows the workload cost for every (potentially parametrized) approach per storage consumption of the solutions' index selection. In the example, it visualizes the results of 7 approaches. An individual data point corresponds to a single index selection run with a fixed approach and parameters, e.g., running the AutoAdmin approach with `max_indexes=4` and `max_index_width=2`.

With this graph, we can analyze approaches on a high level. For example, we can see how much workloads benefit in general from indexes and which approach finds the best indexes for which storage budget. In our example, AutoAdmin finds the best indexes for the selected workload for storage budgets of 5 GB and above.

However, the summary graph hides detailed information about the index selection (e.g., which indexes have been selected) that could be used for a more detailed understanding. Individual data points (corresponding to parametrized approaches) can be selected via click for a deeper analysis. Selected data points are then included in the graph that visualizes the cost per query, and their details are added to the area below; both are described below in more detail.

Cost per Query. The graph in the top right shows the cost per query without indexes (as baseline) and for every selected data point. In the example, four data points are selected. Data points for the same base approach have the same border color. The color intensity corresponds to the storage consumption.

Using this graph, we can inspect the most costly queries and how much the individual queries benefit from the current indexes. At

[2]https://github.com/klauck/index_analysis

the same time, the most expensive queries may be good candidates for additional tuning. In our example workload, query 19 and 21 have the highest cost but also strongly benefit from indexes.

Selection Details. The area below the summary and cost-per-query graph shows details for user-selected data points. First, it states the approach and the applied parameters as well as key metrics, i.e., (1) the relative workload cost of the selection based on the workload cost without indexes, (2) the storage consumption of the current index selection, and (3) the calculation time of the selection run.

Below, we can inspect the selected indexes: first, a chart visualizes the storage consumption per index; second, we can inspect the attributes per index and which queries benefit from the index.

In our example workload, the AutoAdmin approach currently has three indexes. The single-attribute index on the `l_orderkey` attribute has an estimated storage consumption of about 1.5 GB and is used in query 4, 18, and 21.

Interactivity. This view also allows to adapt the current index selection interactively. Thereby, we can tune index configurations and directly observe the impact on storage and workload cost. We can adapt existing indexes, i.e., remove individual attributes by clicking on them or appending an attribute (to the end) of the index using the drop-down menu next to the index. In addition, we can remove an existing index by removing all of its attributes. Further, we can also add an arbitrary index to the selection. In this process, we first add a single-attribute index using the drop-down menu at the bottom of the index overview. Afterward, we can add additional attributes to the single-attribute index to obtain the desired multi-attribute index.

Each individual adaption triggers a re-evaluation of the new index selection, and the results are visualized in the storage consumption, queries benefitting, and cost per query. Because index adaptions only affect subsets of the queries and the platform caches cost estimates, the number of cost estimations in the database system for this re-evaluation is usually low.

The detailed information of individual approaches is beneficial when comparing multiple approaches and in interplay with the cost per query graph. Having multiple selected parametrized approaches, we can compare their selected indexes. Using the cost per query, we can detect possibilities to improve the current selection. Further, we can interactively try various adaptions and see whether they have the desired effects. While specific approaches often try out slight (single-attribute or single-index) adaption during their selection process, it is often too costly to try every adaption of adding multiple multi-attribute indexes, which is possible with our application.

4 DEMO EXPERIENCE

We present how to configure an index selection evaluation and conduct an exemplary run. Users can then inspect the results and adapt the index selections on their own with our guidance. For example, together with the attendee, we want to improve the current index configuration and find another beneficial index by interacting with our application. In the demo, users can also analyze and adapt evaluation results for TPC-H, TPC-DS, and the Join Order Benchmark. Providing pre-computed results is useful because the index selection for larger workloads and specific approach settings may take over an hour [9]. Further, attendees can define custom workloads and try to find the best indexes for them. Finally, we show how to implement new selection approaches and explain how the platform can be extended for further database systems.

5 CONCLUSIONS

We present our index selection evaluation platform and the interactive result analysis tool. Using these applications, we can evaluate indexes for custom workloads on an existing database. Further, our platform enables the automatic evaluation of index selection approaches for databases. The automatic evaluation facilitates improving existing index selection approaches and crafting new ones because it provides the database, workload, cost evaluation, and baseline implementation, which otherwise often makes up a dominant share of the development time. On top of the evaluation platform, we built an interactive analysis tool to compare results on a high level, deeply inspect the selected indexes, and adapt the current selection. Besides gaining insights about specific index selection approaches, our tool also helps in understanding the usage of indexes in general.

