# Machine Learning for Databases: Foundations, Paradigms, and Open problems

Gao Cong
gaocong@ntu.edu.sg
Nanyang Technological University
Singapore

Jingyi Yang
jyang028@e.ntu.edu.sg
Nanyang Technological University
Singapore

Yue Zhao
zhao0342@e.ntu.edu.sg
Nanyang Technological University
Singapore

## ABSTRACT

This tutorial delves into the burgeoning field of Machine Learning for Databases (ML4DB), highlighting its recent progress and the challenges impeding its integration into industrial-grade database management systems. We systematically explore three key themes: the exploration of foundations in ML4DB and their potential for diverse applications, the two paradigms in ML4DB, *i.e.*, using machine learning as a replacement versus enhancement of traditional database components, and the critical open challenges such as improving model efficiency and addressing data shifts. Through an in-depth analysis, including a survey of recent works in major database conferences, this tutorial encapsulates the current state of ML4DB, as well as charts a roadmap for its future development and wider adoption in practical database environments.

## CCS CONCEPTS

• **Information systems → Data management systems**.

## KEYWORDS

Machine Learning for Databases, Learned Query optimization, Learned Index

**ACM Reference Format:**
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-Companion '24), June 9–15, 2024, Santiago, AA, Chile.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3626246.3654686

## 1 INTRODUCTION

The field of Machine Learning for Databases, or ML4DB, has witnessed substantial growth over the past few years. Machine learning techniques have been proposed to improve various aspects of a data management system, from database core components like index, data layout, scheduler, query optimizer, to database advisors like performance prediction, knob tuning, and index/view advisors.

Despite significant research effort and a growing body of literature, ML4DB is still in its nascent stages, with limited real-world adoption in industrial-grade data management systems. A few tutorials [13, 21–23, 41, 45, 47, 58] have either overviewed machine learning applications for different database components/problems, or focused on machine learning methods for a particular database problem such as query optimization. However, there has been a lack of in-depth analysis identifying common themes in the application of ML across various database components and problems. This tutorial aims to fill that gap by delineating three important themes in current ML4DB research.

*(1) ML4DB Foundations:* In recent years, there are many advances in ML4DB research, spanning challenging data management problems, including cost estimation [29, 38], query optimization [27, 28], index advisor [5, 37], and many others [52]. A typical ML4DB work focuses on a specific database problem, and henceforth design a machine learning solution to solve problem. However, it is unclear if there are some common components for foundation in the literature of the field. Database systems are renowned for their ability to provide abstraction, particularly through the use of relational models and relational operators. Therefore, in this part of the tutorial, we aim to generalize and give some foundations in ML4DB studies.

We identify two aspects for ML4DB foundations. First, we identify that query plan representation is a common first step in a bunch of ML4DB applications. These studies typically propose new approaches to represent query plans along with task models to solve their tasks. We systematically analyze the designs and behaviors of the representation component in ML4DB systems. Second, pretrained and unified models are proposed to tackle multiple data management problems. In this way, the expense of data collection and model training can be largely alleviated for a new task and dataset with few-shot learning techniques.

*(2) ML4DB Paradigms:* Initial ML4DB research predominantly adopted the "replacement" approach, where machine learning models substituted traditional database components. For example, the first learned index RMI [17] proposed to learn the cumulative density function of the data distribution as a replacement of B-tree index; the first learned query optimizer NEO [28] proposed to replace the traditional database query optimizer with a reinforcement learning based framework. This approach, however, has faced challenges in robustness and generalization capability. To address these challenges, recent research have adopted a different approach, which retains the traditional database components to leverage their expert domain knowledge, while using machine learning to improve their performance in a data-driven manner. This "ML-enhanced" paradigm has shown improved robustness and generalization capability, and some of the representative work, *e.g.*, BAO [27], has driven the adoption of ML4DB techniques in the industrial-grade systems [33, 54]. To analyze the two paradigms, we surveyed the publications from SIGMOD and VLDB since 2018, and counted the number of publications on machine learning for data index and query optimizer (QO) that adopts the "replacement" approach and



*SIGMOD-Companion '24, June 9–15, 2024, Santiago, AA, Chile*

ACM ISBN 979-8-4007-0422-2/24/06.
https://doi.org/10.1145/3626246.3654686

the "ML-enhanced" approach, respectively. The results are shown in Figure 1. From Figure 1, we observe a noticeable shift from the "replacement" paradigm to the "ML-enhanced" paradigm. We will provide an in-depth discussion on the two paradigms, with a focus on two database problems: database index and query optimizer. Our discussion will include an overview of methods based on the "replacement" approaches, the limitation of these methods, and how ML-enhanced methods would address these limitations.

