Towards an End-to-End Data Quality Optimizer

Valerie Restat
Chair of Databases and Information Systems
University of Hagen
Hagen, Germany
0000-0002-5960-5886

Meike Klettke
Chair of Data Engineering
University of Regensburg
Regensburg, Germany
0000-0003-0551-8389

Uta Störl
Chair of Databases and Information Systems
University of Hagen
Hagen, Germany
0000-0003-2771-142X

***Abstract*—To provide good results and decisions in data-driven systems, data quality must be ensured as a primary consideration. An important aspect of this is data cleaning. Although many different algorithms and tools already exist for data cleaning, an end-to-end data quality solution is still needed. In this paper, we present our vision of a well-founded end-to-end data quality optimizer. In contrast to many studies that consider data cleaning in the context of machine learning, our approach focuses on areas of application where the downstream analysis is unknown. Our proposed adaptive and easily extendable framework operates similarly to proven methods of database query optimization. Analogously, it consists of the following parts: Rule-based optimization, where the appropriate data cleaning algorithms are selected based on use case constraints, optimizer hints in the form of best practices, and cost-based optimization, where the cost is measured in terms of data quality. Accordingly, the result is a data cleaning pipeline that provides the best possible data quality. The choice of different optimization goals enables further flexibility, e.g. for environments with limited resources.**

***Index Terms*—data quality, data cleaning, optimization**

I. Introduction

Since data are important resources in all organizations, it is of crucial importance to ensure their quality. For this reason, data cleaning is a central aspect of working with data. At the same time, it is a complex and very often time-consuming process that has been intensively studied by the data management community. Since many different types of error can occur in data, it usually requires many different algorithms and tools. Abedjan et al. [1] have investigated the best strategy for a holistic approach to error detection. They conclude that "there is no single dominant data cleaning tool for all data sets and blindly combining the tools will likely decrease precision" [1]. Therefore, they emphasize the importance for an *end-to-end data quality solution* [1].

In many works (e.g. [2]–[4]), data cleaning is generally considered in the context of machine learning. This is useful if the machine learning model is known and the entire process can be accessed. Nevertheless, our own experience with applications in the banking sector shows that data preprocessing often takes place separately from a subsequent analysis. Various teams work on the cleaning of the data and the subsequent analysis. People who preprocess data have no knowledge of its subsequent use. Vice versa, people who analyze the data and create machine learning models cannot influence data preprocessing. This requires a data preparation solution independent of the downstream application.

The contribution of this paper is our vision of such a solution as an end-to-end data quality optimizer. For this, we propose an adaptable and extensible framework. It consists of a sequence of a rule-based and a cost-based optimization of data cleaning pipelines. In the context of rule-based optimization, the appropriate data cleaning methods are selected depending on the use case. In addition, optional best practices comparable to optimizer hints can be included. Depending on these optimizations, the pipeline that results in the best data quality is selected in the cost-based optimization. Different optimization goals can be set for this, e.g. for environments with limited resources or applications in which the data must be processed near real time. By focusing on data quality, our approach allows applicability for use cases where the subsequent analysis is unknown. Measuring data quality in terms of metrics is an open research question without any standardization yet in existence. We introduce our proposed solution in Section III-C.

The remainder of the paper is structured as follows: First, Section II provides an overview of the state of the art. Our vision of an end-to-end data quality optimizer, including rule-based optimization, best practices, and cost-based optimization is presented in Section III. Section IV concludes with a summary and an outlook.

II. State of the art

To evaluate data quality, suitable metrics are needed. Existing metrics (such as in [5] and [6]) address only a few aspects of data quality. Thus, a systematic framework that considers many different dimensions of data quality is required. In previous work, we proposed CheDDaR (Checking Data – Data Quality Review) [7], a framework that permits a flexible evaluation of data quality and data preparation results. We plan to implement this for cost-based optimization.

Similarly, appropriate transformation processes need to be developed to improve data quality. The topic is of great interest in research and practice. For this reason, a range of different data cleaning methods exist. A description of the end-to-end data cleaning process can be found in [8]. Numerous methods for error detection and repair are described there. Analogous to the multitude of methods, there are also many different data cleaning tools. For example, the combination of Raha and Baran represents an end-to-end data cleaning pipeline [9]. Nonetheless, its error classification only covers a part of all

possible errors, and it is only suitable for structured data and not as flexible as our approach. Many other tools also exist, but they are either use-case-specific or only deal with one or a few error types. An overview can be found in [10]. However, as described, Abedjan et al. [1] have shown that there is no single dominant data cleaning tool yet. Accordingly, users are challenged to decide on the most suitable method or tool as well as composition for a particular problem [11]. This emphasizes the need for an end-to-end data quality solution.