REFERENCES

[1] Renata Borovica, Ioannis Alagiannis, and Anastasia Ailamaki. 2012. Automated physical designers: what you see is (not) what you get. In *Proceedings of the International Workshop on Testing Database Systems (DBTEST)*. 9:1–9:6.

[2] Nicolas Bruno and Surajit Chaudhuri. 2005. Automatic Physical Database Tuning: A Relaxation-based Approach. In *Proceedings of the International Conference on Management of Data (SIGMOD)*. 227–238.

[3] Surajit Chaudhuri and Vivek Narasayya. 2020. Anytime Algorithm of Database Tuning Advisor for Microsoft SQL Server. (2020). https://www.microsoft.com/en-us/research/publication/anytime-algorithm-of-database-tuning-advisor-for-microsoft-sql-server, accessed: July 17, 2024.

[4] Surajit Chaudhuri and Vivek R. Narasayya. 1997. An Efficient Cost-Driven Index Selection Tool for Microsoft SQL Server. In *Proceedings of the International Conference on Very Large Databases (VLDB)*. 146–155.

[5] Debabrata Dash, Neoklis Polyzotis, and Anastasia Ailamaki. 2011. CoPhy: A Scalable, Portable, and Interactive Index Advisor for Large Workloads. *Proceedings of the VLDB Endowment* 4, 6 (2011), 362–372.

[6] Stefan Halfpap. 2023. Hybrid Index Selection Using Integer Linear Programming Based on Cached Cost Estimates of Heuristic Approaches. In *Proceedings of the 1st Workshop on Simplicity in Management of Data, (SiMoD) @ SIGMOD*. 5:1–5:4.

[7] Andrew Kane. 2017. Introducing Dexter, the Automatic Indexer for Postgres. https://medium.com/@ankane/introducing-dexter-the-automatic-indexer-for-postgres-5f8fa8b28f27, accessed: July 17, 2024.

[8] Jan Kossmann, Stefan Halfpap, Marcel Jankrift, and Rainer Schlosser. 2020. Magic mirror in my hand, which is the best in the land? An Experimental Evaluation of Index Selection Algorithms. *Proceedings of the VLDB Endowment* 13, 11 (2020), 2382–2395.

[9] Jan Kossmann, Alexander Kastius, and Rainer Schlosser. 2022. SWIRL: Selection of Workload-aware Indexes using Reinforcement Learning. In *Proceedings of the International Conference on Extending Database Technology (EDBT)*. 2:155–2:168.

[10] Rainer Schlosser and Stefan Halfpap. 2020. A Decomposition Approach for Risk-Averse Index Selection. In *Proceedings of the International Conference on Scientific and Statistical Database Management (SSDBM)*. 16:1–16:4.

[11] Rainer Schlosser, Jan Kossmann, and Martin Boissier. 2019. Efficient Scalable Multi-Attribute Index Selection Using Recursive Strategies. In *Proceedings of the International Conference on Data Engineering (ICDE)*. 1238–1249.

[12] Tarique Siddiqui and Wentao Wu. 2024. ML-Powered Index Tuning: An Overview of Recent Progress and Open Challenges. *SIGMOD Rec.* 52, 4 (2024), 19–30.

[13] Gary Valentin, Michael Zuliani, Daniel C. Zilio, Guy M. Lohman, and Alan Skelley. 2000. DB2 Advisor: An Optimizer Smart Enough to Recommend Its Own Indexes. In *Proceedings of the International Conference on Data Engineering (ICDE)*. 101–110.

[14] Kyu-Young Whang. 1985. Index Selection in Relational Databases. In *Proceedings of the International Conference on Foundations of Data Organization (FoDO)*. 487–500.

[15] Wei Zhou, Chen Lin, Xuanhe Zhou, Guoliang Li, and Tianqing Wang. 2023. Demonstration of ViTA: Visualizing, Testing and Analyzing Index Advisors. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*. 5133–5137.