NeurDB: An AI-powered Autonomous Data System

Beng Chin Ooi[1], Shaofeng Cai[1], Gang Chen[2], Yanyan Shen[3], Kian-Lee Tan[1], Yuncheng Wu[4], Xiaokui Xiao[1], Naili Xing[1], Cong Yue[1], Lingze Zeng[1], Meihui Zhang[5] & Zhanhao Zhao[1*]

[1]*National University of Singapore, Singapore;*
[2]*Zhejiang University, China;*
[3]*Shanghai Jiao Tong University, China;*
[4]*Renmin University of China, China;*
[5]*Beijing Institute of Technology, China*

Abstract In the wake of rapid advancements in artificial intelligence (AI), we stand on the brink of a transformative leap in data systems. The imminent fusion of AI and DB (AI×DB) promises a new generation of data systems, which will relieve the burden on end-users across all industry sectors by featuring AI-enhanced functionalities, such as personalized and automated in-database AI-powered analytics, self-driving capabilities for improved system performance, etc. In this paper, we explore the evolution of data systems with a focus on deepening the fusion of AI and DB. We present NeurDB, our next-generation data system designed to fully embrace AI design in each major system component and provide in-database AI-powered analytics. We outline the conceptual and architectural overview of NeurDB, discuss its design choices and key components, and report its current development and future plan.

Keywords AI×DB, in-database AI, intelligent data system

1 Introduction

In tandem with digitization and digitalization initiatives across all industries, data systems have evolved to become the cornerstone of mainstream online applications integral to our everyday lives. Traditionally, data systems (aka databases or data management systems) have been designed to meet the transactional and analytical data processing requirements of these applications. However, with the rapid advancements in artificial intelligence (AI), we are witnessing a surge in AI-powered applications across sectors, which aim to relieve the burden on end-users by offering users personalized, automated AI-enhanced analytics and predictions. To support such emerging AI applications, we are now standing on the brink of a transformative leap in data systems, poised to realize a data system that genuinely aligns with the demand of end-users for intelligent and automatic data services.

At present, the realms of data systems and AI models exist as independent fields, each fundamental in its own right but isolated in functionality. Take the healthcare sector as an example: clinicians require AI-powered assistance for disease progression modeling, and their data scientists need to navigate the complexities of both AI and database systems to construct prediction tasks. This separation presents significant challenges in the development and operational efficiency of AI applications, as illustrated in Figure 1(a). The need for expertise in both domains to create and integrate these systems adds complexity to application development. Further, user requests often trigger extensive interactions between AI models and databases, whether for data retrieval, model training, or fine-tuning, thereby impacting application performance. These barriers hinder the user-friendliness and operational efficiency of applications, posing a major obstacle to the broader adoption of AI technologies.

Next-generation data systems are set to experience a transformative revolution with the imminent fusion of AI and DB (AI×DB) to dismantle the barriers preventing end-users from fully embracing AI

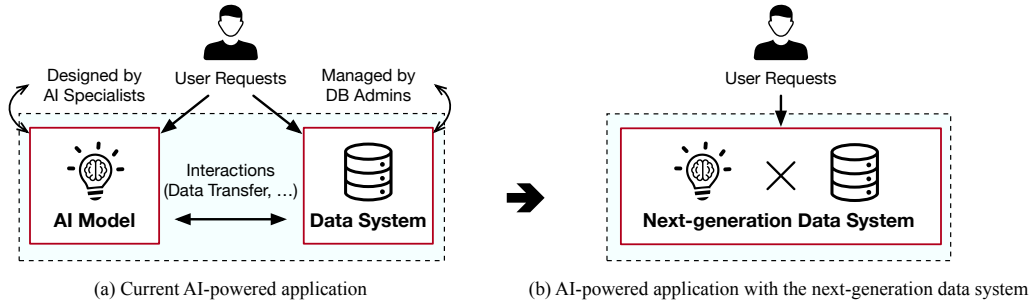---

* Corresponding author (email: zhanhao@nus.edu.sg)

Figure 1 Evolution to AI-powered applications with next-generation data systems

technologies. As we envisioned in 2016 [1], AI and databases can mutually enhance each other's capabilities. Yet years later, despite considerable progress in integrating AI with data systems, a significant gap remains between the potential of this integration and its current state of usability. As shown in Figure 1(b), bridging this gap requires a seamless integration of AI and database technologies, creating a unified data system that transcends the capabilities of each in isolation. The transition to this new technological paradigm is an ambitious journey. With an expected influx in the data volume, continuous shifts in data dynamics and workloads, and the emergence of increasingly more complex AI models and applications, we anticipate a substantial endeavor in the realization of next-generation data systems. Such evolution promises not only to upgrade system functionalities through in-database AI-powered analytics but also to spur the development of novel AI-driven data system architectures and methodologies.

In this position paper, we trace the evolutionary trajectory of data systems and characterize related works on AI×DB, establishing the timeliness and necessity of advancing toward next-generation data systems. We showcase promising AI-powered applications for which we believe the next-generation data system has much to offer. Through a comprehensive analysis of user needs derived from these applications, we identify key challenges in deepening the fusion of AI and DB to realize next-generation data systems. Firstly, enabling in-database AI-powered analytics to render data systems inherently AI-friendly proves to be intricate, particularly in accommodating modern AI applications like Large Language Models (LLMs) and generative AI. Secondly, the vision of fully autonomous, intelligent data systems is still a distant reality, as existing data systems necessitate extensive manual oversight and maintenance. Lastly, the fusion of AI and databases introduces new challenges in data privacy, which existing techniques often overlook.

We present NeurDB, our initiative towards creating a next-generation system that seamlessly supports AI applications and fully embraces AI techniques. We introduce methodologies and insights to address the above challenges, focusing on creating an in-database AI-friendly ecosystem, developing an intelligent self-driving data system, and providing privacy preservation and trusted AI×DB. For each objective, we delve into various new techniques employed in NeurDB, including in-database dynamic model selection and slicing, fast-adaptive learned database components, in-database federated learning, etc. We also outline future pathways for the development of NeurDB, highlighting the open research challenges that must be addressed along this journey.

The structure of this paper is organized as follows. In Section 2, we present the relevant background to drive the discussion on next-generation data systems. Section 3 defines key features and characteristics central to the next-generation data systems. We detail our contributions to NeurDB by introducing several key techniques in Section 4, Section 5, and Section 6. We discuss the development of NeurDB in Section 7 and conclude in Section 8.

2 Background

In this section, we describe the evolutionary trajectory of data systems, review the related work on AI×DB, and discuss emerging AI-powered applications in healthcare and e-commerce, respectively.