This need is also discussed in [2] and a vision of a holistic data cleaning framework is presented. While the authors also use optimizers to combine different signals (data cleaning information) and produce a pipeline to detect and repair errors, the extent to which different signals and optimizers can be integrated into a holistic data cleaning approach remains an open research question. Furthermore, their focus lies on machine learning and human interaction. This is also the focus of most other work in this area [2]–[4], [12]. These approaches use a downstream model or application to optimize cleaning. In contrast, our approach is suitable for applications in which the subsequent analysis is unknown. Hence, we use data quality for optimization. There is no standard definition of data quality yet, as the authors of [2] have also pointed out. Moreover, in distinction to other work, we provide the ability to define further optimization goals and attempt to reduce human interaction where possible.

III. End-to-end data quality optimizer

In this section we present our vision of an end-to-end data quality optimizer. The process is comparable to well-established methodologies from the database management field of query optimization [13], [14]. The aim is to translate a user-submitted query into an effective plan. In our vision of an end-to-end data quality optimizer, the aim is the translation of a concrete user request (data set, available resources, time constraints, etc.) into a data cleaning pipeline that results in the best possible data quality.

To address the requirements of different use cases, various optimization goals are possible. In our smart city projects, our project partners often reported that cities would only have limited hardware with few resources. In other use cases, processing may need to be as "green" as possible or as close to real time as possible. It is therefore helpful to be able to select different optimization goals.

a) Notation: Before explaining the process, we introduce the notation used. In the following, a distinction is made between *algorithms* and *algorithm classes*. An algorithm class is represented by separate slices in Figure 1. For each type of error, there is an algorithm class. An example of an algorithm class would be missing value imputation. An algorithm is a concrete method for repairing the corresponding error type. An example would be missing value imputation using mean imputation. In Figure 1, algorithms are represented by gears.

b) End-to-end data quality optimizer: Overview: Figure 2 shows an overview of the entire process of the end-to-end data quality optimizer. The first step is the *rule-based optimization* [13]. Its goal is to narrow down the search space with all possible algorithms. Part 1 filters algorithms not suitable for the use case (by constraints such as data characteristics or the available resources) and Part 2 filters algorithms which do not affect the order of the pipeline. In a second step, similarly to optimizer hints [15], *best practices* can be included. Subsequently, the next step is the *cost-based optimization* [14]. In this case, costs are measured in terms of data quality. Hence, the best possible pipeline in terms of data quality is selected in the remaining search space.

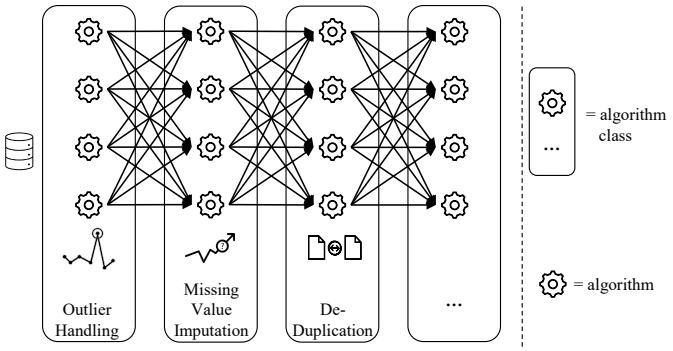

Fig. 1. Overview of Data Cleaning Pipelines with different Algorithm Classes and Algorithms

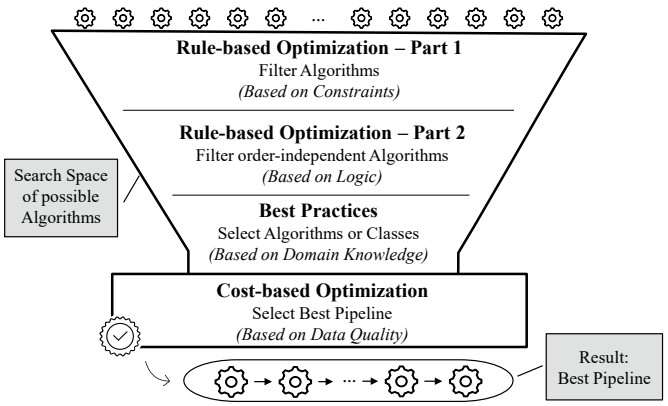

Fig. 2. Overview of our Vision of an End-to-End Data Quality Optimizer

The components are built step-by-step and modularly expandable. Each of these steps will be described in more detail below.