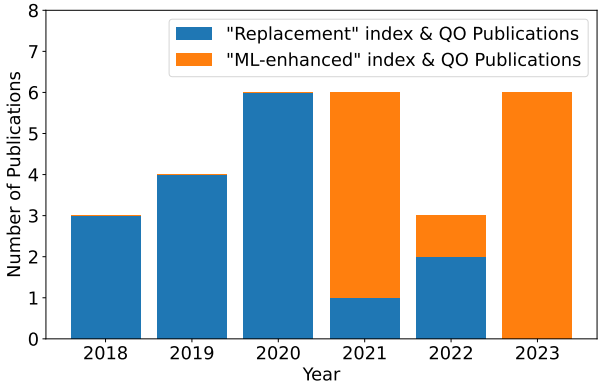


**Figure 1: Publication trend in machine learning for index & Query Optimizer (QO)**

*(3) ML4DB Open Problems:* Several key problems need to be solved in order to make ML4DB more practical and ready for mainstream adoption. **1) Model Efficiency.** Traditional database components are highly optimized in terms of both space efficiency and time efficiency. To become an integral part of the database management system, the machine learning model must also be optimized in terms of model size, and training & inference efficiency. We will discuss advancements in lightweight ML4DB models, as well as techniques to improve the training & inference speed of ML4DB models. **2) Handling Data and workload Shift.** Machine learning models are often trained on static datasets, whereas databases experience continuous changes in query workloads and data. For models trained from a historical workload, potential workload shifts may impact the model performance; while for models trained from a given data instance, database insertion and updates must be taken into account. We introduce approaches that address these dynamic conditions, ensuring sustained model performance. **3) Foundation models for ML4DB tasks.** Current ML4DB models excel in single task and dataset which limit the applicability in deployment. New techniques can be developed to train the models from diverse training data and capture general features, so that it can adapt to different datasets. Techniques such as meta-learning can be explored as well, so that the trained model can generalize across tasks. **4) Generating training data of high quality.** Training data is costly to collect due to the need of executing query workloads, and many ML4DB studies are limited to either synthetic datasets or small real-world datasets, which may not fully reflect the behavior of the practical systems. At the same time, large database with real customer data is not transparent to database tuners due to privacy constraints. Hence, developing methodologies for creating, augmenting, or simulating realistic and privacy-compliant datasets at low cost is crucial and remains an open problem.

## 2 TARGET AUDIENCE AND LENGTH

**Target Audience.** The intended audience includes SIGMOD attendees from both research and industry communities who are interested in either existing tools/techniques on ML4DB or promising research directions on ML4DB. We will not require any prior background knowledge in database or machine learning domain. The tutorial will be self-contained, and we will include a broad introduction and motivating examples for non-specialists to follow.
**Tutorial Breakdown.** The intended length of this tutorial is 3 hours. We plan to break down the tutorial into 3 sections.
Section 1: ML4DB Foundations (*75 mins*)

(1) Overview of ML4DB systems. (*5 mins*)
(2) Query plan representation problem and techniques. (*10 mins*)
(3) Feature encoding. (*10 mins*)
(4) Tree models. (*10 mins*)
(5) Comparative analysis. (*15 mins*)
(6) Pretrained models. (*25 mins*)

Section 2: ML4DB Paradigms. (*75 mins*) The second section will present the two paradigms in ML4DB research: "replacement" versus "ML-enhanced".

(1) Overview of ML4DB paradigms. (*5 mins*)
(2) Learned index basics. (*10 mins*)
(3) ML-enhanced index insertion. (*10 mins*)
(4) ML-enhanced index bulkloading. (*10 mins*)
(5) ML-enhanced index search. (*10 mins*)
(6) Learned query optimizer basics. (*10 mins*)
(7) Bandit optimizer. (*10 mins*)
(8) ML-enhanced query optimization. (*10 mins*)

Section 3: Open Problems. The third section covers some key problems in ML4DB that should be addressed. (*30 mins*)

(1) Model efficiency (*10 mins*)
(2) Handling data & workload shifts (*10 mins*)
(3) Foundation models for ML4DB tasks (*5 mins*)
(4) Generating training data of high quality (*5 mins*)

## 3 TUTORIAL OUTLINE

We start with a brief overview of this tutorial, to give the audience a clear outline and talk goals.

### 3.1 ML4DB Foundations

In this section of the tutorial, we discuss two aspects of ML4DB foundations: query plan representation and pretrained model.

**1) Query plan representation.**

A significant number of ML4DB systems focus their research on understanding and optimizing query execution characteristics. Intuitively, these systems delve into how databases process and execute queries, seeking to enhance performance. Hence and unsurprisingly, query plans have been used as the input to these systems, including cost estimation [14, 29, 38], index recommendation [5], join order selection [30, 52], view generation [53], and query optimization [27, 28, 31, 56]. Despite targeting on different tasks, these models share a common reliance on the representations of query plans. Specifically, they seek to learn the correlations between the

properties of the query plans and the targeted outputs that is associated to the tasks. Therefore, we identify query plan representation as a foundation to the success of these ML4DB systems.

A physical query plan describes an execution pipeline for generating outputs from a query. A query plan is structured as a directed tree, in which each node describes a unit operation, and each edge describes the dependencies between two nodes, where nodes represent operations and edges denote the flow of data. More concretely, each node is responsible for processing a part of the query passing the results to its parent node [1].

To illustrate, consider E2E-Cost [38], a system designed to estimate the cardinality and cost associated with a query plan. It encodes a query plan in two stages: first, it extracts and encodes relevant information from each node into vector form. Subsequently, a custom TreeLSTM model [39] is employed to aggregate these node encodings in a bottom-up fashion recursively. The final state vector obtained from the root node serves as the representation of the query plan. This representation is then utilized by estimation models, such as multi-layer perceptrons (MLPs), to perform the estimation tasks [10].

Formally, we abstract and define the query plan representation as follows: consider a ML4DB system, query plan representation is the procedure to convert an arbitrary query plan into a vector representation, which is subsequently the input to a machine learning model of the ML4DB system. The vector has to encompass the important and discriminative features, so that the machine learning model can leverage the vector representation to enhance a database management system or provide insights to the database user.

