Learned Cost Models for Query Optimization: From Batch to Streaming Systems

Roman Heinrich
TU Darmstadt & DFKI
roman.heinrich@dfki.de

Xiao Li†
IT University of Copenhagen
xliq@itu.dk

Manisha Luthra
TU Darmstadt & DFKI
manisha.luthra@dfki.de

Zoi Kaoudi
IT University of Copenhagen
zoka@itu.dk

ABSTRACT

Learned cost models (LCMs) have recently gained traction as a promising alternative to traditional cost estimation techniques in data management, offering improved accuracy by capturing complex interactions between queries, data, and runtime behavior. While initially developed for batch systems, LCMs are now increasingly applied to stream processing as well, where real-time demands pose new challenges. This tutorial presents the first unified overview of LCMs across both batch and stream processing systems, examining their role as essential components in modern query optimizers. We explore key aspects of LCM design—including input representations and model architectures—and highlight how these models deal with query optimization tasks.

PVLDB Reference Format:
Roman Heinrich, Xiao Li, Manisha Luthra, Zoi Kaoudi. Learned Cost Models for Query Optimization:
From Batch to Streaming Systems. PVLDB, 18(12): 5482 - 5487, 2025.
doi:10.14778/3750601.3750699

1 INTRODUCTION

The rise of learned cost models. In recent years, learned cost models (LCMs) have emerged as a powerful tool in data management, aiming to overcome the limitations of traditional, handcrafted cost estimation techniques. By leveraging machine learning (ML), LCMs can learn complex relationships among a query plan, data distribution, and runtime behavior, often achieving significantly improved accuracy. While their initial application was largely focused on batch processing systems, such as analytical databases and data warehouses, the growing complexity and real-time demands of modern applications have also spurred the development of LCMs for stream processing systems.

LCM as a core component for query optimization. As cost estimation is a core component of query optimization, LCMs are increasingly seen as essential building blocks for learned query optimizers. They are being used to replace or augment traditional cost models in critical optimization tasks—including plan enumeration and join ordering [8, 9, 33, 34, 50–52], operator placement [13, 28, 32], and parallelism tuning [1, 2] . These models not only improve plan quality but also open up new possibilities for optimization strategies that generalize across queries, datasets, and even hardware [13, 17] or user-defined functions [47].

No unified overview of LCMs for query optimizers. Despite rapid advances in this area, a clear gap remains: while previous tutorials have explored LCMs within the scope of ML for databases, there is no unified overview of how LCMs contribute to the various components of query optimization–especially across both batch and stream processing paradigms. This tutorial addresses that gap by presenting a structured and comparative perspective on learned cost modeling for both batch and stream processing systems. We highlight the design goals, input representations, model architectures, learning objectives, and use cases of existing approaches, with a special focus on their integration into query optimizers.

Related tutorials. There have been numerous tutorials that discuss various aspects at the intersection of data systems and ML [10, 24–26] in the past years. In the past, some tutorials touch upon the topic of learned query optimizers [20, 43, 56, 57], yet to the best of our knowledge, this is the first tutorial to offer a focused joint view of LCMs for both batch and stream processing. Existing tutorials focus on how to incorporate ML into query optimization for databases [24], the design of learned query optimizers [56, 57], or robustness in learned query optimizers [20]. On the other hand, our tutorial zooms in on the different types of LCMs and query plan featurization methods while it provides an insightful overview for both batch and streaming systems.

Targeted audience and required background. The tutorial targets researchers, developers, and system architects who are keen to know the state-of-the-art LCMs in both batch and stream processing scenarios. The tutorial requires the audience to have some familiarity with basic data management and ML concepts.

2 TUTORIAL OUTLINE

The tutorial is intended for 3 hours with a survey-based format providing a unified overview of LCMs for query optimization in both batch and stream processing systems. We structure it as follows:

- **Part I: Motivation & Background:** We motivate the need for LCMs and provide a background on query optimization, both traditional and learned, as well as on foundational cost models.
- **Part II: LCMs for Batch Data:** In this part, we discuss works focusing on LCMs in general which can be applied to query optimization as well as on LCMs proposed within learned query optimizers. This part focuses on batch data systems.
- **Part III: LCMs for Streaming Data:** Similarly, we discuss works for stream processing systems that propose LCMs in general which

† Corresponding author.

Proceedings of the VLDB Endowment, Vol. 18, No. 12 ISSN 2150-8097.
doi:10.14778/3750601.3750699

can be applied to query optimization as well as on LCMs proposed for distributed query optimization which concerns the tasks of operator placement and parallelism.

- **Part IV: Open Problems and Future Directions:** At the end of the tutorial, we will highlight the open challenges and suggest future directions for research.

2.1 Background

Query optimization is a core database process dating back many years [19]. Given a query, there are many plans that a data system can execute to get the results. All plans are equivalent in terms of their output but not in terms of their cost, i.e., the time required to compute the final results. Initially, query optimizers were rule-based; they were relying solely on heuristics and rules that would transform a logical plan to an execution one (e.g., push down selection predicates). Later on, cost-based optimization came to replace or augment rule-based ones. Traditional cost-based query optimizers rely on analytical cost models (mathematical formulas) which estimate the cost of an execution (sub)plan. In the beginning of the tutorial we will revise the main analytical formulas introduced for System R [41] and are still used in practice along with their drawbacks and limitations.