A. Rule-based optimization

The selection of the most suitable data cleaning pipeline begins with rule-based optimization. The search space of possible algorithms is reduced by constraints and rules.

1) Filtering algorithms by constraints: In the first step, algorithms that do not suit the use case constraints, e.g. the data, are filtered out. Coming back to the example of missing value imputation: If a column consists of string values, a mean imputation is not suitably applicable. Another example would be the influence of missing rates. In [16] it was shown that the K-nearest Neighbors Method leads to significantly worse

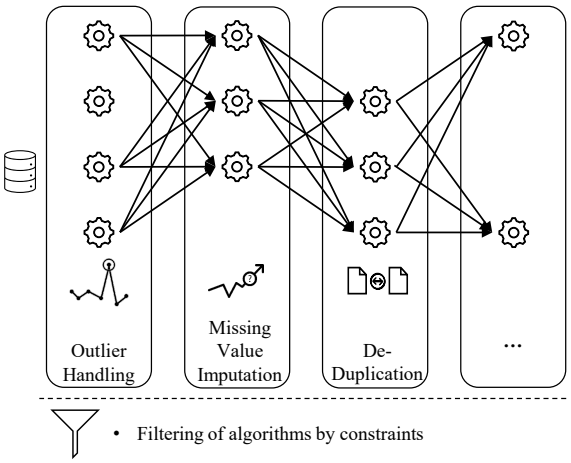

Fig. 3. Rule-based Optimization – Part 1: Filtering of Algorithms by Constraints

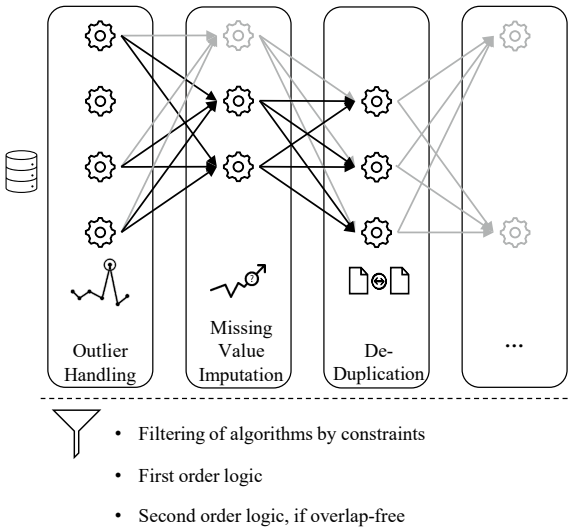

Fig. 4. Rule-based Optimization – Part 2: Filtering of order-independent Algorithms

results compared to other algorithms once the missing rate increases. Based on this, it can be concluded that this method should not be used if the missing rate is high. So, these algorithms can be removed from the search space. Figure 3 shows the result of this step. Individual algorithms (= gears) have been removed.

Research challenges: To perform this step, future research needs to investigate which algorithms – depending on the input data and the optimization goals – are suitable and which are not. As a first step, our current research focuses on missing value imputation. Our initial analyses show that many different aspects have an impact, such as size, dimensionality, data type, variability, distribution, associations, missing mechanism, missing rate, and missing pattern. Other algorithm classes like outlier handling will follow.

2) Filtering order-independent algorithms: The next step is to filter out those algorithms or classes which are not relevant for the pipeline order. These are still part of the pipeline. They can be executed at any time, since they do not play any role in the order of the pipeline.

a) First-order logic: Some data engineering algorithms generate the same result regardless of the order of the algorithms in the pipeline. For example, if we have one algorithm that contains a projection (e.g., to shape the data set) and another one that replaces the values of an attribute with other values (e.g., to convert them to a different physical unit). In that case, the order of execution of both algorithms is arbitrary. We can generalize this to data engineering algorithms that can be represented in first-order logic. Unfortunately, only a few real-world algorithms fall into this class. In addition to the projection mentioned before, selection (e.g., a sample selection in data engineering) and type cast (e.g., converting integer to double) fall into this class as well. Other data cleaning algorithms are more complex and belong to the next class.

b) Second-order logic, if overlap-free: The next class is second-order logic. The execution of algorithms that belong to this class is not arbitrary. For example, if you use a method to impute missing values and the method is based on clustering the entire tuples, it will produce different results if an algorithm has modified the values of a column before it. Hence, it means that the order is important here. But even in this class of algorithms, you can find cases where you can swap the order of the algorithms without affecting the result. To identify these algorithms whose execution can occur at any point in the pipeline, it is necessary to check which part of the data is accessed by the corresponding algorithm. For example, if an algorithm is cleaning a column that is not used by any other algorithm, this cleaning is *overlap-free* and thus not relevant for the order of the pipeline.