**Table 1: Summary of Query Plan Representation Methods in ML4DB studies.**

| Method | Application | Tree Model |
|---|---|---|
| AVGDL [53] | View Selection | LSTM |
| AIMeetsAI [5] | Index Selection | Feature Vector |
| ReJOIN [30] | Join Order Selection | Feature Vector |
| BAO [27] | Optimizer | TreeCNN |
| NEO [28] | Optimizer | TreeCNN |
| Prestroid [14] | Cost Estimation | TreeCNN |
| E2E-Cost [38] | Cost/Card Estimation | TreeLSTM |
| RTOS [52] | Join Order Selection | TreeLSTM |
| Plan-Cost [29] | Cost Estimation | TreeRNN |
| QueryFormer [56] | General Purpose | Transformer |

We summarize the query plan representation methods in existing studies in Table 1. Evidently, there is no consensus on the choice of the model architectures to encode query plans. For example, the choice of tree models is not tied to a specific ML4DB application.

We model the query plan representation problem in two stages: **feature encoding**, and **tree model**, which is a common pipeline in existing methods. We summarize and discuss the strategies in both stages.

**Feature Encoding.** A query plan consists of information of heterogeneous types and correlated properties. Hence, a necessary step in all query plan representation methods is to select and encode some important features. We summarize the features and categorize them to two types: semantic features and database statistics. Semantic features refer to those describing the workloads of a query plan node, such as operator, table, predicate, etc. Database statistics refer to the extra information that is inferred from the meta data, such as histograms, samples, and cardinality or cost estimates generated by a database [57]. A common practice is to select a subset of features for every node and concatenate the representation vector of each component, depending on their task requirements [56].

**Tree Models.** A query plan takes the form of a directed tree, and thus forbids simple linear models to aggregate the node encodings. To transform the tree structures of query plans into a single fixed-size vector, existing studies use different strategies to model the tree structure. We summarize the strategies below:

- **LSTM.** LSTM is a recurrent neural network (RNN) with shortcut connections [12]. As there is no natural order to traverse a query plan with the tree structure, depth first search (DFS) is employed to first flatten a query plan [53]. The hidden state vector from the last node is treated as the query plan representation.
- **TreeCNN.** TreeCNN is a variant of the traditional convolutional neural networks (CNN), designed to handle tree-structured data [32]. It consists of convolutional layers with triangular shapes (parent-child-child) to slide over all the sub-trees [28].
- **TreeRNN.** TreeRNN models maintain the tree structure when aggregating node information. Some work use TreeLSTM, which generalizes the traditional LSTM cell by accepting inputs from multiple channels [38, 52]. Other work proposes novel RNN-units to aggregate features with a tree structure [29].
- **Transformer.** Transformer utilize a stack of attention layers to aggregate information from a sequence [42]. QueryFormer generalizes to the tree structure of a query plan by modifying the positional encoding and attention formulation [56].
- **Feature Vector.** We name the methods that do not use learnable parameters 'Feature Vector'. They directly encode the important features into a pre-defined size vector. To incorporate tree structures with varied shape and size, zero-padding is applied [5, 30].

To understand the behavior of the representation learning designs, a recent evaluation study provides a comparative evaluation of these methods and their components across different ML4DB tasks [57]. This analysis isolates the query plan representation components from existing ML4DB researches and interchanging them across different tasks. The findings [57] show that the optimal selection of feature encoding and tree model hinges on the optimization goal: whether the focus is on absolute prediction accuracy or relative performances. Further, it shows an interesting result where the choices of feature encoding are often more important than tree model designs and choices, although the existing studies typically focus on the latter.

**2) Pretrained Model.** One significant challenge for ML4DB is the tremendous cost of collecting training data, which typically involves collecting tens of thousands of execution traces when deploying to a new database system with a new dataset. This can be both time and resource consuming for practitioners. Motivated by the success of pretraining in large language models, several research studies explore the possibility of a pre-trained model that

can be applied to different tasks. This line of research alleviates the cost of data collection for each individual tasks, and therefore we identify it as a foundation for ML4DB systems.

Hilprecht and Binnig [11] envisioned zero-shot learning is the solution to practical ML4DB systems, as a learned model can be used to new databases out of the box without the need of retraining. They propose a prototype design, which disentangle the general features from the database-specific features, and inject database features from external cardinality estimates. Paul et al [35] proposes a pretrained model for query plans. Different from the studies discussed in query plan representation, it proposes an purely unsupervised pretraining strategy, which consumes datasets from multiple domains with different characteristics. It demonstrates capability of fine-tuning to new datasets or an arbitrary task quickly. MTMLF (short for Multi-task Meta-learning Framework) [46] proposes a more ambitious system, which leverages meta-learning techniques to capture transferable knowledge across various DBMS tasks. MTMLF splits the ML-required features to four quadrants: database-specific, database-agnostic, task-specific, and task-agnostic knowledge, and designs a specialized model for each component. It proposes a meta-learning framework to learn each component from multiple data sources simultaneously. As a result, it can leverage transferable knowledge beyond single database tasks.

Although these studies present the significant potential of a pretrained model across tasks, it is worth noting that these systems are still in their early stages with preliminary prototypes and results. The optimal design and strategy remain inconclusive, leaving room for further research and development.

Take-away: attendees become familiar with the common foundations for various aspects of ML4DB researches. They will understand how the studies of foundation models improve the performance and practicality of the ML4DB systems.