Learned query optimization became recently the third approach for query optimization after the rule- and cost-based ones. The first ideas relied on replacing different components of the query optimizer with ML models. For instance, first works proposed end-to-end learned query optimizers which replaced the plan enumeration with reinforcement learning and the cost model and cardinality estimator with ML models [34]. Then, other works focused on one part of the optimizer, for example, replacing join order enumeration with reinforcement learning [52], replacing the cost model with an ML model [12, 17, 21, 29, 35, 39, 42], or replacing cardinality estimation analytical models with ML models [22]. While the first approaches aimed at replacing parts or the entire traditional query optimizer, later works aim at using the traditional query optimizer and leverage ML to guide it via hints [33]. We will provide a quick overview of these different approaches and show that regardless the approach followed, in all of these works there is an LCM which is used for estimating costs of the plans.

2.2 LCMs in Batch Systems

In this part, we introduce LCMs for query optimization in batch data systems. There exist many studies on this topic [8, 9, 12, 16, 17, 22, 29, 33–35, 39, 40, 42, 48, 50–52, 54], among which most LCMs aim at traditional databases and only few of them target cloud databases, such as Amazon Redshift [40, 48], or cross-platform systems as Apache Wayang [5, 6, 21]. Thus, in this tutorial, we mainly discuss LCMs for traditional databases.

In traditional databases, LCMs aim to replace the heuristic or formula-based cost models to help in the generation and selection of better execution plans. Research revolving around this target can be generally divided into two groups: **LCM in general** and **LCM in learned query optimizers**. The former group formulated their research problems as cost estimations and designed ML models to predict the costs of plans. Although these studies are not directly targeted at query optimization tasks, their proposed cost estimation modes can be used for query optimization tasks to guide plan selection or pruning. On the contrary, the latter group devise their research problems into designing learned query optimizers and in their frameworks they propose LCMs to evaluate the costs of generated (sub)plans.

Next, we first present some typical LCM studies in each of the two groups. Subsequently, we make a taxonomy of LCMs in both groups in Table 1 according to their characteristics.

2.2.1 LCMs in general. In this group of LCM studies, the main purpose is the design of an effective ML model to accurately predict the costs of plans [12, 17, 22, 29, 35, 39, 42, 54]. For example, Ganapathi et al. [12] represent the physical query plan as a fixed-size vector with elements denoting the counts of query operator instances, and leverage a regression tree model to predict the cost of a query plan. Kipf et al. [22] propose a deep-set based neural network and encode the SQL query instead of its physical query plan as the input of the model. Though the model is originally designed for cardinality estimation, it has also been used for cost estimation [17, 42, 49]. The study [42] encodes the plan structure using a tree-based neural network while another study [35] also models the plan structure but takes a more modular approach called neural units. Both have shown boosting in the performance with plan structure modeling. Moreover, Zhao et al. [54] propose a general query plan representation method based on a tree-structured transformer and it has been tested on various tasks including cost estimation to show the effectiveness of the representation method. Hilprecht et al. proposed a database-agnostic model that only encodes transferable features so that the model can generalize across databases. Also, Liang et al. proposed another database-agnostic model that employs self-attention and tree-structured attention mechanisms. Finally, Rieger et al. proposed the most recent cost estimation model that decomposes a query plan into pipelines and leverages a regression tree to predict the execution time of each pipeline individually. To better compare these proposed LCMs, we will present a taxonomy of these LCMs in Section 2.2.3.

2.2.2 LCMs in learned query optimizers. Different from the above group of studies which primarily focus on how to design effective LCMs for accurate cost estimation, the studies in the current group target the task of query optimization, and their main focus is to devise effective query optimizers [8, 9, 33, 34, 50–52]. However, in their proposed query optimizers, they usually design a cost model based on ML to estimate the costs of (sub)plans to help the query optimizer select better plans or subplans. For instance, Marcus et al. [34] propose the first end-to-end query optimizer based on reinforcement learning. As the value network of their framework, a tree convolution neural network (i.e., tree-CNN) is utilized to predict the cost of plans. Also, Yu et al. [52] present a learned query optimizer that uses reinforcement learning (short for RL) for join order selection. It employs a tree-structured long short-term memory (i.e., tree-LSTM) network to estimate the cost of a join plan at a specific join state. Marcus et al. [33] introduced an ML-aided query optimizer that uses optimized hints to steer a traditional query optimizer. Similar to [34], their framework also uses a tree-CNN to estimate the cost of a query plan. However, more transferable features such as cardinality and cost estimates are considered, which enables this model to generalize across different database instances.

Table 1: Taxonomy of LCMs in Batch Data Systems.

Model	Input Features					Query Plan Representation	Model Architecture	DB-agnostic	Original Task
	Query Encode	Plan Encode	Cardinality Estimates	DB Cost Estimates	DB Statistics				
Flat Vector [12]		✓	✓			Flat	Regression Tree	Yes	Cost Estimation
MSCN [22]	✓				✓	Flat	Deep Sets	No	Card./Cost Estimation
End-to-End [42]	✓	✓			✓	Graph	TreeNN	No	Cost Estimation
QPP-Net [35]		✓	✓	✓	✓	Graph	Neural Unit Tree	No	Cost Estimation
QueryFormer [54]	✓	✓			✓	Graph	Transformer	No	General Purpose
Zero-Shot [17]		✓	✓		✓	Graph	GNN	Yes	Cost Estimation
DACE [29]		✓	✓	✓		Graph	Transformer	Yes	Cost Estimation
T3 [39]		✓	✓			Pipeline	Regression Tree	Yes	Cost Estimation
NEO [34]	✓	✓			✓	Graph	Tree-CNN + RL	No	Query Optimizer
RTOS [52]	✓	✓				Graph	Tree-LSTM + RL	No	Join Order Selection
Bao [33]		✓	✓	✓		Graph	Tree-CNN + RL	Yes	Query Optimizer
Balsa [50]	✓	✓	✓			Graph	Tree-CNN + RL	No	Query Optimizer
HybridQO [51]	✓	✓	✓	✓		Graph	Tree-LSTM + MHPE	No	Query Optimizer
LEON [9]	✓	✓	✓	✓		Graph	Tree-CNN + LTR	No	Query Optimizer
LOGER [8]	✓	✓				Graph	Tree-LSTM + RL	Yes	Query Optimizer