Figure 4 shows the result of this step. Individual algorithms and classes are greyed out. As noted, these must still be performed, although they do not play a role in the choice of the execution order.

Research challenges: To perform this step, future research needs to analyze which algorithms belong to first-order logic and which to second-order logic. Even though it is likely that only very few algorithms belong to first-order logic and most algorithms are not free of overlap, it is still worth investigating these cases to possibly minimize the search space. Thus, it needs to be analyzed which algorithms are free of overlaps.

For error detection, we have already classified different error levels [17]. This includes the following levels:

- An Attribute Value of a Single Tuple
- The Values of a Single Attribute
- The Attribute Values of a Single Tuple
- The Attribute Values of Several Tuples

In our current research, we are investigating the same for the repair of errors. We further investigate which special features arise when working with semi-structured data. This also includes the analysis of error types of structured data that may also occur in semi-structured data and which new

error types emanate. When working with semi-structured data, greater schema flexibility is allowed. As a result – in addition to errors in the data itself – structural errors can also occur. An example of this would be missing attributes.

B. Best Practice

After the rule-based optimization, only the applicable algorithms relevant for the order of the pipeline are left. To further restrict the search space, additional best practices can be applied. In the analogy to query optimization, this step corresponds with optimizer hints. These are used to guide the optimizer into the right direction [15]. Such hints can be set separately for each individual query [15]. Similarly, in our vision, depending on the use case domain experts can contribute their knowledge, for example, if they know that a certain algorithm will produce the best results for the data, or that certain algorithms already proved to work together effectively.

This is comparable to the *explore* and *exploit* principle known in the AutoML area [18]. Here, it is possible to balance exploring (evaluating as many hyperparameters as possible) and exploiting (allocating more resources to promising hyperparameters). With our proposed best practices, it is possible to balance between focusing on promising data cleaning methods (exploit) and searching the entire search space (explore). This is enhanced by different optimization goals.

In contrast to rule-based optimization, in this step algorithms can be selected specifically for inclusion in the final pipeline. Additionally, an explicit order of algorithms for the pipeline can be specified in this step. To continue with the example of missing value imputation: Hasan et al. [19] state that it is recommended to standardize the data first when using distance-based algorithms (such as missing value imputation with K-nearest Neighbors). They also state that when using regression-type algorithms, outlier rejection must first take place. Median-based algorithms, on the other hand, are robust to outliers, since they calculate the missing values from the most frequent values, which must not be outliers [19].

Research challenges: In addition to recommendations for the explicit use of algorithms – depending on the use case – research related to the ordering of algorithms is particularly relevant in this area. The size of the search space is largely influenced by the number of permutations. When more best practices exist for ordering, it will lead to fewer permutations and thus minimize the search space. Doing so will notably improve the efficiency of the cost-based optimization. This step is explained in greater detail below.

C. Cost-based optimization

After the search space has been narrowed down further according to best practices, the order of all algorithms is determined. As with database systems [20], different levels of optimization can be set. Depending on the search space and the best practices given, the optimization may vary regarding compilation time and the quality of the pipeline generated. For example, if not enough time and resources are available, the search space can be limited by best practices. If sufficient time and resources are provided, algorithms not specified by best practices can still be considered for cost-based optimization.