2.1 Evolutionary trajectory of data systems

The evolution of data systems can be segmented into several phases, each marked by innovations addressing the changing needs of data storage, management, and analysis. As shown in Figure 2, these phases
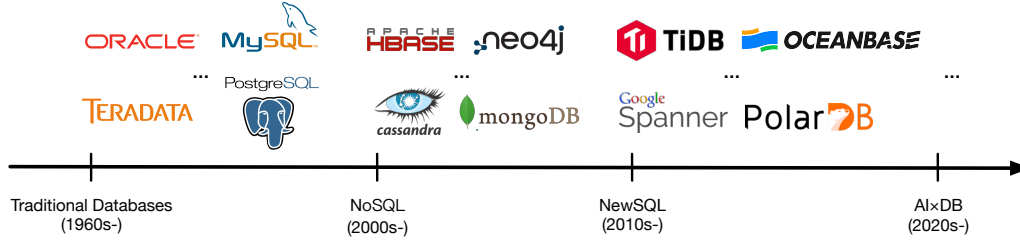
Figure 2 Evolutionary trajectory of data systems

include the era of databases, the movement of NoSQL, and the emergence of NewSQL [2].

Phase 1: Traditional databases. The origins of data systems can be traced back to the 1960s. It was the introduction of relational databases in the 1970s that propelled data systems into the commercial spotlight. Based on the relational model, relational databases excel at managing structured data, offering robust transaction support to ensure ACID (Atomicity, Consistency, Isolation, Durability) properties [3], and providing a powerful query language, namely SQL (Structured Query Language). Due to these strengths, many of today's applications continue to rely on traditional relational databases, such as Oracle, MySQL, etc. However, traditional databases typically adopt single-machine deployment, which restricts their scalability and flexibility.

Phase 2: NoSQL. As the volume, velocity, and variety of data began to expand, the limitations of traditional databases, particularly in terms of scalability, became apparent. These problems sparked the NoSQL movement in the mid to late 2000s. NoSQL systems were designed to offer scalable data processing, catering to the needs of big data and real-time online applications. They could handle unstructured and semi-structured data, supporting a variety of data models, e.g., key-value, document, wide-column, graph, etc. Despite their advantages, NoSQL systems often sacrifice transaction guarantees, leading to challenges in data consistency and integrity.

Phase 3: NewSQL. By the end of the 2000s, it was clear that many critical applications, such as e-banking and e-commerce, required both the scalability offered by NoSQL, as well as the ACID guarantees provided by relational databases. To address this need, NewSQL systems were designed to distribute data across multiple machines for enhanced scalability, while offering robust transaction support to ensure ACID properties. More recently, with the advancements in cloud computing, NewSQL has begun migrating to the cloud. This transition has given rise to many cloud-native databases that leverage the elasticity, resilience, and efficiency provided by the cloud.

Next-generation data systems. The emergence of AI-powered applications across various domains is catalyzing the evolution of data systems towards a deeper fusion with AI. Without this critical advancement, data systems will fail to offer automated and personalized data services demanded by modern AI models such as LLMs and generative AI.

2.2 Related work on AI×DB

The aspiration of integrating DB systems and AI technologies was first expressed some forty years ago [4]. With the advancement in both fields over the years and the increasing demand for "intelligent" data systems offering advanced analytics, the fusion of both technologies become more imminent. In 2016, the mutual benefits that deep learning and databases can offer each other have been elaborated [1], and more recently, various research works on AI×DB have been proposed.

For one thing, databases can significantly enhance AI applications. Databases have the potential to efficiently manage numerous AI models using their inherent user-friendly SQL interfaces, and provide high-quality data with their transaction guarantees to enhance AI model accuracy. For another, AI can substantially optimize databases. AI-driven techniques, such as knob tuning, can automatically refine database configurations to improve performance, which often surpasses traditional manual tuning done by database administrators (DBAs). However, existing approaches on AI×DB typically focus on optimizing within their specific domains, and have not yet embraced a full fusion of AI and DB into a unified data system. We now delve deeper into the existing works to highlight the integration challenges and opportunities.

1) *DB support for AI.* Database techniques under this category aim to improve data quality for more accurate model training, provide automatic model selection, and accelerate model inference. However,

existing approaches are often built on top of the database rather than fully integrated within it, which limits their effectiveness and efficiency. First, while various approaches, such as data cleaning, data integration, data discovery, data enrichment, and data lineage, are proposed to improve data quality, they are not directly supported by existing databases. Implementing these techniques often requires complex combinations of multiple SQL commands, which can be user-unfriendly and inefficient. Second, several systems [5] that are built on top of databases can automatically select suitable models for AI applications. However, these systems often need to retrieve and transfer data from the database, which can be costly and time-consuming due to the increasing volume of data. Third, recent proposals [6, 7] target to enhance model inference efficiency, however, they typically assume static models that can be buffered in memory, which is not always the case with today's dynamic, large-scale models.

2) *AI support for DB.* AI has shown potential for optimizing database systems. Previous works [8] introduce automatic knob tuning to improve system performance, but they provide limited insight into the possibilities of AI-powered database components, leaving broader opportunities for AI enhancements largely untapped. Despite several efforts to create learned database components, such as learned query optimizers for more accurate query plans and learned indexes for faster key searches, these solutions rely heavily on offline training on static datasets. This paradigm limits their effectiveness when dealing with the dynamic workloads common in modern applications, rendering them impractical for real-world use. Further, existing works often neglect the potential of combining AI with recent advancements in databases, such as the disaggregated architecture that separates computation from storage in cloud environments to improve scalability and elasticity.

2.3 Emerging AI-powered applications

AI-powered applications are increasingly becoming integral across various sectors, which aim to relieve the burden on end-users by offering personalized, automated AI-powered analytics and predictions. Here we outline the transformative impact of AI in healthcare and e-commerce domains.

2.3.1 *Intelligent healthcare*

Healthcare has continuously improved over the years, and with recent advancements in AI techniques, it is becoming smarter. We envision that the future of AI-powered healthcare promises exceptional intelligence, enabling more precise diagnoses, customized treatment plans, and proactive health management [9] AI agents will play a crucial role in assisting physicians in achieving more precise diagnoses. By leveraging AI models trained on extensive datasets, including genetic profiles, medical imaging, electronic health records, physicians' notes, etc., these agents can reveal patterns not detectable by the human eye in areas such as neurology and orthopedics. Further, AI-powered virtual assistants will offer 24/7 support to patients. They will handle inquiries about medication schedules, offer advice on managing chronic conditions, provide insights into physiological changes, and ensure adherence to treatment plans.

2.3.2 *Intelligent e-commerce*

AI is revolutionizing e-commerce with automated fraud detection, personalized recommendations, and intelligent resource management tailored to workload demands. By learning from transactional logs, AI models can uncover and predict anomalies to achieve accurate and real-time fraud detection, protecting businesses and customers from financial losses [10]. Further, AI provides more efficient and enjoyable online shopping by offering personalized recommendations based on customer browsing history, purchase records, and preferences. During high-traffic events such as Black Friday and Double Eleven (Singles' Day), AI-powered resource management is able to dynamically adjust the system to handle sudden spikes in demand, which guarantees the availability and efficiency of e-commerce systems.

3 Overview of **NeurDB**

In this section, we describe the key principles of designing **NeurDB** and present the conceptual and architectural overview of **NeurDB**.