In the study [50], the authors also leverage a tree-CNN to predict the cost of a subplan but differing from previous studies they adopt a two-step training strategy (i.e., bootstrapping and fine-tuning) which makes their latency predictions more accurate. Also, Yu et al. [51] combine tree-LSTM networks with a multi-head performance estimator (short for MHPE) to predict the uncertainty-aware costs of a query plan. Their cost model outputs both estimated costs and variances that indicate the uncertainty of the predictions. Chen et al. [9] propose a learning-to-rank (short for LTR)–based cost model by which they predict a calibration coefficient that can be multiplied by the cost estimates of traditional query optimizers to obtain a better prediction of costs. In the proposed query optimizer by Chen et al. [8], a cost model is built based on a state and action network with a tree-LSTM network used for plan node representation. In the following, we will introduce a taxonomy of all the LCMs we have discussed in batch data systems.

2.2.3 Taxonomy of LCMs in batch data systems. To more clearly show the similarities and differences in the design of these LCMs in batch data systems, we make a taxonomy from different dimensions of these LCMs as shown in Table 1.

• **Input Features**. The input features may include information from different aspects such as query (e.g., predicates encoding), plan (e.g., physical operators encoding), cardinality estimates, DB cost estimates, or DB statistics (e.g., histograms or bitmaps). Different LCMs may choose subsets of these features as their input. It is noteworthy that the query plan is always considered as an input feature and is somehow encoded, except for the MSCN model which takes the SQL query encoding instead of the plan encoding as its input features. Also, cardinality estimates have been frequently taken as input features given that this dimension of features can enhance the performance of cost estimation considerably [16, 23].
• **Query Plan Representation.** The query plan is one of the input features of most LCMs. However, the ways of representing the query plans may differ among LCMs. For instance, earlier methods such as Flat Vector [12] and MSCN [22] represent the query plan in a flat way while most methods in Table 1 represent query plans as graphs. This is due to the fact that the query plan is inherently tree-structured. It is noteworthy that the latest work [39] represents the query plan as pipelines which is demonstrated to improve the accuracy of cost estimation.
• **Model Architecture.** For the LCMs whose original tasks are cost estimations, their model architectures differ significantly. They cover different learning approaches such as regression trees, deep sets, or transformers in order to enhance the performance of cost estimation. However, for the LCMs that are embedded in the query optimizers, Tree-CNN and Tree-LSTM networks are used more often. While most of them are combined with reinforcement learning (RL), a few of them consider different frameworks such as learning to rank (LTR) [9] or multi-head performance estimator (MHPE) [51].
• **Database Agnostic.** Whether an LCM is database agnostic determines whether it can be generalized across different databases. To design a database-agnostic LCM, those database specific features should not be taken into account such as table identifiers or columns. For instance, Bao [33] adopts a strategy of feature representation that is agnostic to the underlying schemas. Also, transfer learning techniques such as zero-shot learning [17] or pretraining [29] can be considered in the meantime.

2.3 LCMs in Streaming Systems

Cost estimation is a cornerstone of query optimization not only in batch data systems but also in stream processing systems. While mature in traditional databases, cost models for data stream processing systems (DSPS) remain underexplored [15, 38]. This is due to the dynamic nature of data streams, the heuristic-driven methods readily becomes inaccurate under shifting workloads [3]. This makes use of LCMs in streaming even more important such that they can learn from complex behavior of streaming workloads and better predict systems performance. Thus, in this tutorial we study existing work that has proposed use of LCMs to predict query latency, resource usage, and other execution costs, offering higher accuracy and adaptability in DSPS as summarized in Table 2. With regard to query optimization, this tutorial focuses on peer-reviewed work that applies ML to cost modeling in streaming engines, including its use cases of query optimization on operator placement and parallelism tuning, the two main components of streaming engines. Hence, we organize and discuss existing LCMs in DSPS in these two main categories similarly to the one for traditional databases.

2.3.1 LCMs in general. In this category, we present existing LCMs that predict performance metrics for DSPS such as processing times,

Table 2: Taxonomy of LCMs in Data Streaming Systems

Model	Features	Model Architecture	Intended Task
Moira [11]	Stream Statistics Hardware Monitoring	Support Vector Machine	Cost Estimation (resource, latency and throughput)
Imai et al. [18]	Hardware Characteristics Hardware Utilization	Linear Regression	Cost Estimation (throughput)
Li et al. [27]	Hardware Characteristics Hardware Utilization	Support Vector Regression	Cost Estimation (latency)
ZeroTune [2]	Query Plan, Stream Statistics Hardware Characteristics	Graph Neural Networks	Operator Parallelism
COSTREAM [13]	Query Plan Stream Statistics Hardware Characteristics	Graph Neural Networks	Operator Placement
Li et al [28]	Hardware Utilization Stream Characteristics	Reinforcement Learning	Operator Placement
Decima [32]	Hardware Utilization Stream Statistics	Reinforcement Learning	Operator Placement
Ni et al. [37]	Hardware Utilization Stream Statistics	Reinforcement Learning	Operator Placement

throughput or classify backpressure [4, 11, 15, 18, 27]. For example, Li et al. [27] propose a topology-aware method to predict the average tuple processing time for a given scheduling of operators based on the current topology and runtime statistics. Imai et al. [18] present a model to predict the throughput of the system using linear regression while minimizing time and cost for model training. Alnafessah et al. [4] use Bayesian optimization to find an optimal system configuration in streaming. Foroni et al. [11] introduced an incremental learning based approach for dynamically estimating costs of a query by taking into account the changes in the data streams. Finally, the authors of this tutorial paper propose a generalizable LCM to estimate costs of data streaming query such as latency that can predict costs for unseen workloads out-of-the-box [15].