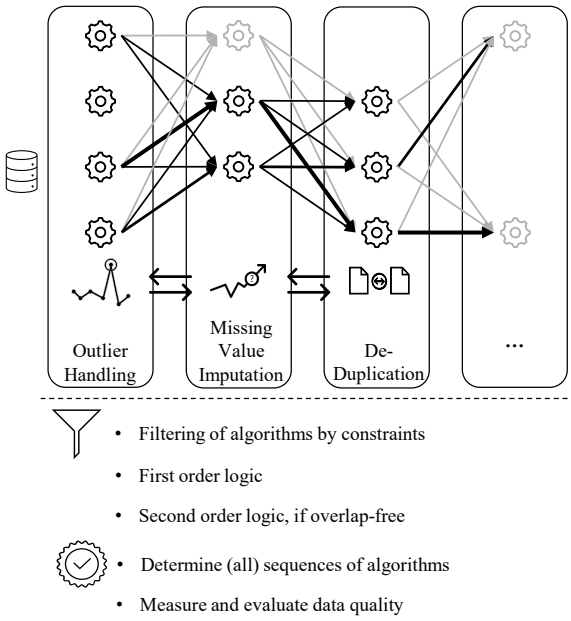

Fig. 5. Cost-based Optimization: Determining Sequences and measuring and evaluating Data Quality

Depending on the remaining size of the search space, either all permutations are formed or, in case too many combinations remain, a Monte Carlo Simulation [21], [22] is conceivable. The result is a multitude of different pipelines. It should be noted that these pipelines also include those algorithms that are irrelevant to the execution order. For each possible pipeline, the cost – that is, the data quality – are measured and evaluated. This step is demonstrated graphically in Figure 5. Finally, the pipeline that achieves the highest data quality is selected.

Research challenges: For this step, a framework of metrics is needed to assess data quality. We have already developed a framework called CheDDaR (Checking Data – Data Quality Review) [7] to measure data quality and evaluate data cleaning algorithms. This will be applied for the cost-based optimization. The advantage of CheDDaR is that it takes many different aspects of data quality into account. The metrics correspond to the extensive error classification that we have performed in [17]. The fewer errors in the data, the better the data quality. Furthermore, CheDDaR can be used flexibly, depending on the extent to which ground truth is available or domain experts are available for verification. It is thus well suited for evaluating the different pipelines and allows to select the pipeline that achieves the highest data quality. The applications of CheDDaR go beyond cost-based optimization, but we are currently working on an implementation specifically for this purpose.

D. Further Improvements

In addition to the described procedure, further improvements are conceivable:

a) Human in the loop: Data cleaning essentially focuses on humans [2]. This is taken into account in our data quality framework CheDDaR [7], which distinguishes between different impacts of domain knowledge. It is nonetheless also conceivable to involve the user even more. Although the envisioned optimizer aims to minimize human involvement, it could be beneficial to offer an optional possibility for manual adaption. One approach would be an interactive dashboard, for example in the form of a WebUI. Here, users, primarily domain experts with insider knowledge of the data, could intervene and manually adjust the pipeline. The proposed incorporation of best practices, where domain experts can provide input, already enables such an interactive human-in-the-loop approach.

Fairness also plays an important role. This should be considered as early as possible in the pipeline to avoid a potential bias. For this, it is conceivable that tools like *mlinspect* [23] are integrated into the pipeline. Mlinspect is a library to assist in bias detection, as this cannot be fully automated [23]. Looking at the beginning of the pipeline, CheDDaR [7] could also be helpful during data loading. Serious errors caused by the generation of the data could thus be detected from the start and new data requested accordingly.

b) Robustness of pipelines: Another challenge for data engineering pipelines is the constant change of data and their schema. Algorithms vary in their robustness to such changes. For example, a missing value imputation where a location is imputed depending on the zip code may no longer work if the attribute zip code is renamed by a structural change. In contrast, deduplication is robust to this change. In our current research, we purposefully perform a comprehensive analysis of the robustness of data engineering pipelines depending on different influencing factors. We investigate structural as well as semantic changes. If we know which algorithms are robust to what kind of change, we can determine the impact of such changes on the pipeline. In addition, we know which algorithms need to be adapted because they may no longer work after changes.

IV. Conclusion and Outlook

In this paper we have presented our vision of an end-to-end data quality optimizer along with the corresponding research challenges. It consists of a rule-based optimization, best practices, and a cost-based optimization. In rule-based optimization, possible algorithms are filtered out depending on the constraints of the use case. Comparable to optimizer hints, best practices can then be incorporated. Afterwards, in the cost-based optimization, the combination of suitable algorithms that leads to the highest possible data quality is selected.

In contrast to other work that focuses on cleaning for machine learning, our solution can be used in scenarios where further processing and analysis are unknown. Another distinctive factor is that different optimization goals can be pursued with this approach. This makes it flexible for various applications.

The approach is also suitable for working with streaming data. But, this entails further special features that are currently examined by us.

The modularity of the presented process provides the advantage that it can be enhanced later. For example, the optimizer can be extended to automatically select transformations or request additional data, as is envisioned in [11].