3.1   Key design principles

With the dynamic nature of workloads, data systems must inherently adapt to changes in data storage, data structures, system configurations, etc. Traditionally, such dynamism is manually managed through actions like data insertion or update, index creation, and configuration tuning. However, with the fusion of AI and DB, next-generation data systems anticipate a shift towards AI-driven autonomy, characterized by self-assembling and self-optimizing capabilities. This shift entails that the entire system architecture can dynamically and autonomously adapt to varying demands under the guidance of AI. For example, under heavy AI-enhanced analytical workloads, the system's embedded AI agent is expected to autonomously allocate more GPU resources and assemble appropriate AI models tailored to the analytical tasks. In contrast, for standard transactional workloads that do not require AI-specific operations, the system might revert to a conventional, non-AI architecture. These observations lead to our definitions of three essential properties for the design of next-generation AI-powered autonomous data systems.

Dynamicity. Dynamicity refers to the property that a data system can evolve autonomously in response to changing workloads and environmental conditions. Dynamicity affects both the data managed by the system and the system architecture, including its components and operational behaviors. This property involves a controlled system evolution from one state to another, ensuring a linear progression through these states.

Reliability. Reliability is defined as the ability of a data system to consistently meet performance and accuracy standards, even during adaptation and evolution. That is, the data system is able to perform operations with optimal accuracy and consistency across all states. However, reliability often requires a static or controlled environment to prevent unexpected state transitions that may compromise this property.

Scalability. Scalability refers to the system's ability to effectively increase capacity by either scaling up (enhancing existing infrastructure) or scaling out (adding more nodes/resources to the system). This property guarantees that as the workload grows, the system can scale and/or reconfigure to maintain or improve performance, while achieving strategic resource allocation and utilization.

Data systems are dynamic in nature as the data distributions and query patterns evolve over time, and therefore, the systems have to adapt for the dynamicity while guaranteeing reliability.

FRP conjecture. For a dynamic and real-time data system, computational models or algorithms must be designed based on the Filter-and-Refine Principle, so that the system only considers the data subspace or strategies that potentially yield the near-optimal performance.

Due to the unpredictable nature of real-world environments and the inherent uncertainties in prediction accuracy and latency, the data systems must adapt and provide acceptable performance. The system optimization workflow often incurs several iterations. We are therefore using the Filter-and-Refine Principle (FRP) to guide the design of NeurDB. For a dynamic system with huge amounts of data, FRP is the basis for adaptivity, efficiency, and scalability. The filter-and-refine strategy is typically used to speed up computationally intensive tasks so that they can be completed within the pre-defined latency and/or resource constraints. In some tasks, such as a join, we have to remove data that are surely not contributing to the answer quickly, and then perform computationally intensive steps on a much smaller subset of the original data. Likewise, we eliminate query processing plans that are obviously suboptimal quickly, and only look for a near-optimal plan. For iterative computations, we should take actions that move to the target within a smaller search space.

3.2   Key features

Next-generation data systems are confronted with three critical challenges that must be addressed to efficiently and effectively support AI-powered applications.

3.2.1   *In-database AI-powered analytics*

AI-powered analytics are fundamental in constructing emerging AI applications, as discussed in Section 2.3. Unlike traditional data analytics that typically offers a limited statistical view based on filtering or aggregation operations [11–13], AI-powered analytics is more dynamic to capture and discover the complexity and intricacies of underlying data patterns [14]. However, existing approaches to AI-powered analytics [15, 16] fall short in efficiency and effectiveness, primarily due to the inherent limitations of using separate data systems and AI models. Critical operations required by AI models, such as data

preparation, model training, and making inferences, necessitate extensive interactions with data systems to retrieve the corresponding data. With the continuous emergence of new AI models, such as Deep Neural Networks (DNNs) [17,18] and LLMs, such interactions become increasingly costly due to the vast amounts of data these models require.

To address this, we assert that in-database, AI-powered analytics is crucial and should be a key feature of NeurDB. Consider a user named Tom browsing an e-commerce app. For a personalized recommendation, we aim to enable NeurDB to directly answer complex analytical queries such as "What are the top 10 recommended items for Tom?" by automatically selecting the appropriate AI model and delivering results without costly data transformation. Such capability would significantly enhance the user-friendliness, effectiveness, and efficiency of AI-powered analytics.

3.2.2 *Intelligent self-driving data system*

Enhancing database systems with AI techniques is becoming increasingly popular. For example, automatic knob tuning can optimize database configurations by selecting the best system parameters. Similarly, data structures like indexing can be improved with learning methods that adapt to data distribution patterns. However, existing approaches primarily focus on using AI for knob tuning or optimizing specific system components based on specific logs or patterns. As a result, their designs often appear heuristic and static, failing to provide thorough optimization that fully interacts with and enhances the system. It still lacks a holistic approach where AI is the first citizen to fully exploit AI's potential in data systems.

We envision that in-database AI-powered analytics will catalyze a comprehensive evolution of data system architecture and techniques. This transformation requires AI optimizations to make the data system scalable, adaptive, and intelligent.

3.2.3 *Privacy-persevering and trusted AI×DB*

As the need for analytics on sensitive data and data collaboration intensifies [19], ensuring data privacy and trustworthiness has become indispensable. Existing privacy-persevering techniques typically focus on AI or databases separately, overlooking the specific challenges presented by AI×DB. On the one hand, while privacy-enhancing techniques work in the AI domain, such as federated learning [20,21] that enables multiple parties to collaboratively train models and make inferences, these approaches are not directly applicable to in-database AI models. On the other hand, traditional database security methods [22], such as data masking and auditing, are insufficient to protect the data during the interaction with AI components.

We envision that new privacy issues will emerge during the development of NeurDB, requiring the data system to natively support data and AI model management in a privacy-preserving and trusted manner.

3.3 Conceptual framework of NeurDB

To address these challenges, we establish the conceptual framework of NeurDB in Figure 3, which underpins its overall design and functionality.

3.3.1 *In-database AI-friendly ecosystem*

We advance beyond existing works by ensuring that NeurDB can inherently support AI applications, thereby improving model accuracy, efficiency, and usability. We focus on two aspects: a) Developing easy-to-use, AI-friendly functionalities in NeurDB. We enhance database functionalities to facilitate efficient and user-friendly AI model management, i.e., allowing users to conveniently use AI models and retrieve the required data by models; b) Investigating efficient in-database model training and inference techniques. First, we support in-database neural architecture search to minimize data transfer overhead, ensuring more efficient model training. Second, we explore in-database model slicing techniques to serve as a query optimizer for AI models, to enhance model inference efficiency. Third, we propose to design component-based modeling, such as modular neural networks, enabling a mix-and-match approach for improved efficiency and accuracy of AI models.