## 3.2 ML4DB paradigms: Replacement vs. ML-enhanced

In this section of the tutorial, we discuss the two paradigms of ML4DB research: "Replacement" vs. "ML-enhanced". Earlier methods in ML4DB typically adopt the "replacement" approach, *i.e.*, directly replacing database components with machine learning models. Typical examples of learned database components that replace existing database components include learned indexes like recursive model index (RMI) [17], recursive spatial model index (RSMI) [36], LISA [25] and learned query optimizers like DQ [18], Rejoin [29], NEO [28] and RTOS [52]. While the learned database components with the "replacement" approach proved to outperform traditional database components in some cases, they have two major limitation: **1) Poor robustness** and **2) Limited generalization capability**. ML-enhanced methods [2, 4, 7, 9, 24, 27, 33, 48, 54] address these limitation by using machine learning models to aid traditional database component rather than replacing them. We discuss ML-enhanced methods in the context of two important database component: database index and query optimizer. We first discuss the ML4DB work with the "replacement" approach and analyze their limitation in terms of robustness and generalization capability. We then discuss the ML-enhanced methods and how they address the limitations. We conclude by analyzing the key advantages of ML-enhanced methods.

**Machine Learning for Database Index.** We will start with an overview of learned index and learned multi-dimensional/spatial index. The initial idea of learned index [17] was to replace 1-d index structures like B-tree with a machine learning model that learns the cumulative density function (CDF) of the data. The model then maps the search key to the storage id based on the learned CDF. Several variants [6, 8, 16, 26] have been proposed to achieve better efficiency and robustness. Learned multi-dimensional/spatial indexes extended the idea to multi-dimensional/spatial data. ZM index [43] was proposed for 2-d spatial data points. It linearizes the spatial points using a space-filling curve, *e.g.*, Z-curve, and learns to model the CDF of the z-order values. RSMI [36] improved over ZM index by introducing a rank space-based ordering and a recursive partitioning strategy. Instead of relying on space-filling curves, LISA [25] was proposed to directly learn a mapping from spatial data points to 1-d value.

Learned index suffer from poor robustness and limited generalization capability. From the robustness perspective, due to the approximate error of the machine learning models, the predicted location of the records may deviate from the actual location. Therefore, learned index typically involve auxiliary structure or verification step to ensure the correctness of the result, *e.g.*, replacing the bottom level models with B-Tree [17], or using exponential search [6]. Moreover, learned spatial indexes may produce approximate results for certain types of query, *e.g.*, KNN query [36, 43]. From the generalization perspective, existing learned spatial indexes only support point data, and cannot handle data with geometry.

ML-enhance indexes applies machine learning models to database index in a different manner. ML-enhanced indexes typically keep the basic structures of the traditional database indexes, while use machine learning technique to aid certain index operations, *e.g.*, insertion, bulk-loading, and query. We will discuss ML-enhanced methods that aid three types of index operations: insertion, bulk-loading and query.

- **Insertion**: RLR-tree [9] proposed to build a better R-tree by improve R-tree insertion operation with reinforcement learning. Specifically, they use reinforcement learning based techniques to improve the heuristic rules used in the *chooseSubtree* function, *i.e.*, how to choose a subtree for insertion, and the *splitNode* function, *i.e.* how to split a fully-inserted node. RW-tree [7] proposed to improve the *chooseSubtree* and *splitNode* function in a workload-aware manner, by optimizing the two functions for a historical workload using a learned cost model. Li et al [24] proposed a new linearization method for multi-dimensional data that adopt different mapping schemes for different data subspaces.
- **Bulk-loading**: PLATON [48] is a top-down R-tree packing method with learned partition policy that explicitly optimizes the query performance with regard to the given data and workload instance. PLATON leverages a light-weight reinforcement learning technique, Monte Carlo Tree Search , to learn the partition policy and (MCTS) propose optimization techniques to drastically reduce the MCTS algorithm's time complexity and make it a linear-time algorithm.
- **Query**: AI+R tree [2] combines the R-tree with a machine learning based AI-tree to enhance the query performance.

The AI-tree improves high-overlap range queries by transforming the search operation of an R-tree into a multi-label classification task and skipping the extraneous leaf node accesses. The AI+R process high-overlap queries using the AI-tree and the low-overlap queries using the R-tree to achieve balanced performance across both types of queries.

*Take-away:* attendees become familiar with ML-enhanced methods in the context of database index. They will understand the initial learned index and its variants, the limitations of the learned indexes with the "replacement" approach and how ML-enhanced methods address these limitations.

**Machine Learning for Query Optimizer.** We will start by introducing the basic concepts of learned query optimizer. NEO [17] is the first end-to-end learned query optimizer that produces complete execution plans. NEO bootstraps from existing query optimizers, *e.g.*, PostgreSQL query optimizer, and train a value network that predicts the best expected total query run time given any partial query plan. The value network is trained from query latency signals under a reinforcement learning framework. Subsequent work RTOS [52] proposed to improve the join order selection process by levering Tree-LSTM for query plan representation. Tree-LSTM based plan representation captures the structural information of a join tree and handles schema change & multi-alias table names. In addition, RTOS improves training efficiency by using both query plan cost estimation and query latency as feedback.

Learned query optimizers suffer from poor robustness and limited generalization capability. From the robustness perspective, learned query optimizers suffer from the cold start problem and the performance of the model largely depends on the amount of diverse training query plans. It is costly to gather training data from real execution traces. When learned query optimizers are trained from limited amount of data, the model performance may drastically degrade on unseen queries. Although some learned query optimizers propose to increase the diversity of training query plan by using estimated cost signals [52] or simulation-to-reality learning [51], the query optimizer performance may still fluctuate, which is not desirable for a "safety-critical" system. From the generalization perspective, existing learned query optimizers can only handle select-project-join (SPJ) queries, but fail to handle more complicated queries such as queries involving subqueries.