References

[1] Z. Abedjan *et al.*, "Detecting data errors: Where are we and what needs to be done?" *Proc. VLDB Endow.*, pp. 993–1004, 2016.

[2] F. Neutatz, B. Chen, Z. Abedjan, and E. Wu, "From cleaning before ML to cleaning for ML," *IEEE Data Eng. Bull.*, no. 1, pp. 24–41, 2021.

[3] M. Boehm *et al.*, "SystemDS: A declarative machine learning system for the end-to-end data science lifecycle," in *CIDR'20*, 2020.

[4] S. Krishnan and E. Wu, "AlphaClean: Automatic generation of data cleaning pipelines," *CoRR*, 2019. [Online]. Available: http://arxiv.org/abs/1904.11827

[5] B. Heinrich and D. Hristova, "A quantitative approach for modelling the influence of currency of information on decision-making under uncertainty," *J. Decis. Syst.*, vol. 25, no. 1, pp. 16–41, 2016.

[6] R. H. Blake and P. Mangiameli, "The effects and interactions of data quality and problem complexity on classification," *ACM J. Data Inf. Qual.*, vol. 2, no. 2, pp. 8:1–8:28, 2011.

[7] V. Restat, M. Klettke, and U. Störl, "FAIR is not enough - A metrics framework to ensure data quality through data preparation," in *BTW'23*, 2023, pp. 917–929.

[8] I. F. Ilyas and X. Chu, *Data Cleaning*, ser. ACM Books. ACM, 2019.

[9] M. Mahdavi and Z. Abedjan, "Semi-supervised data cleaning with raha and baran," in *CIDR*, 2021.

[10] M. Hameed and F. Naumann, "Data Preparation: A Survey of Commercial Tools," *SIGMOD Rec.*, pp. 18–29, 2020.

[11] M. Klettke and U. Störl, "Four Generations in Data Engineering for Data Science: The Past, Presence and Future of a Field of Science," *Datenbank-Spektrum*, pp. 59–66, 2021.

[12] P. Li *et al.*, "CleanML: A Study for Evaluating the Impact of Data Cleaning on ML Classification Tasks," in *ICDE'21*. IEEE, 2021, pp. 13–24.

[13] J. C. Freytag, "A rule-based view of query optimization," in *SIGMOD'87*. ACM, 1987, pp. 173–180.

[14] T. Neumann, "Query optimization (in relational databases)," in *Encyclopedia of Database Systems, Second Edition*. Springer, 2018. [Online]. Available: https://doi.org/10.1007/978-1-4614-8265-9_293

[15] L. Woltmann *et al.*, "Fastgres: Making learned query optimizer hinting effective," *Proc. VLDB Endow.*, no. 11, p. 3310–3322, 2023.

[16] C. Tsai and Y. Hu, "Empirical comparison of supervised learning techniques for missing value imputation," *Knowl. Inf. Syst.*, no. 4, pp. 1047–1075, 2022.

[17] V. Restat, G. Boerner, A. Conrad, and U. Störl, "GouDa - generation of universal data sets: improving analysis and evaluation of data preparation pipelines," in *DEEM'22*. ACM, 2022, pp. 2:1–2:6.

[18] J. Giovanelli and G. Pisano, "Towards human-centric automl via logic and argumentation," in *EDBT/ICDT Workshops*. CEUR-WS.org, 2022.

[19] K. Hasan *et al.*, "Missing value imputation affects the performance of machine learning: A review and analysis of the literature (2010–2021)," *Informatics in Medicine Unlocked*, p. 100799, 2021.

[20] I. F. Ilyas, J. Rao, G. M. Lohman, D. Gao, and E. T. Lin, "Estimating compilation time of a query optimizer," in *SIGMOD'03*. ACM, 2003, pp. 373–384.

[21] N. N. Dalvi and D. Suciu, "Efficient query evaluation on probabilistic databases," *VLDB J.*, no. 4, pp. 523–544, 2007.

[22] P. J. Haas, "Monte carlo methods for uncertain data," in *Encyclopedia of Database Systems, Second Edition*. Springer, 2018. [Online]. Available: https://doi.org/10.1007/978-1-4614-8265-9_80692

[23] S. Grafberger, S. Guha, J. Stoyanovich, and S. Schelter, "MLINSPECT: A data distribution debugger for machine learning pipelines," in *SIGMOD'21*. ACM, 2021, pp. 2736–2739.