3.3.2 *Intelligent self-driving data system*

We fundamentally redesign the database architecture and propose AI-driven database optimizations tailored for AI-powered hybrid transactional and analytical processing (AI+HTAP). To this end, we focus
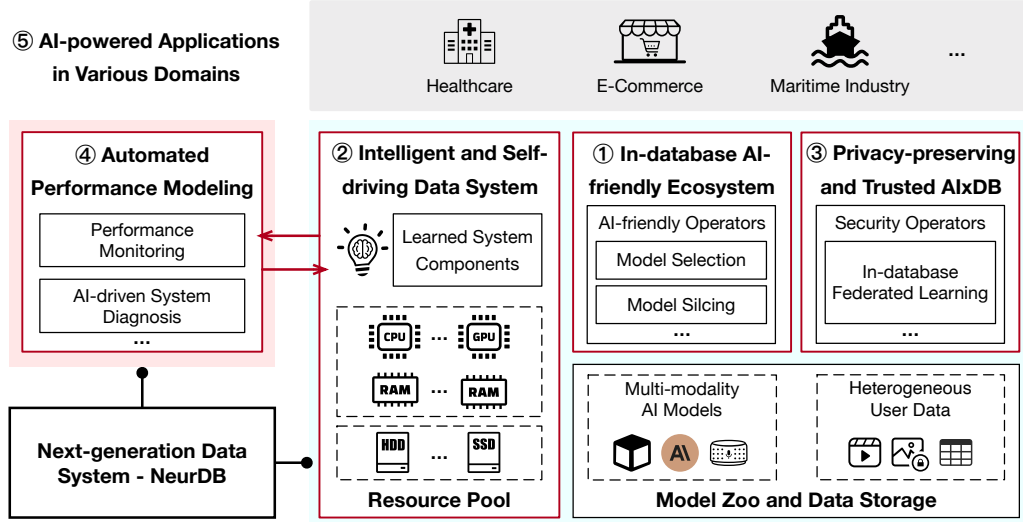
Figure 3   Conceptual framework of NeurDB

on investigating scalable intelligent database design and advanced AI-driven database optimization techniques. First, we aim to identify the requirements for making AI an integral part of NeurDB, and propose a new AI-powered data system architecture. Second, we incorporate AI components with the recent evolution of databases, such as cloud-native disaggregation, new hardware, etc., to make NeurDB scalable and self-driving. Third, we ensure high-performance AI+HTAP by designing specific data structures to provide efficient and effective AI functionalities, such as in-database Retrieval-Augmented Generation (RAG), etc.

3.3.3   *Privacy-preserving and trusted AI×DB*

We develop new techniques to address the issues that arise with AI×DB for enhanced data protection. Our focus is on two main areas: a) Privacy-preserving AI×DB. We propose incorporating secure operators (e.g., secure multiparty computation protocols) into databases to facilitate in-database federated learning for strong privacy protection with high efficiency. Moreover, it is crucial to ensure that AI models are not vulnerable to various inference attacks. Hence, we aim to design mechanisms that offer in-database differential privacy to protect the data utilized for training the AI models. b) Trusted AI×DB. We propose integrating the verifiable data structure into the native database engine to ensure the integrity of data accessed by in-database AI models. Further, we propose employing zero-knowledge proof (ZKP) techniques to verify that the computational results derived from the data are well-formed and trustworthy.

3.3.4   *Automated performance modeling for AI×DB*

New automated validation approaches are required to ensure the correct, secure, and performant functioning of NeurDB. However, existing methods face limitations: they assume the data system will produce deterministic results, rely on self-contained test cases, and expect a stable system state. This assumption does not apply to systems like NeurDB, which is AI-driven, adapts to its workloads, and natively supports various AI workloads. To tackle these issues, we propose to design a new adaptive and automated validation system that can automatically generate test inputs for NeurDB to validate the system performance, security and fairness, and the effectiveness and efficiency of the supported AI applications. In addition, to gain insights into usability and deployment challenges, we plan to explore real-world deployment opportunities to validate the practicality of our proposed systems and techniques. We plan to utilize NeurDB to support actual AI application development, such as AI healthcare analytics developed by clinicians for their data, and implement NeurDB in industry settings, by working with cloud service providers and e-commerce vendors.
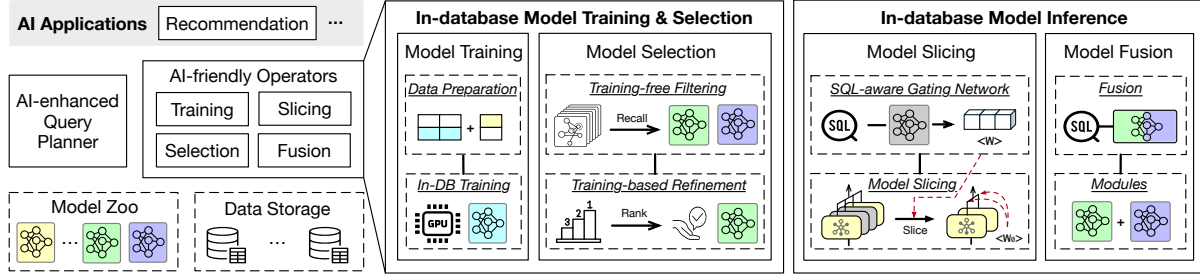
Figure 4 Overview of in-database AI-friendly ecosystem

3.3.5 *AI-powered applications in various domains*

We plan to develop a set of application-oriented toolkits to provide built-in AI as a service for non-experts in AI and computer science, especially for users in the healthcare and e-commerce domains. Healthcare, with its complex data and growing need for accurate and personalized predictions, and e-commerce, with its vast consumer data and dynamic user needs and market trends, represent two critical domains where the integration of AI×DB can yield transformative results for enhancing user experience and improving operational efficiency [23, 24]. Current practices in these domains treat data management and AI as separate components, requiring domain knowledge in both areas [25]. Such a segregated solution limits the potential of AI applications and creates barriers to innovation and vast application. We aim to build our next-generation AI×DB system to support more personalized, adaptive, and scalable healthcare and e-commerce applications: a) Declarative interface with minimal domain knowledge requirement. NeurDB aims to introduce a user-friendly, declarative interface, akin to LLMs, which simplifies the interaction with AI models. b) Usability and scalability for large-scale data. Through NeurDB's unique AI-powered analytics, we aim to improve applications like Foodlg, which we designed to monitor and balance patients' diets, to support the entire treatment process autonomously with NeurDB's integration.

4 In-database AI-powered analytics

In this section, we elaborate on the architecture of the in-database AI-friendly ecosystem and detail the key techniques that we have proposed to support in-database AI-powered analytics for NeurDB.

4.1 AI-friendly ecosystem overview

We establish the AI-friendly ecosystem of NeurDB following the architecture illustrated in Figure 4. At the core of this ecosystem is an AI-enhanced planner, tasked with various automatically activating AI-friendly operators to process complex analytical queries. These operators integrate the entire workflow of AI applications within the data system, which ensures NeurDB is inherently AI-friendly. In NeurDB, we introduce two main categories of AI-friendly operators: model training & selection operators and model inference operators. 1) Model training & selection operators are designed to ensure task-specific model effectiveness without costly data transformation. In particular, the model training operator invokes the training process for specified tasks within the database, while the model selection operator aims to obtain the best-performing model with the optimal network topology for each task. 2) Model inference operators are employed to perform accurate and efficient inference. Specifically, the model slicing operator adjusts the AI model to better suit the given analytical query, and the model fusion operator enables modular neural networks that combine multiple models with a mix-and-match technique.