2.3.2 LCMs for query optimization (placement and parallelism). In this category, we present approaches that apply costs derived from an LCM for query optimization tasks of DSPS such as to identify an optimal operator placement or parallelism level [1, 2, 13, 28, 31, 32, 37]. We have contributed to a major proportion of approaches in this category [1, 2, 13] that leverages a zero-shot learning-based cost model [15] for optimization decisions like placement [13] and parallelism tuning [1, 2]. The main idea of the generalizable cost model is to learn from a broad training dataset and the joint graph representation encoding in graph neural network that enables learning from the placement and parallelism decisions. Furthermore, the authors have proposed adaptive placement approaches previously [30, 31] where ML-based cost derivation was used for placement approach selection. The interest in generalizable placement has been also identified by other authors from IBM [37] that propose graph-aware encoder-decoder architecture using deep reinforcement learning to find optimized solutions for placement for unseen queries. Moreover, the use of deep reinforcement learning for placement [28, 32] and parallelism [53] has been motivated by several other authors.

2.4 Open Problems & Future Directions

Although there are already many proposed solutions for LCMs in the literature, there are still many open problems that need to be tackled. We highlight some of them in the following.

How to collect training data efficiently? A hurdle that comes with ML models in general is the collection of training data. Although unsupervised methods exist, many of the proposed techniques are based on supervised learning models which require the collection of ground-truth labels. These labels typically include the runtime of a large number of execution plans which shall be both optimal and suboptimal. This can lead to prohibitively large execution times. Although there have been initial efforts towards this goal [44–46], there are still many open issues such as aligning the label collection to the hardware of the production environment.

How to train LCMs for query optimization? Since the goal in query optimization is to find a good plan, what really matters in query optimization is the relative order of the execution plans and not their estimated cost or runtime. Thus, there have been initial works that focus on learning-to-rank models and incorporate them in query optimization [7, 55]. Still, there has not been done a thorough study of how learning-to-rank models fit with current plan enumeration algorithms or if new algorithms shall be devised. Moreover, it has been recently shown that current LCMs are not really good at query optimization tasks like join ordering [16]. Better LCMs are needed to meet the need of current query optimizers.

How to evaluate LCMs appropriately? Q-error [36] is often used in the literature as the metric to evaluate whether an LCM is performing well. While this metric makes sense if we aim at comparing the accuracy of LCMs, it is not representative for measuring the impact of an LCM on query optimization. Hence there is a need for defining new metrics tailored for query optimization [16].

How to make LCMs interpretable? Most of the existing LCMs are based on neural models that are mainly black boxes and hence cannot be easily debugged or trusted. So far there has been no work on models for data systems. Thus another open research direction is how to make LCMs more explainable and interpretable. One possible way of doing that would be to combine learned models with interpretable representations (e.g., symbolic models) to explain why specific costs are predicted that way. Further, explainers specific to the model architecture can be adapted to explain the predictions of the specific data system [14].

3 PRESENTERS

Roman Heinrich is a PhD candidate at the Systems Group at TU Darmstadt and DFKI Darmstadt, focusing on learned cost estimation and query optimization for data systems.

Xiao Li is a postdoctoral researcher at IT University of Copenhagen. His research topic is learned query optimization for data systems.

Manisha Luthra is a research group lead at DFKI and at the Systems Group in TU Darmstadt. She is an expert on streaming systems and has proposed several early works on learned cost modeling and optimization of streaming systems. She has received several prestigious awards including the German National award for her dissertation and the Athena Young Investigator award.

Zoi Kaoudi is an Associate Professor at the IT University of Copenhagen. Her research interests lie at the intersection of machine learning, data management with a focus on query optimization, and knowledge graphs. She has previously presented tutorials at ICDE 2013, SIGMOD 2014, ICDE 2018, and VLDB 2022. She has received the best demonstration award at ICDE 2022.

ACKNOWLEDGMENTS

This research is funded by Carlsberg Foundation (CF-23-0836), the Athene Young Investigator Programme of TU Darmstadt, etaGPT project under grant number 03EN4107 and DFKI Darmstadt.

REFERENCES

[1] Pratyush Agnihotri, Boris Koldehofe, Carsten Binnig, and Manisha Luthra. 2023. Zero-Shot Cost Models for Parallel Stream Processing. In *Proceedings of the Sixth International Workshop on Exploiting Artificial Intelligence Techniques for Data Management, aiDM@SIGMOD 2023, Seattle, WA, USA, 18 June 2023*, Rajesh Bordawekar, Oded Shmueli, Yael Amsterdamer, Donatella Firmani, and Andreas Kipf (Eds.). ACM, 5:1–5:5. https://doi.org/10.1145/3593078.3593934

[2] Pratyush Agnihotri, Boris Koldehofe, Paul Stiegele, Roman Heinrich, Carsten Binnig, and Manisha Luthra. 2024. ZERoTuNE: Learned Zero-Shot Cost Models for Parallelism Tuning in Stream Processing. In *40th IEEE International Conference on Data Engineering, ICDE 2024, Utrecht, The Netherlands, May 13-16, 2024*. IEEE, 2040–2053. https://doi.org/10.1109/ICDE60146.2024.00163