We will discuss how ML-enhanced query optimizers apply machine learning to improve query optimizers in a completely different manner. We will focus our discuss on three ML-enhanced query optimizers, Bandit optimizer [15, 27, 33, 54], ML-aided query optimizer LEON [4], and ParamTree [50].

- **Bandit Optimizer:** BAO [27] was proposed to reduce the training overhead of learned query optimizers, adapt to the changes in data & workload, and improve tail performance. BAO use machine learning techniques to generate per-query optimization hints to improve the performance of an existing query optimizer. BAO formulate the hint-set selection problem as a contextual multi-armed bandit, and uses Thompson Sampling, a well-studied RL algorithm to solve the bandit. Due to its practicality and robustness, BAO was deployed to steer production databases at Microsoft [33, 54]. We will further introduce recent work on hint-based query optimizer that improves over BAO, *e.g.*, AutoSteer [3]. While BAO effectively learns to select the hint-set for queries, it requires a hand-crafted collection of right hint-sets. A new collection of hint-set needs to be hand-crafted in order to apply BAO to a new database system, which can incur significant integration cost. AutoSteer uses a greedy algorithm to systematically explore promising hint-sets and generates the hint-set collection dynamically on a per-query basis.
- **LEON:** LEON [4] propose a new framework for ML-aided query optimization. LEON uses ML to help expert query optimizers self-adjust to a certain dataset or workload, while allowing the ML-aided query optimizer to fall back to the expert query optimizer if the ML models fails. Specifically, LEON leverages a mixed cost estimation model that combines the expert cost model and a ML model trained with a pairwise ranking objective. LEON propose a robust plan exploration strategy to balance exploration and exploitation during plan search and uses a ML model-guided pruning strategy to improve planning efficiency.
- **ParamTree:** ParamTree [50] argues that if the hyperparameters of the formula-based cost models in databases like PostgreSQL are properly tuned, formula-based cost models can outperform learned cost models and learned query optimizers. ParamTree identifies the set ofhyperparamters affecting cost model performance, *i.e.,* **R-param**, and proposed a two-stage learning framework to learn **R-param** under both static and dynamic configuration environment. ParamTree can be integrated with existing query optimizers and exhibits good explanability and adaptability.

*Take-away:* attendees become familiar with ML-enhanced methods in the context of database query optimizer. They will understand how reinforcement learning based query optimizers work, the limitation associated with these early learned query optimizers, and how ML-enhanced methods address these limitations.

## 3.3 Open Problems

In this section, we discuss the key open problems that need to be solved before a mainstream adoption of ML4DB and some of the initial attempts at solving these problems.

**(1) Model Efficiency.** Model efficiency is one of the most important concern when integrating with performance-critical systems like databases. An ideal model should be efficiency in terms of both the space consumption and the training & inference speed. There have been some initial attempts at improving model efficiency for cardinality estimation and query optimization.

- **Cardinality Estimation.** Early work on machine learning for cardinality estimation used sophisticated models to achieve high model accuracy, which sacrifices model efficiency. Zhao et al [55] propose to apply a Bayesian deep learning technique, neural network gaussian process, to cardinality estimation, which enables model training in a few seconds.
- **Query Optimization.** Learned query optimizer like NEO [28] rely on plans generated by an existing expert optimizer and real execution traces for training, which can be highly inefficient if expensive plans are generated. Balsa [51] propose to

learn without an existing expert optimizer. Balsa employs simulation-to-reality learning, where it trains the model from simulation based on PostgreSQL's cardinality estimator to avoid disastrous plans and achieve fast convergence. When fine-tuning the model from real execution experience, Balsa proposes a safe execution framework with timeout to eliminate unpredictable stalls.

**(2) Handling data & workload shifts.** Databases operate under dynamic environments with constant changes in the data and the workload. However, most machine learning models require retraining in face of data & workload shifts, which can be very costly. Therefore, it is crucial to study efficient model update techniques under data & workload shifts. Some recent work have proposed machine learning based cardinality estimation [20, 34, 44] and data generation [19] techniques that adapts to data & workload drifts.

**(3) Foundation models for ML4DB tasks.** Current foundation models such as query plan representation models excel in single task and dataset setting, where the effectiveness latch onto the task and dataset-specific patterns. This severely limits the practicality of ML4DB systems. Self-supervised learning techniques can be incorporated to train the representation model from diverse training data and capture general features, so that it can adapt to different datasets. Techniques such as meta-learning can be explored so that the trained model can generalize across tasks.

**(4) Generating training data of high quality.** Training data collection in ML4DB research is extremely costly, as it usually involves executing query workloads. Therefore, it calls for a need for generating high-quality training data. This challenge is not just about volume but also the quality and diversity of data. High-quality datasets must accurately reflect real-world scenarios, addressing biases and ensuring completeness and representativeness [49]. Furthermore, privacy and availability further complicate this issue, where the database administrator does not have full access to the real customer data. As a result, developing methodologies for creating, augmenting, or simulating realistic and privacy-compliant datasets at low cost remain as an open problem.