Consider the recommendation task in e-commerce as an illustrative example. NeurDB efficiently and effectively supports such tasks with its AI-friendly functionalities. For personalized item recommendations, the planner of NeurDB can automatically generate an execution plan that might include a model selection operator followed by a model slicing operator. In particular, the model selection operator identifies an appropriate model architecture for the task from the model zoo, which is then used to perform the inference. However, the selected model, often generalized to capture common patterns and trends applicable to all users, may not achieve the highest accuracy for personalized recommendations due to its lack of focus on individual user characteristics, e.g., age and income. To address this, we employ the model slicing operator to automatically refine the model, ensuring it better aligns with specific user
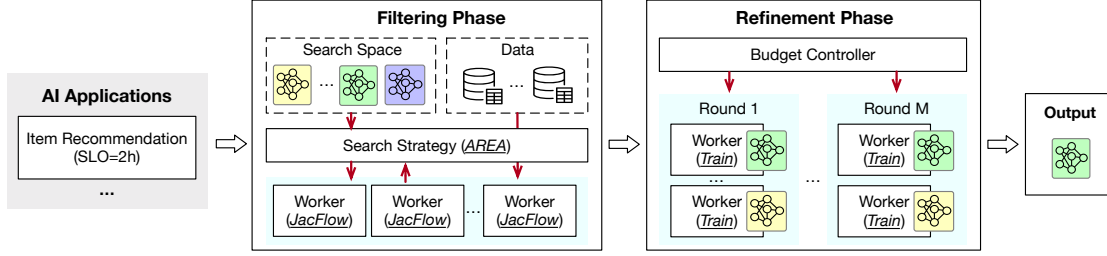
Figure 5 SLO-aware dynamic model selection

characteristics for more accurate recommendations. Moreover, NeurDB might perform a model fusion operator, which integrates models from related tasks, such as a model for the click-through rate (CTR) prediction, to boost recommendation accuracy.

Integrating AI functionalities into databases requires more than simply embedding AI models within them [26–29]. Besides supporting AI-friendly operators, data systems also need to efficiently track model evolution, manage pipelines, and interact with other internal/external tools, e.g., dynamic model versioning, automated pipeline generation, cross-model interaction, etc. Such system level interactions beyond monolithic AI models have been implemented in some of our solutions such as MLCask [26], which functions like a Git-like system for managing the entire machine learning lifecycle, and IP8Value [30], which is an application specific data system that leverages AI and LLM for intellectual property management.

We now detail the key techniques that NeurDB employs to build the AI-friendly ecosystem.

4.2 Dynamic model selection

Given the costly data transformation required by existing systems, we are therefore motivated to support in-database dynamic model selection. However, providing efficient and effective in-database model selection is not trivial. Existing solutions are typically categorized into two types: training-based and training-free. Training-based algorithms [18, 31] can effectively measure the model performance but typically require training and evaluating hundreds to thousands of candidate models, which are computationally prohibitive in both time and resources [32]. In contrast, training-free model selection algorithms [33–35] estimate model performance by computing certain statistics of candidate models without training, namely Training-Free Model Evaluation Metrics (TFMEMs). While leveraging TFMEMs allows for the quick evaluation of numerous models, these approaches often compromise accuracy, potentially limiting the discovery of the most effective models compared to training-based algorithms that offer more precise evaluations.

Unlike existing works, we aim to develop a model selection algorithm for NeurDB that harnesses the advantages of both training-free and training-based paradigms. Our idea is to employ training-free algorithms to filter down the candidate model set, and then use training-based algorithms for refinement, i.e., obtain the most effective model. Recognizing the inherent trade-off between training-free and training-based paradigms, careful coordination of these two phases is essential. To address this, we incorporate Response-time Service-level Objectives (SLOs) [36, 37], which are strict response-time thresholds predefined by users. Failing to meet the SLOs can lead to reduced query service quality or even financial losses [36]. Based on SLOs, we allocate specific time budgets for both the training-free and training-based phases to properly balance efficiency and effectiveness.

We propose an SLO-aware model selection algorithm, which consists of a training-free filtering phase and a training-based refinement phase as shown in Figure 4. First, in the training-free filtering phase, we efficiently explore a candidate set of promising models with higher scores. Specifically, we utilize an enhanced TFMEM named JacFlow [38] to calculate a score for each model under consideration. JacFlow integrates expressivity-based [39] and trainability-based [40] TFMEMs to improve model ranking accuracy. Second, in the training-based refinement phase, we determine the optimal model from the candidate set through multiple rounds of examination. Each round involves training the candidate models for a specified number of epochs, progressively narrowing the candidate set for the next round's evaluation. Further, to align this process with SLO, we have developed an SLO-aware function that manages three critical parameters: the number of models to be scored in the filter phase (N), the candidate set size (K), and the training epochs used in the refinement phase (U). By so doing, we ensure the selection of
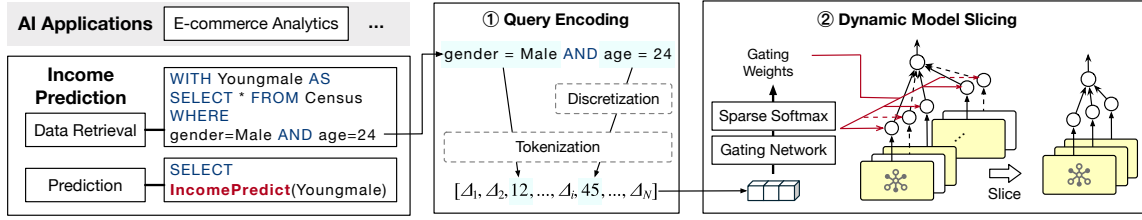
Figure 6   SQL-aware dynamic model slicing

a higher-performing model while meeting the predefined response-time threshold. For further details on the SLO-aware model selection algorithm, please refer to [41].

As shown in Figure 5, we present a running example of the model selection process with an SLO of two hours for the item recommendation task discussed in Section 4.1. During the filtering phase, we obtain a candidate set of promising models. To enhance efficiency, we deploy multiple workers to score the models using JacFlow in parallel. Given the potential of numerous models, scoring each model is infeasible and costly. Thus, we implement the asynchronous regularized evolution algorithm (AREA) [42] to search for the most promising models for scoring and optimally distribute these models among workers. The filter phase terminates once N models are scored, from which the top K scored models are selected to form the candidate set. In the subsequent refinement phase, we train and evaluate the models in the candidate set with multiple rounds. In each round, each model in the candidate set is trained for U epochs and then assessed to retain only the top $1/\eta$ performing models, with η being a predefined factor regarding the epoch count. We initiate the first round with $U = U_{init}$, and in subsequent rounds, U is adjusted to $U' \cdot \eta$, where U' denotes the epoch count used in the previous round. This incremental training approach allows each model to receive a progressively larger training budget in successive rounds, facilitating a more precise evaluation. In this way, we finally obtain the best-performing model.