[3] Alexander Alexandrov, Rico Bergmann, Stephan Ewen, Johann-Christoph Freytag, Fabian Hueske, Arvid Heise, Odej Kao, Marcus Leich, Ulf Leser, Volker Markl, Felix Naumann, Mathias Peters, Astrid Rheinländer, Matthias J. Sax, Sebastian Schelter, Mareike Höger, Kostas Tzoumas, and Daniel Warneke. 2014. The Stratosphere platform for big data analytics. *VLDB J.* 23, 6 (2014), 939–964. https://doi.org/10.1007/S00778-014-0357-Y

[4] Ahmad Alnafessah, Gabriele Russo Russo, Valeria Cardellini, Giuliano Casale, and Francesco Lo Presti. 2021. *AI-Driven Performance Management in Data-Intensive Applications*. John Wiley & Sons, Ltd, Chapter 9, 199–222. https://doi.org/10.1002/9781119675525.ch9

[5] Kaustubh Beedkar, Aurélien Bertrand, Haralampos Gavriilidis, Augusto José Fonseca, Zoi Kaoudi, Mingxi Liu, Volker Markl, Juri Petersen, Fábio Porto, Víctor Ribeiro, Mads Sejer Pedersen, Lucas Giusti Tavares, Michalis Vargiamis, and Chen Xu. 2025. Apache Wayang in Action: Enabling Data Systems Integration via a Unified Data Analytics Framework. In *Companion of the 2025 International Conference on Management of Data, SIGMOD/PODS 2025, Berlin, Germany, June 22-27, 2025*, Volker Markl, Joseph M. Hellerstein, and Azza Abouzied (Eds.). ACM, 35–38. https://doi.org/10.1145/3722212.3725081

[6] Kaustubh Beedkar, Bertty Contreras-Rojas, Haralampos Gavriilidis, Zoi Kaoudi, Volker Markl, Rodrigo Pardo-Meza, and Jorge-Arnulfo Quiané-Ruiz. 2023. Apache Wayang: A Unified Data Analytics Framework. *SIGMOD Rec.* 52, 3 (2023), 30–35. https://doi.org/10.1145/3631504.3631510

[7] Henriette Behr, Volker Markl, and Zoi Kaoudi. 2023. Learn What Really Matters: A Learning-to-Rank Approach for ML-based Query Optimization. In *Datenbanksysteme für Business, Technologie und Web (BTW 2023), 20. Fachtagung des GI-Fachbereichs „Datenbanken und Informationssysteme" (DBIS), 06.-10, März 2023, Dresden, Germany, Proceedings (LNI)*, Birgitta König-Ries, Stefanie Scherzinger, Wolfgang Lehner, and Gottfried Vossen (Eds.), Vol. P-331. Gesellschaft für Informatik e.V., 535–554. https://doi.org/10.18420/BTW2023-25

[8] Tianyi Chen, Jun Gao, Hedui Chen, and Yaofeng Tu. 2023. LOGER: A Learned Optimizer towards Generating Efficient and Robust Query Execution Plans. *Proc. VLDB Endow.* 16, 7 (2023), 1777–1789. https://doi.org/10.14778/3587136.3587150

[9] Xu Chen, Haitian Chen, Zibo Liang, Shuncheng Liu, Jinghong Wang, Kai Zeng, Han Su, and Kai Zheng. 2023. LEON: A New Framework for ML-Aided Query Optimization. *Proc. VLDB Endow.* 16, 9 (2023), 2261–2273. https://doi.org/10.14778/3598581.3598597

[10] Gao Cong, Jingyi Yang, and Yue Zhao. 2024. Machine Learning for Databases: Foundations, Paradigms, and Open problems. In *Companion of the 2024 International Conference on Management of Data, SIGMOD/PODS 2024, Santiago, Chile, June 9-15, 2024*, Pablo Barceló, Nayat Sánchez-Pi, Alexandra Meliou, and S. Sudarshan (Eds.). ACM, 622–629. https://doi.org/10.1145/3626246.3654686

[11] Daniele Foroni, Cristian Axenie, Stefano Bortoli, Mohamad Al Hajj Hassan, Ralph Acker, Radu Tudoran, Goetz Brasche, and Yannis Velegrakis. 2018. Moira: A Goal-Oriented Incremental Machine Learning Approach to Dynamic Resource Cost Estimation in Distributed Stream Processing Systems. In *Proceedings of the International Workshop on Real-Time Business Intelligence and Analytics, BIRTE 2018, Rio de Janeiro, Brazil, August 27, 2018*, Malú Castellanos, Panos K. Chrysanthis, Badrish Chandramouli, and Shimin Chen (Eds.). ACM, 2:1–2:10. https://doi.org/10.1145/3242153.3242160

[12] Archana Ganapathi, Harumi A. Kuno, Umeshwar Dayal, Janet L. Wiener, Armando Fox, Michael I. Jordan, and David A. Patterson. 2009. Predicting Multiple Metrics for Queries: Better Decisions Enabled by Machine Learning. In *Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, March 29 2009 - April 2 2009, Shanghai, China*, Yannis E. Ioannidis, Dik Lun Lee, and Raymond T. Ng (Eds.). IEEE Computer Society, 592–603. https://doi.org/10.1109/ICDE.2009.130

[13] Roman Heinrich, Carsten Binnig, Harald Kornmayer, and Manisha Luthra. 2024. Costream: Learned Cost Models for Operator Placement in Edge-Cloud Environments. In *40th IEEE International Conference on Data Engineering, ICDE 2024, Utrecht, The Netherlands, May 13-16, 2024*. IEEE, 96–109. https://doi.org/10.1109/ICDE60146.2024.00015

[14] Roman Heinrich, Oleksandr Havrylov, Manisha Luthra, Johannes Wehrstein, and Carsten Binnig. 2025. Opening The Black-Box: Explaining Learned Cost Models for Databases. *Proc. VLDB Endow.* 18, 12 (2025).