## 4 RELATED TUTORIALS

In recent years, several tutorials introduce the application of machine learning on different components of database systems and these tutorials are organized based on database problems. In 2021, Li et al. [23] introduce the use of machine learning to solve three types of database management problems: NP-hard problems, regression problems and prediction problems. The authors review the studies on both AI4DB and DB4AI in another tutorial in the same year [22]. These tutorials outlines the techniques in each of the categories in database configuration, optimization and database design. Further, it elaborates on methods to improve the practicality of AI applications using database, spanning model inference, model training, and data governance aspects. Next, Li and Zhou [21] classify ML4DB into two types of applications: database core, database advisor, and training data generation. The former part covers using ML methods to address the NP-hard problems in database optimization, while the latter covers autonomous services like trend prediction, knob tuning, etc. Jindal and Interlandi [13] introduces the ML4DB techniques in the context of cloud system applications. It breaks down the cloud data system to four different layers: platforms, query engines, services, developers, respectively, and covers comprehensively the usefulness of ML methods for each aspect at each layer. It also envisions open problems in cloud applications. Wasay et al [45] covers several topics in the interaction of ML techniques and data systems. In addition to the introduction of ML applications in database tasks, it discusses the accuracy-time tradeoff of deep learning models from system perspective. It further covers responsible deep learning topics from three dimensions: fairness, interpretability, and environmental impact. All these tutorials give bird views of the application of machine learning to different database components from different perspectives.

Other relevent tutorials focus on a specific ML4DB problem or application. Trummer [40] introduces pretrained large language models and the potential of their applications to data management problems. It demonstrates that large language models like GPT can be used to generate instructions to tune and optimize a data system. Tsesmelis and Simitsis [41] introduces the various ML4DB aspects that center around query optimization, including learned query optimizer, cost estimators, runtime prediction, and the benchmarks. Yan et al [47] talk about the join order selection problem, the traditional techniques as well as the advances in deep reinforcement learning approaches, with open-source demonstrations.

However, none of these tutorials looks into the foundations and the paradigms in the development of ML4DB in a systematic manner as we plan to do in this tutorial. The proposed tutorial was not presented in other venues.

## 5 BIOGRAPHIES

All the presenters have jointly worked on machine learning on data management problems and published their research results in main database venues, such as SIGMOD, VLDB, ICDE and EDBT.

**Gao Cong** is a Professor in the School of Computer Science and Engineering at Nanyang Technological University (NTU), and a co-director for Singtel Cognitive and Artificial Intelligence Lab for Enterprises@NTU. He previously worked at Aalborg University, Denmark, Microsoft Research Asia, and the University of Edinburgh. His current research interests include ML4DB, spatial data management, spatial-temporal data mining, and recommendation systems. He received the best paper runner-up awards at the WSDM'20 and WSDM'22 conferences for two of his research papers. His citation in Google Scholar was over 18,000 with H-index 71.

**Jingyi Yang** is a PhD candidate in the Interdisplinary Graduate Programme at Nanyang Technological University (NTU) and a part-time tutor at NTU. His research focuses on ML4DB and spatial databases. His work appears at leading data management venues such as SIGMOD and VLDB. He obtained his Bachelor's degree in Computer Science (Honor Highest Distinction) from Nanyang Technological University in 2021.

**Yue Zhao** is a PhD candidate at Nanyang Technological University, Singapore, and a part-time tutor at NTU. His research focuses on representation learning in ML4DB. He publishes papers at leading data management venues such as VLDB. He obtained his Bachelor's degree in Electrical Engineering with First Class Honor and several Dean's List Awards from National University of Singapore in 2019.

# REFERENCES


[1] 2021. *Documentation PostgreSQL 12, Explain.* https://www.postgresql.org/docs/12/sql-explain.html

[2] Abdullah-Al Abdullah-Al-Mamun, Ch. Md. Rakin Haider, Jianguo Wang, and Walid G. Aref. 2022. The "AI + R" - tree: An Instance-optimized R - tree. In *2022 23rd IEEE International Conference on Mobile Data Management (MDM)*. 9–18. https://doi.org/10.1109/MDM55031.2022.00023

[3] Christoph Anneser, Nesime Tatbul, David Cohen, Zhenggang Xu, Prithviraj Pandian, Nikolay Laptev, and Ryan Marcus. 2023. AutoSteer: Learned Query Optimization for Any SQL Database. *Proceedings of the VLDB Endowment* 16, 12 (2023), 3515–3527.

[4] 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. *Proceedings of the VLDB Endowment* 16, 9 (2023), 2261–2273.

[5] Bailu Ding, Sudipto Das, Ryan Marcus, Wentao Wu, Surajit Chaudhuri, and Vivek R Narasayya. 2019. Ai meets ai: Leveraging query executions to improve index recommendations. In *Proceedings of the 2019 International Conference on Management of Data*. 1241–1258.

[6] Jialin Ding, Umar Farooq Minhas, Jia Yu, Chi Wang, Jaeyoung Do, Yinan Li, Hantian Zhang, Badrish Chandramouli, Johannes Gehrke, Donald Kossmann, et al. 2020. ALEX: an updatable adaptive learned index. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 969–984.

[7] Haowen Dong, Chengliang Chai, Yuyu Luo, Jiabin Liu, Jianhua Feng, and Chaoqun Zhan. 2022. RW-Tree: A Learned Workload-aware Framework for R-tree Construction. In *2022 IEEE 38th International Conference on Data Engineering*. IEEE.

[8] Paolo Ferragina and Giorgio Vinciguerra. 2020. The PGM-index: a fully-dynamic compressed learned index with provable worst-case bounds. *Proceedings of the VLDB Endowment* 13, 8 (2020), 1162–1175.

[9] Tu Gu, Kaiyu Feng, Gao Cong, Cheng Long, Zheng Wang, and Sheng Wang. 2023. The RLR-Tree: A Reinforcement Learning Based R-Tree for Spatial Data. *Proceedings of the ACM on Management of Data* 1, 1 (2023), 1–26.

[10] Simon Haykin. 1994. *Neural networks: a comprehensive foundation*. Prentice Hall PTR.