4.3   Dynamic model slicing

Model slicing is a critical technique for adjusting AI models to meet the specific requirements of various analytical tasks. It customizes the architecture of deep learning models based on analytical queries, thus improving the model inference accuracy. As data volumes grow, analytical queries often focus on specific subsets of the data rather than the entire database. Traditionally, a single general model is used to support inference for different analytical queries [43, 44]. However, such a model, optimized to capture common patterns across the entire database, may not deliver as accurate inferences as a dedicated model trained on a specific subset of interest. For example, when examining patient data, analysts prefer to study patterns or trends among individuals diagnosed with a particular disease. A model exclusively trained from the data of diabetic patients might provide more precise predictions than the general model for all patients regarding their readmission rates. Given the vast number of possible analytical queries, training a dedicated model for each one is also computationally impractical. To this end, we consider leveraging query conditions to perform model slicing. In particular, for each analytical query, we analyze and encode the query conditions, which indicate the specific data required for the query, and then use this information to adjust the model architecture accordingly.

We proposed LEADS [45], a new SQL-aware dynamic model slicing technique that enables the customization of the predictive model for a subset of data retrieved by analytical queries. The core of LEADS has three parts: (1) the construction of the general model, (2) the encoding of the analytical query, and (3) the slicing via SQL-aware gating network. The goal of the first step is to improve the capability of the predictive base model. We scale it horizontally following the Mixture-of-Expert (MoE) technique to form a general model. The general model is composed of multiple replicas of the base model. These replicas, termed experts, are trained to specialize in different problem subspaces for more effective predictive modeling. Given the predictive base model as $\mathcal{F}(\mathbf{x})$, the general model is represented as $\mathcal{F} = [\mathcal{F}_1, \mathcal{F}_2, \cdots, \mathcal{F}_K]$, where K is the number of experts. In the second step, we model the filter condition in the analytical query. Considering the potential combination of filter conditions, for simplicity, we focus on individual analytical queries, referred to as primitive queries. In the primitive query, each attribute should be associated with zero or one predicate, with predicates across attributes conjoined using the logical operator $\wedge$ (AND). The i-th attribute value of predicates is denoted by q_i. Particularly, a padding value is used if there are no predicates in attributes. Given N attributes in the dataset, we can obtain

a fixed-dimensional embedding of the analytical query, $\mathbf{q}$, by concatenating the attribute values of the predicates: $\mathbf{q} = [q_1, q_2, \cdots, q_N]$. Further, a SQL-aware gating network $\mathcal{G}$, a combination of a two-layer neural network and a sparse softmax layer, is employed. For a specific analytical query $\mathbf{q}$, the SQL-aware gating network dynamically generates gating weights for each expert in the general model, denoted as $\mathbf{w} = \mathcal{G}(\mathbf{q}) = [w_1, w_2, \cdots, w_K]$. $w_i = 0$ indicates that the corresponding i-th expert is not required in the current predictive modeling. Thus, we only activate a small fraction of experts $\mathcal{F}_i$ for the model inference.

Figure 6 shows a practical application of LEADS in business analytics. Consider an analytical query that predicts the income of 24-year-old men. The query's execution can be divided into two phases: data retrieval and prediction. During data retrieval, filter conditions are applied and undergo query encoding. As depicted in Figure 6, the categorical value "gender = Male" is directly mapped to a token. Meanwhile, the numerical value "age = 24" is discretized into a range before tokenization. If a column lacks a filter condition, the corresponding position is padded with a default token. This process yields a fixed-sized query token capturing metadata about incoming predictive data. Subsequently, we customize the general model via model slicing. The query embedding is fed into a gated network. Leveraging a well-designed sparse softmax, recalibrated gating weights are obtained, where low scores are set to zero. Based on these gating weights, the experts within the scaled-up general model are activated selectively. Experts with zero scores are disregarded, and the remaining activated models constitute the slice model. This sliced model is then applied to the subsequent predictions, facilitating insightful analytics tasks.

4.4 Dynamic model construction with plugins

Component-based modeling is a neural network technique designed to manage AI models by dividing them into several modules. This technique enables independent development of each module, which can be easily added, removed, or replaced in a plug-and-play manner. Such flexibility and extensibility facilitate easier model implementation, upgrades, and maintenance. We have utilized component-based modeling in the healthcare domain and demonstrated its benefits for data analytics in our previous work [46–48]. There are also several related works to support component-based modeling in various other domains. However, as AI models become more advanced and complex, existing approaches face two key limitations. First, given today's diverse AI models, there may exist a large number of modules for various purposes. Consequently, efficiently managing these modules while ensuring flexible and extensible model construction at runtime is challenging. Second, establishing these modules often requires specific domain knowledge such as functions, data flows, or analytical pipelines, and therefore, this process is rather complex for end users with limited domain knowledge.

Unlike existing works, we aim to design a data-driven and database-native component-based modeling technique to make NeurDB generalizable for various AI-powered analytics. Our basic idea is to leverage the fundamental features offered by databases, such as efficient query processing and data storage, to embed and integrate the entire component-based modeling process within the database. However, solving the problem of efficient module management and user-friendly module designs is not trivial. We first create tailored storage for module management, which includes an index specifically designed for a large number of modules. Building on this, we then develop a model fusion operator for efficiently retrieving the required modules during the execution of analytical queries. We extend the query executor to support this model fusion operator. The fusion operator is executed following the widely used volcano execution model, which first processes independent modules in parallel, and then applies multimodal fusion to capture cross-module interactions. To improve user-friendliness, we enhance the query optimizer to guide the model fusion. In particular, the model fusion operator can be automatically triggered upon receiving a complex analytical query. This enhanced query optimizer generates a proper execution plan including the appropriate modules needed for this query. Further, we plan to employ the query `analyze` and `explain` syntax to enhance the interpretability of the model fusion process.

5 AI-powered autonomous data system

In this section, we present the architecture of autonomous NeurDB, and detail the key components that we have developed to transform NeurDB into an AI-powered autonomous data system.

5.1 Autonomous architecture overview

The ideation of making databases autonomous and adaptive was made in various forms some forty years ago [49,50], and it has been revisited over times [8,51]. A fully autonomous data system typically refers to a system that can dynamically adapt itself without human intervention. Such a system is expected to determine appropriate actions, execute them, and learn from the outcomes to achieve dynamicity. These actions can include performance optimization, system repairs like replacing a failed node, and other self-maintenance tasks. Although significant progress has been made toward achieving a fully autonomous data system, existing systems still require some level of human intervention. For example, system repairs can be triggered automatically but often require human validation to avoid data inconsistency. However, with the rapid advancement of AI techniques and new database architectures, it is about time to design a fully autonomous data system by leveraging these new developments and foundations.