[15] Roman Heinrich, Manisha Luthra, Harald Kornmayer, and Carsten Binnig. 2022. Zero-shot cost models for distributed stream processing. In *16th ACM International Conference on Distributed and Event-based Systems, DEBS 2022, Copenhagen, Denmark, June 27 - 30, 2022*, Yongluan Zhou, Panos K. Chrysanthis, Vincenzo Gulisano, and Eleni Tzirita Zacharatou (Eds.). ACM, 85–90. https://doi.org/10.1145/3524860.3539639

[16] Roman Heinrich, Manisha Luthra, Johannes Wehrstein, Harald Kornmayer, and Carsten Binnig. 2025. How Good are Learned Cost Models, Really? Insights from Query Optimization Tasks. *Proc. ACM Manag. Data* 3, 3, Article 172 (June 2025), 27 pages. https://doi.org/10.1145/3725309

[17] Benjamin Hilprecht and Carsten Binnig. 2022. Zero-Shot Cost Models for Out-of-the-box Learned Cost Prediction. *Proc. VLDB Endow.* 15, 11 (2022), 2361–2374. https://doi.org/10.14778/3551793.3551799

[18] Shigeru Imai, Stacy Patterson, and Carlos A. Varela. 2017. Maximum Sustainable Throughput Prediction for Data Stream Processing over Public Clouds. In *Proceedings of the 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGRID 2017, Madrid, Spain, May 14-17, 2017*. IEEE Computer Society / ACM, 504–513. https://doi.org/10.1109/CCGRID.2017.105

[19] Yannis E. Ioannidis. 1997. Query Optimization. In *The Computer Science and Engineering Handbook*, Allen B. Tucker (Ed.). CRC Press, 1038–1057.

[20] Amin Kamali, Verena Kantere, and Calisto Zuzarte. 2024. Robust Query Optimization in the Era of Machine Learning: State-of-the-Art and Future Directions. In *40th IEEE International Conference on Data Engineering, ICDE 2024, Utrecht, The Netherlands, May 13-16, 2024*. IEEE, 5371–5375. https://doi.org/10.1109/ICDE60146.2024.00408

[21] Zoi Kaoudi, Jorge-Arnulfo Quiané-Ruiz, Bertty Contreras-Rojas, Rodrigo Pardo-Meza, Anis Troudi, and Sanjay Chawla. 2020. ML-based Cross-Platform Query Optimization. In *36th IEEE International Conference on Data Engineering, ICDE 2020, Dallas, TX, USA, April 20-24, 2020*. IEEE, 1489–1500. https://doi.org/10.1109/ICDE48307.2020.00132

[22] Andreas Kipf, Thomas Kipf, Bernhard Radke, Viktor Leis, Peter A. Boncz, and Alfons Kemper. 2019. Learned Cardinalities: Estimating Correlated Joins with Deep Learning. In *9th Biennial Conference on Innovative Data Systems Research, CIDR 2019, Asilomar, CA, USA, January 13-16, 2019, Online Proceedings*. www.cidrdb.org. http://cidrdb.org/cidr2019/papers/p101-kipf-cidr19.pdf

[23] Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter A. Boncz, Alfons Kemper, and Thomas Neumann. 2015. How Good Are Query Optimizers, Really? *Proc. VLDB Endow.* 9, 3 (2015), 204–215. https://doi.org/10.14778/2850583.2850594

[24] Guoliang Li and Xuanhe Zhou. 2022. Machine Learning for Data Management: A System View. In *38th IEEE International Conference on Data Engineering, ICDE 2022, Kuala Lumpur, Malaysia, May 9-12, 2022*. IEEE, 3198–3201. https://doi.org/10.1109/ICDE53745.2022.00297

[25] Guoliang Li, Xuanhe Zhou, and Lei Cao. 2021. AI Meets Database: AI4DB and DB4AI. In *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh Srivastava (Eds.). ACM, 2859–2866. https://doi.org/10.1145/3448016.3457542

[26] Guoliang Li, Xuanhe Zhou, and Lei Cao. 2021. Machine Learning for Databases. *Proc. VLDB Endow.* 14, 12 (2021), 3190–3193. https://doi.org/10.14778/3476311.3476405

[27] Teng Li, Jian Tang, and Jielong Xu. 2016. Performance Modeling and Predictive Scheduling for Distributed Stream Data Processing. *IEEE Trans. Big Data* 2, 4 (2016), 353–364. https://doi.org/10.1109/TBDATA.2016.2616148

[28] Teng Li, Zhiyuan Xu, Jian Tang, and Yanzhi Wang. 2018. Model-free Control for Distributed Stream Data Processing using Deep Reinforcement Learning. *Proc. VLDB Endow.* 11, 6 (2018), 705–718. https://doi.org/10.14778/3184470.3184474

[29] Zibo Liang, Xu Chen, Yuyang Xia, Runfan Ye, Haitian Chen, Jiandong Xie, and Kai Zheng. 2024. DACE: A Database-Agnostic Cost Estimator. In *40th IEEE International Conference on Data Engineering, ICDE 2024, Utrecht, The Netherlands, May 13-16, 2024*. IEEE, 4925–4937. https://doi.org/10.1109/ICDE60146.2024.00374