[11] Benjamin Hilprecht and Carsten Binnig. 2021. One Model to Rule them All: Towards Zero-Shot Learning for Databases. *CoRR* abs/2105.00642 (2021). arXiv:2105.00642 https://arxiv.org/abs/2105.00642

[12] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[13] Alekh Jindal and Matteo Interlandi. 2021. Machine Learning for Cloud Data Systems: The Progress so Far and the Path Forward. *Proc. VLDB Endow.* 14, 12 (2021), 3202–3205.

[14] Johan Kok Zhi Kang, Gaurav, Sien Yi Tan, Feng Cheng, Shixuan Sun, and Bingsheng He. 2021. Efficient Deep Learning Pipelines for Accurate Cost Estimations Over Large Scale Query Workload. In *SIGMOD*. ACM, 1014–1022.

[15] Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *ICLR*.

[16] Andreas Kipf, Ryan Marcus, Alexander van Renen, Mihail Stoian, Alfons Kemper, Tim Kraska, and Thomas Neumann. 2020. RadixSpline: a single-pass learned index. In *Proceedings of the Third International Workshop on Exploiting Artificial Intelligence Techniques for Data Management*. 1–5.

[17] Tim Kraska, Alex Beutel, Ed H Chi, Jeffrey Dean, and Neoklis Polyzotis. 2018. The case for learned index structures. In *Proceedings of the 2018 international conference on management of data*. 489–504.

[18] Sanjay Krishnan, Zongheng Yang, Ken Goldberg, Joseph Hellerstein, and Ion Stoica. 2018. Learning to optimize join queries with deep reinforcement learning. *arXiv preprint arXiv:1808.03196* (2018).

[19] Meghdad Kurmanji and Peter Triantafillou. 2023. Detect, Distill and Update: Learned DB Systems Facing Out of Distribution Data. *Proceedings of the ACM on Management of Data* 1, 1 (2023), 1–27.

[20] Beibin Li, Yao Lu, and Srikanth Kandula. 2022. Warper: Efficiently adapting learned cardinality estimators to data and workload drifts. In *Proceedings of the 2022 International Conference on Management of Data*. 1920–1933.

[21] Guoliang Li and Xuanhe Zhou. 2022. Machine Learning for Data Management: A System View. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. 3198–3201.

[22] Guoliang Li, Xuanhe Zhou, and Lei Cao. 2021. AI meets database: AI4DB and DB4AI. In *Proceedings of the 2021 International Conference on Management of Data*. 2859–2866.

[23] Guoliang Li, Xuanhe Zhou, and Lei Cao. 2021. Machine learning for databases. In *Proceedings of the First International Conference on AI-ML Systems*. 1–2.

[24] Jiangneng Li, Zheng Wang, Gao Cong, Cheng Long, Han Mao Kiah, and Bin Cui. 2023. Towards Designing and Learning Piecewise Space-Filling Curves. *Proceedings of the VLDB Endowment* 16, 9 (2023), 2158–2171.

[25] Pengfei Li, Hua Lu, Qian Zheng, Long Yang, and Gang Pan. 2020. LISA: A learned index structure for spatial data. In *Proceedings of the 2020 ACM SIGMOD international conference on management of data*. 2119–2133.

[26] Baotong Lu, Jialin Ding, Eric Lo, Umar Farooq Minhas, and Tianzheng Wang. 2021. APEX: A High-Performance Learned Index on Persistent Memory. *arXiv preprint arXiv:2105.00683* (2021).

[27] Ryan Marcus, Parimarjan Negi, Hongzi Mao, Nesime Tatbul, Mohammad Alizadeh, and Tim Kraska. 2021. Bao: Making learned query optimization practical. In *Proceedings of the 2021 International Conference on Management of Data*. 1275–1288.

[28] Ryan Marcus, Parimarjan Negi, Hongzi Mao, Chi Zhang, Mohammad Alizadeh, Tim Kraska, Olga Papaemmanouil, and Nesime Tatbul. 2019. Neo: A learned query optimizer. In *VLDB*.

[29] Ryan Marcus and Olga Papaemmanouil. 2018. Deep reinforcement learning for join order enumeration. In *Proceedings of the First International Workshop on Exploiting Artificial Intelligence Techniques for Data Management*. 1–4.

[30] Ryan Marcus and Olga Papaemmanouil. 2018. Deep Reinforcement Learning for Join Order Enumeration. In *aiDM@SIGMOD 2018*. ACM, 3:1–3:4.

[31] Songsong Mo, Yile Chen, Hao Wang, Gao Cong, and Zhifeng Bao. 2023. Lemo: A Cache-Enhanced Learned Optimizer for Concurrent Queries. *Proceedings of the ACM on Management of Data* 1, 4 (2023), 1–26.

[32] Lili Mou, Ge Li, Lu Zhang, Tao Wang, and Zhi Jin. 2016. Convolutional Neural Networks over Tree Structures for Programming Language Processing. In *AAAI, 2016*. AAAI Press, 1287–1293.

[33] Parimarjan Negi, Matteo Interlandi, Ryan Marcus, Mohammad Alizadeh, Tim Kraska, Marc Friedman, and Alekh Jindal. 2021. Steering query optimizers: A practical take on big data workloads. In *Proceedings of the 2021 International Conference on Management of Data*. 2557–2569.

[34] Parimarjan Negi, Ziniu Wu, Andreas Kipf, Nesime Tatbul, Ryan Marcus, Sam Madden, Tim Kraska, and Mohammad Alizadeh. 2023. Robust Query Driven Cardinality Estimation under Changing Workloads. *Proceedings of the VLDB Endowment* 16, 6 (2023), 1520–1533.