We propose a new autonomous data system architecture for NeurDB, shown in Figure 7. This architecture is disaggregated into two distinct layers: the computation layer and storage layer, enabling each layer to scale independently [52]. Such disaggregation facilitates the system's efficiency, elasticity, resilience, and scalability. Each layer consists of multiple nodes: in the computation layer, nodes process transactions and deliver results to users, while in the storage layer, nodes are dedicated to data storage. NeurDB includes essential operational components in the standard database design of the computation layer, such as the query optimizer, query executor, transaction manager, etc. These components are facilitated by AI to boost not only performance but also functionalities. For instance, the query optimizer in NeurDB is now tasked with co-optimizing plans for both AI operators and traditional data operators, e.g., JOIN, incorporating insights from the AI-powered planner discussed in Section 4.1. To accommodate the diverse and dynamic workloads in real-world applications, we introduce an AI-enhanced adaptor to enable NeurDB as a self-assembling and self-optimizing data system. With the flexibility of the disaggregated architecture, there are more optimization spaces and strategies that can be explored by the adaptor. In the computation layer, the adaptor intelligently assembles system components, such as selecting appropriate indexes and adjusting the concurrency control strategy, and allocates resources dynamically in response to specific workload demands. In the storage layer, the data layout is automatically optimized based on workload requirements, with data co-location adjustments made to ensure fast access.

We now illustrate the autonomy of NeurDB by discussing its efficient support of dynamic AI+HTAP workloads. As shown in Figure 7, NeurDB creates specialized nodes tailored to the demands of various workloads in the computation layer. For online analytical processing (OLAP) workloads, nodes are provisioned with both CPU and GPU resources to handle the computational intensity. In particular, for AI-powered analytics, NeurDB configures nodes that combine extensive CPU and GPU resources, coupled with an AI-optimized query optimizer. It employs high-dimensional learned indexes, suitable for RAG, and adopts optimistic concurrency control to manage the anticipated low-contention OLAP workloads. In contrast, for online transactional processing (OLTP) workloads, we would allocate nodes equipped primarily with CPU resources to optimize resource utilization, as transactional tasks can be efficiently processed without the need for GPU support. NeurDB opts for B^+-tree indexes and employs lock-based concurrency control to handle potential high-contention reads and writes in OLTP workloads. Regarding the storage layer, the adaptor can automatically organize a replica of data into a column format for OLAP workloads, which is more suitable for OLAP query patterns, while keeping other replicas in row format.

Developing such an efficient and effective autonomous data system is not trivial, particularly given the dynamic nature of workloads. The equipped AI model must adapt to these variations to maintain accuracy while minimizing impact on system performance. For example, while learning from specific instances of datasets and query patterns can potentially obtain improved query plans, such improvements may be short-lived due to workload shifts and data updates. We therefore need to design new techniques to overcome these challenges. In particular, we aim to design dynamic and online AI models that facilitate continuous optimization of system components for better performance. Given that the Filter-and-Refine paradigm (FRP) is used as a common-known strategy in indexing [53–55], query processing [56,57], etc., our core idea generally follows the FRP and employs online reinforcement learning (RL) to facilitate dynamicity. RL naturally aligns with FRP, as it involves filtering through a range of possible actions and then refining the most effective strategies based on feedback. In particular, FRP's filtering stage is akin to RL's exploration phase, where the RL agent tries different actions to understand their effects on the environment, gathering information to guide future decisions. Further, the exploitation phase of RL
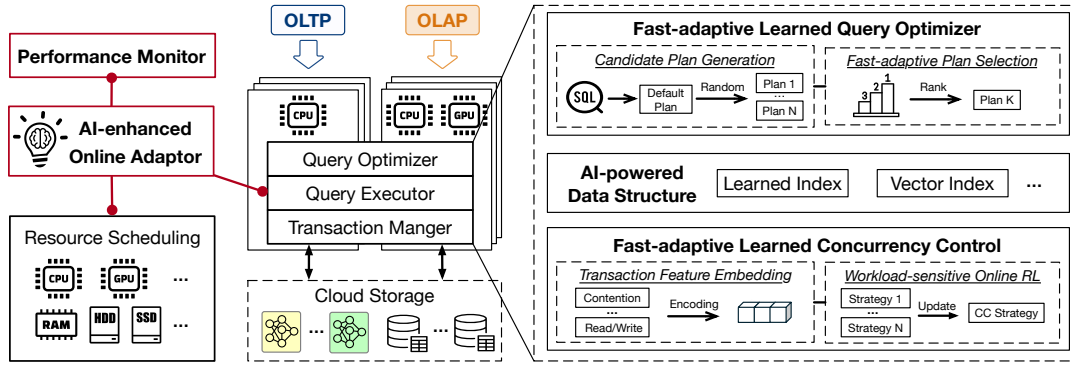
Figure 7 Autonomous architecture of NeurDB

shares the same objectives as the refinement step of the FRP, where the RL agent refines its strategy by focusing on actions that yield higher rewards. During this phase, the RL agent also optimizes its strategy to maximize long-term gains by learning from past experiences. With these design principles in mind, we now elaborate on our key techniques for the query optimizer and concurrency control to realize seamless online adaptation in NeurDB.

5.2 Fast-adaptive learned query optimizer

The query optimizer is a crucial component of database systems, tasked with selecting the most efficient execution plans for given SQL queries. Traditional query optimizers are cost-based and typically consist of three main components: plan enumerator, cardinality estimator, and cost model. Due to the difficulty of cracking the cardinality estimator and cost model, there is a growing interest in harnessing machine learning to enhance existing query optimizers [58–62]. Although those learned query optimizers demonstrate advantages over traditional ones in certain applications, they are not yet practical for real-world database systems. First, machine learning models might overfit to the training data, resulting in poor adaptability to new or dynamic query workloads. Second, existing work adopts periodic retraining to incorporate new data, as a result, models can forget previously learned knowledge when updated with new information.

We therefore propose an effective and fast-adaptive learned query optimizer. For the effectiveness, the optimizer must effectively find a near-optimal execution plan with lower query response time for a single query. For fast adaptivity, the optimizer must quickly learn from limited experience related to unseen queries and can remember the learned knowledge without suffering from catastrophic forgetfulness. To achieve this, we divide the proposed learned query optimizer into two stages: efficient candidate execution plans generation, and fast-adaptive plan selection via online reinforcement learning, as shown in Figure 7.

During the plan generation stage, we generate a set of potential execution plans, while keeping the set size small and ensuring the inclusion of plans that are close to optimal. To this end, we obtain candidate execution plans by mutating the optimizer's cardinality estimates rather than relying on a pre-fixed set of hints as in existing works [59]. In particular, we first employ the default optimizer to obtain a base execution plan for a given query. We next construct a set of varied cardinalities by applying multiplicative factors to the original cardinalities provided in the base plan. After that, we uniformly sample from these mutated cardinalities and feed them together with the original query into the default optimizer again to generate a new execution plan. This process is repeated N times to produce N candidate execution plans. Each plan may differ in execution strategies, such as join order, join algorithm, index usage, etc., due to the varied cardinality estimates.

In the plan selection stage, we utilize online reinforcement learning to select the near-optimal execution plan, while achieving fast-adaptive to unseen queries and remembering the learned knowledge without suffering from catastrophic forgetting. Specifically, we model the execution of queries as a sequence of events and use a sequence model, i.e., decision transformer [63, 64], to select the near-optimal execution plan. For a given query and its corresponding candidate plans, we predict which plan may yield the best rewards (lowest latency) based on not only the current query but also the query execution log. With this, we could capture meaningful context information and query correlations. We further introduce the hyper-network and adaptor layer inside each transformer block. The hyper-network accepts inputs of the query and decides which adapter layer this query should use. If this query is determined as an unseen

query, the hyper-network will generate a new adapter with generated initialized parameters to facilitate fast training. If this is a known query, hyper-network directly uses the existing one.