[30] Manisha Luthra, Boris Koldehofe, Niels Danger, Pascal Weisenburger, Guido Salvaneschi, and Ioannis Stavrakakis. 2021. TCEP: Transitions in operator placement to adapt to dynamic network environments. *J. Comput. Syst. Sci.* 122 (2021), 94–125. https://doi.org/10.1016/j.jcss.2021.05.003

[31] Manisha Luthra, Boris Koldehofe, Pascal Weisenburger, Guido Salvaneschi, and Raheel Arif. 2018. TCEP: Adapting to Dynamic User Environments by Enabling Transitions between Operator Placement Mechanisms. In *Proceedings of the 12th ACM International Conference on Distributed and Event-based Systems, DEBS 2018, Hamilton, New Zealand, June 25-29, 2018*, Annika Hinze, David M. Eyers, Martin Hirzel, Matthias Weidlich, and Sukanya Bhowmik (Eds.). ACM, 136–147. https://doi.org/10.1145/3210284.3210292

[32] Hongzi Mao, Malte Schwarzkopf, Shaileshh Bojja Venkatakrishnan, Zili Meng, and Mohammad Alizadeh. 2019. Learning scheduling algorithms for data processing clusters. In *Proceedings of the ACM Special Interest Group on Data Communication, SIGCOMM 2019, Beijing, China, August 19-23, 2019*, Jianping Wu and Wendy Hall (Eds.). ACM, 270–288. https://doi.org/10.1145/3341302.3342080

[33] Ryan Marcus, Parimarjan Negi, Hongzi Mao, Nesime Tatbul, Mohammad Alizadeh, and Tim Kraska. 2021. Bao: Making Learned Query Optimization Practical. In *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh

Srivastava (Eds.). ACM, 1275–1288. https://doi.org/10.1145/3448016.3452838

[34] Ryan Marcus, Parimarjan Negi, Hongzi Mao, Chi Zhang, Mohammad Alizadeh, Tim Kraska, Olga Papaemmanouil, and Nesime Tatbul. 2019. Neo: A Learned Query Optimizer. *Proc. VLDB Endow.* 12, 11 (2019), 1705–1718. https://doi.org/10.14778/3342263.3342644

[35] Ryan Marcus and Olga Papaemmanouil. 2019. Plan-Structured Deep Neural Network Models for Query Performance Prediction. *Proc. VLDB Endow.* 12, 11 (2019), 1733–1746. https://doi.org/10.14778/3342263.3342646

[36] Guido Moerkotte, Thomas Neumann, and Gabriele Steidl. 2009. Preventing Bad Plans by Bounding the Impact of Cardinality Estimation Errors. *Proc. VLDB Endow.* 2, 1 (2009), 982–993. https://doi.org/10.14778/1687627.1687738

[37] Xiang Ni, Jing Li, Mo Yu, Wang Zhou, and Kun-Lung Wu. 2020. Generalizable Resource Allocation in Stream Processing via Deep Reinforcement Learning. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. AAAI Press, 857–864. https://doi.org/10.1609/aaai.v34i01.5431

[38] Tobias Pfandzelter, Sören Henning, Trever Schirmer, Wilhelm Hasselbring, and David Bermbach. 2022. Streaming vs. Functions: A Cost Perspective on Cloud Event Processing. In *IEEE International Conference on Cloud Engineering, IC2E 2022, Pacific Grove, CA, USA, September 26-30, 2022*. IEEE, 67–78. https://doi.org/10.1109/IC2E55432.2022.00015

[39] Maximilian Rieger and Thomas Neumann. 2025. T3: Accurate and Fast Performance Prediction for Relational Database Systems With Compiled Decision Trees. *Proc. ACM Manag. Data* 3, 3, Article 227 (June 2025), 27 pages. https://doi.org/10.1145/3725364

[40] Gaurav Saxena, Mohammad Rahman, Naresh Chainani, Chunbin Lin, George Caragea, Fahim Chowdhury, Ryan Marcus, Tim Kraska, Ippokratis Pandis, and Balakrishnan (Murali) Narayanaswamy. 2023. Auto-WLM: Machine Learning Enhanced Workload Management in Amazon Redshift. In *Companion of the 2023 International Conference on Management of Data, SIGMOD/PODS 2023, Seattle, WA, USA, June 18-23, 2023*, Sudipto Das, Ippokratis Pandis, K. Selçuk Candan, and Sihem Amer-Yahia (Eds.). ACM, 225–237. https://doi.org/10.1145/3555041.3589677

[41] Patricia G. Selinger, Morton M. Astrahan, Donald D. Chamberlin, Raymond A. Lorie, and Thomas G. Price. 1979. Access Path Selection in a Relational Database Management System. In *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data, Boston, Massachusetts, USA, May 30 - June 1*, Philip A. Bernstein (Ed.). ACM, 23–34. https://doi.org/10.1145/582095.582099

[42] Ji Sun and Guoliang Li. 2019. An End-to-End Learning-based Cost Estimator. *Proc. VLDB Endow.* 13, 3 (2019), 307–319. https://doi.org/10.14778/3368289.3368296

[43] Dimitris Tsesmelis and Alkis Simitsis. 2022. Database Optimizers in the Era of Learning. In *38th IEEE International Conference on Data Engineering, ICDE 2022, Kuala Lumpur, Malaysia, May 9-12, 2022*. IEEE, 3213–3216. https://doi.org/10.1109/ICDE53745.2022.00301

[44] Francesco Ventura, Zoi Kaoudi, Jorge-Arnulfo Quiané-Ruiz, and Volker Markl. 2021. Expand your Training Limits! Generating Training Data for ML-based Data Management. In *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh Srivastava (Eds.). ACM, 1865–1878. https://doi.org/10.1145/3448016.3457286