[35] Debjyoti Paul, Jie Cao, Feifei Li, and Vivek Srikumar. 2021. Database Workload Characterization with Query Plan Encoders. *Proc. VLDB Endow.* 15, 4 (2021), 923–935.

[36] Jianzhong Qi, Guanli Liu, Christian S Jensen, and Lars Kulik. 2020. Effectively learning spatial indices. *Proceedings of the VLDB Endowment* 13, 12 (2020), 2341–2354.

[37] Jiachen Shi, Gao Cong, and Xiaoli Li. 2022. Learned Index Benefits: Machine Learning Based Index Performance Estimation. *Proc. VLDB Endow.* 15, 13 (2022), 3950–3962.

[38] Ji Sun and Guoliang Li. 2019. An end-to-end learning-based cost estimator. *VLDB* 13, 3 (2019), 307–319.

[39] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In *Proc. ACL 2015*. 1556–1566.

[40] Immanuel Trummer. 2022. From BERT to GPT-3 Codex: Harnessing the Potential of Very Large Language Models for Data Management. *Proc. VLDB Endow.* 15, 12 (2022), 3770–3773.

[41] Dimitris Tsesmelis and Alkis Simitsis. 2022. Database Optimizers in the Era of Learning. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. 3213–3216.

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *arXiv:1706.03762* (2017).

[43] Haixin Wang, Xiaoyi Fu, Jianliang Xu, and Hua Lu. 2019. Learned index for spatial queries. In *2019 20th IEEE International Conference on Mobile Data Management (MDM)*. IEEE, 569–574.

[44] Zilong Wang, Qixiong Zeng, Ning Wang, Haowen Lu, and Yue Zhang. 2023. CEDA: Learned Cardinality Estimation with Domain Adaptation. *Proceedings of the VLDB Endowment* 16, 12 (2023), 3934–3937.

[45] Abdul Wasay, Subarna Chatterjee, and Stratos Idreos. 2021. Deep Learning: Systems and Responsibility. In *Proceedings of the 2021 International Conference on Management of Data*. Association for Computing Machinery, 2867–2875.

[46] Ziniu Wu, Pei Yu, Peilun Yang, Rong Zhu, Yuxing Han, Yaliang Li, Defu Lian, Kai Zeng, and Jingren Zhou. [n. d.]. A Unified Transferable Model for ML-Enhanced DBMS. In *12th Conference on Innovative Data Systems Research, CIDR 2022, Chaminade, CA, USA, January 9-12, 2022*.

[47] Zhengtong Yan, Valter Johan Edvard Uotila, and Jiaheng Lu. 2023. Join Order Selection with Deep Reinforcement Learning: Fundamentals, Techniques, and Challenges. *Proceedings of the VLDB Endowment* 16, 12 (2023), 3882–3885.

[48] Jingyi Yang and Gao Cong. 2023. PLATON: Top-down R-tree Packing with Learned Partition Policy. *Proceedings of the ACM on Management of Data* 1, 4 (2023), 1–26.

[49] Jingyi Yang, Peizhi Wu, Gao Cong, Tieying Zhang, and Xiao He. 2022. SAM: Database Generation from Query Workloads with Supervised Autoregressive Models. In *Proceedings of the 2022 International Conference on Management of Data*. 1542–1555.

[50] Jiani Yang, Sai Wu, Dongxiang Zhang, Jian Dai, Feifei Li, and Gang Chen. 2023. Rethinking Learned Cost Models: Why Start from Scratch? *Proceedings of the ACM on Management of Data* 1, 4 (2023), 1–27.

[51] Zongheng Yang, Wei-Lin Chiang, Sifei Luan, Gautam Mittal, Michael Luo, and Ion Stoica. 2022. Balsa: Learning a Query Optimizer Without Expert Demonstrations. In *Proceedings of the 2022 International Conference on Management of Data*. 931–944.

[52] Xiang Yu, Guoliang Li, Chengliang Chai, and Nan Tang. 2020. Reinforcement learning with tree-lstm for join order selection. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 1297–1308.

[53] Haitao Yuan, Guoliang Li, Ling Feng, Ji Sun, and Yue Han. 2020. Automatic View Generation with Deep Learning and Reinforcement Learning. In *ICDE 2020*. IEEE, 1501–1512.

[54] Wangda Zhang, Matteo Interlandi, Paul Mineiro, Shi Qiao, Nasim Ghazanfari, Karlen Lie, Marc Friedman, Rafah Hosn, Hiren Patel, and Alekh Jindal. 2022. Deploying a steered query optimizer in production at Microsoft. In *Proceedings of the 2022 International Conference on Management of Data*. 2299–2311.

[55] Kangfei Zhao, Jeffrey Xu Yu, Zongyan He, Rui Li, and Hao Zhang. 2022. Lightweight and accurate cardinality estimation by neural network gaussian process. In *Proceedings of the 2022 International Conference on Management of Data*. 973–987.

[56] Yue Zhao, Gao Cong, Jiachen Shi, and Chunyan Miao. 2022. QueryFormer: a tree transformer model for query plan representation. *Proceedings of the VLDB Endowment* 15, 8 (2022), 1658–1670.

[57] Yue Zhao, Zhaodonghui Li, and Gao Cong. 2024. A Comparative Study and Component Analysis of Query Plan Representation Techniques in ML4DB Studies. *Proceedings of the VLDB Endowment* 17, 4 (2024).

[58] 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 *EDBT*. 1–4.