[45] Robin Van De Water, Francesco Ventura, Zoi Kaoudi, Jorge-Arnulfo Quiané-Ruiz, and Volker Markl. 2022. DataFarm: Farm Your ML-based Query Optimizer's Food! - Human-Guided Training Data Generation -. In *12th Conference on Innovative Data Systems Research, CIDR 2022, Chaminade, CA, USA, January 9-12, 2022*. www.cidrdb.org. https://www.cidrdb.org/cidr2022/papers/a37-water.pdf

[46] Robin Van De Water, Francesco Ventura, Zoi Kaoudi, Jorge-Arnulfo Quiané-Ruiz, and Volker Markl. 2022. Farming Your ML-based Query Optimizer's Food. In *38th IEEE International Conference on Data Engineering, ICDE 2022, Kuala Lumpur, Malaysia, May 9-12, 2022*. IEEE, 3186–3189. https://doi.org/10.1109/ICDE53745.2022.00294

[47] Johannes Wehrstein, Tiemo Bang, Roman Heinrich, and Carsten Binnig. 2025. GRACEFUL: A Learned Cost Estimator for UDFs . In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*. IEEE Computer Society, Los Alamitos, CA, USA, 2450–2463. https://doi.org/10.1109/ICDE65448.2025.00185

[48] Ziniu Wu, Ryan Marcus, Zhengchun Liu, Parimarjan Negi, Vikram Nathan, Pascal Pfeil, Gaurav Saxena, Mohammad Rahman, Balakrishnan Narayanaswamy, and Tim Kraska. 2024. Stage: Query Execution Time Prediction in Amazon Redshift. In *Companion of the 2024 International Conference on Management of Data, SIGMOD/PODS 2024, Santiago, Chile, June 9-15, 2024*, Pablo Barceló, Nayat Sánchez-Pi, Alexandra Meliou, and S. Sudarshan (Eds.). ACM, 280–294. https://doi.org/10.1145/3626246.3653391

[49] Jiani Yang, Sai Wu, Dongxiang Zhang, Jian Dai, Feifei Li, and Gang Chen. 2023. Rethinking Learned Cost Models: Why Start from Scratch? *Proc. ACM Manag. Data* 1, 4, 255:1–255:27. https://doi.org/10.1145/3626769

[50] Zongheng Yang, Wei-Lin Chiang, Sifei Luan, Gautam Mittal, Michael Luo, and Ion Stoica. 2022. Balsa: Learning a Query Optimizer Without Expert Demonstrations. In *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, Zachary G. Ives, Angela Bonifati, and Amr El Abbadi (Eds.). ACM, 931–944. https://doi.org/10.1145/3514221.3517885

[51] Xiang Yu, Chengliang Chai, Guoliang Li, and Jiabin Liu. 2022. Cost-based or Learning-based? A Hybrid Query Optimizer for Query Plan Selection. *Proc. VLDB Endow.* 15, 13 (2022), 3924–3936. https://doi.org/10.14778/3565838.3565846

[52] Xiang Yu, Guoliang Li, Chengliang Chai, and Nan Tang. 2020. Reinforcement Learning with Tree-LSTM for Join Order Selection. In *36th IEEE International Conference on Data Engineering, ICDE 2020, Dallas, TX, USA, April 20-24, 2020*. IEEE, 1297–1308. https://doi.org/10.1109/ICDE48307.2020.00116

[53] Eleni Zapridou, Ioannis Mytilinis, and Anastasia Ailamaki. 2022. Dalton: Learned Partitioning for Distributed Data Streams. *Proc. VLDB Endow.* 16, 3 (2022), 491–504. https://doi.org/10.14778/3570690.3570699

[54] Yue Zhao, Gao Cong, Jiachen Shi, and Chunyan Miao. 2022. QueryFormer: A Tree Transformer Model for Query Plan Representation. *Proc. VLDB Endow.* 15, 8 (2022), 1658–1670. https://doi.org/10.14778/3529337.3529349

[55] Rong Zhu, Wei Chen, Bolin Ding, Xingguang Chen, Andreas Pfadler, Ziniu Wu, and Jingren Zhou. 2023. Lero: A Learning-to-Rank Query Optimizer. *Proc. VLDB Endow.* 16, 6 (2023), 1466–1479. https://doi.org/10.14778/3583140.3583160

[56] Rong Zhu, Lianggui Weng, Bolin Ding, and Jingren Zhou. 2024. Learned Query Optimizer: What is New and What is Next. In *Companion of the 2024 International Conference on Management of Data, SIGMOD/PODS 2024, Santiago, Chile, June 9-15, 2024*, Pablo Barceló, Nayat Sánchez-Pi, Alexandra Meliou, and S. Sudarshan (Eds.). ACM, 561–569. https://doi.org/10.1145/3626246.3654692

[57] Rong Zhu, Ziniu Wu, Chengliang Chai, Andreas Pfadler, Bolin Ding, Guoliang Li, and Jingren Zhou. 2022. Learned Query Optimizer: At the Forefront of AI-Driven Databases. In *Proceedings of the 25th International Conference on Extending Database Technology, EDBT 2022, Edinburgh, UK, March 29 - April 1, 2022*, Julia Stoyanovich, Jens Teubner, Paolo Guagliardo, Milos Nikolic, Andreas Pieris, Jan Mühlig, Fatma Özcan, Sebastian Schelter, H. V. Jagadish, and Meihui Zhang (Eds.). OpenProceedings.org, 1–4. https://doi.org/10.48786/EDBT.2022.56