5.3   Fast-adaptive learned concurrency control

Concurrency control is crucial in transaction processing to ensure ACID properties [3] and achieve optimal performance. Traditional concurrency control algorithms are generally rule-based. For instance, two-phase locking (2PL) relies on lock compatibility to order transactions, while optimistic concurrency control (OCC) depends on order validation techniques. Given the diversity of real-world workloads, a fixed rule-based approach often does not work well across all scenarios. This limitation has triggered the development of hybrid and learning-based concurrency control algorithms [65–67]. These algorithms typically analyze specific workloads, either manually or automatically, to select the most suitable concurrency control algorithms for different types of transactions. While they outperform traditional methods in certain scenarios, they struggle to adapt to the dynamic workloads in real-world databases. Consequently, they may not consistently yield optimal concurrency control as workloads and data distribution change.

We aim to design a learned concurrency control algorithm that can continually adapt to changing workloads, ensuring both efficiency and practicality. Unlike existing approaches, we do not focus on learning from specific workloads. Instead, we aim to develop a general concurrency control strategy that reflects workload shifts through system statistics, allowing for dynamic adaptation. This means that our concurrency control strategy continuously updates based on system behavior to accommodate workload changes. However, achieving this objective presents two key challenges. First, the system must be modeled precisely to detect workload shifts in real-time. Second, upon adaptation, the model update must be as fast and accurate as possible to ensure a rapid response to changing workloads.

To address these challenges, we propose an online learned concurrency control algorithm that ensures quick adaptation to dynamic workloads. We first design a system representation by carefully selecting critical factors, such as throughput and average lock waiting time, to represent system conditions accurately. Based on these factors, we then establish a general concurrency control strategy that maps specific system conditions to corresponding actions. For each transaction, this strategy determines the appropriate concurrency control action. For instance, when a write operation is performed on high-contention data items, we require a lock immediately to avoid wasted context switching overhead, while for a read operation on low-contention data, we avoid locking to enhance performance.

By monitoring these factors, we can detect workload changes through their impact on system conditions. For example, if the lock waiting time suddenly spikes, it indicates that the system is under high load. In this case, we need to trigger the concurrency control strategy adaption. We then propose a fast concurrency control adaptation model based on online Reinforcement Learning (RL). To ensure the strategy can continuously update with changing workloads, we employ a two-phase adjustment algorithm following FRP. In the filtering phase, we generate several improved strategies using an evolutionary algorithm (EA) and test them over a specific timeframe to identify the top strategy. In the subsequent refinement phase, we use reward-based feedback to further refine and improve the selected strategy.

6   Privacy-preserving and trusted NeurDB

In this section, we present the architecture of NeurDB from the privacy and trustworthiness perspective, detail in-database federated learning, and discuss the online recovery technique to correct data tampering.

6.1   Privacy-preserving and trusted architecture overview

With the escalated demand for performing collaborative analytics on sensitive data, data privacy and trustworthiness have become indispensable. For example, different organizations may train machine learning models on their joint databases, where each organization must protect its own data while being able to verify the authenticity and execution correctness of data from others. Another example is that users may upload data and models to the database systems deployed on the cloud for data analytics; users have to ensure that the data and models are protected and not tampered with by the cloud. To cope with the various needs of users, we design the privacy-preserving and trusted functions of NeurDB based on the following three design goals: a) the secure functions for data privacy and trustworthiness
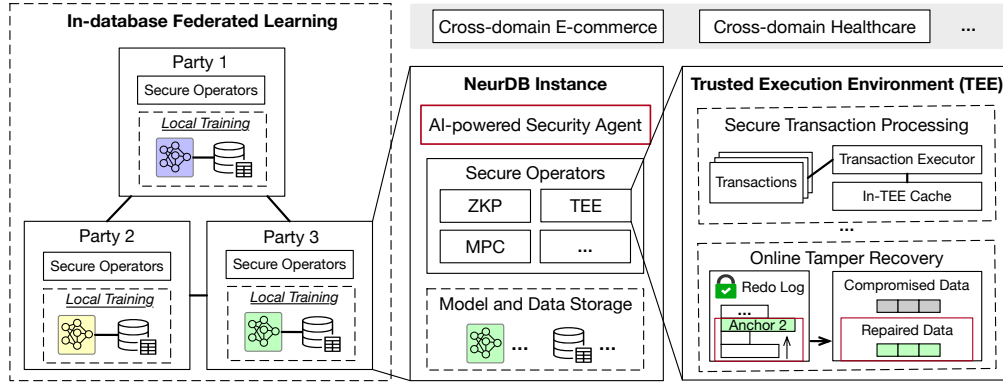
Figure 8   Overview of privacy-preserving and trusted NeurDB

can be added on demand and easily managed by users, b) the database system must natively support privacy-preserving data collaboration while being adaptive to prevent the malicious behaviors timely, c) the system must be autonomous and able to generate task-specific privacy and trustworthiness policies.

NeurDB presents an integral solution of AI×DB to guarantee data privacy and trustworthiness. First, it introduces various built-in secure operators to support hot plugging and user-friendly secure services. Second, it supports in-database federated learning, which enables privacy-preserving and trusted data collaboration. Lastly, NeurDB embeds an AI-powered agent to automate privacy-preserving and trusted task generation and policy configuration. Figure 8 shows the architecture of our privacy-preserving and trusted NeurDB. It consists of a task handler, a coordinator, executors, and an AI-powered security agent. The task handler is responsible for parsing the users' requests and generating execution plans for the secure operators. The coordinator distributes the task to the executors. The latter performs the tasks according to the privacy and trustworthiness policies of the operators and returns the results to the coordinator. The AI-powered security agent monitors the users' requests, and assists users with the task generation and policy configuration.

To enhance the inherent privacy and trustworthiness of NeurDB, we redesign the database's security paradigm by introducing new privacy-preserving and trusted operators. These operators are designed to trigger autonomously in response to predetermined criteria, thereby removing the necessity for manual intervention. Based on our extensive background in federated learning [20, 21, 68, 69], privacy-preserving data analytics [70, 71], and database security [22, 72], we develop innovative techniques for facilitating the autonomous deployment of these operators. We describe the fundamental operators as follows:

- `ENC(TYPE, TEXT)`, `ENC_OP(TYPE, MSG, MSG)`, `DEC(TYPE, MSG)`. These operators are responsible for encrypting data, decrypting data, and executing operations on encrypted messages. The `ENC` operator takes the encryption scheme and plaintext data as input and returns the encrypted message. We support various encryption schemes such as fully homomorphic encryption (FHE) and partial homomorphic encryption (PHE) with additive and multiplicative homomorphism. `ENC_OP` takes in encrypted or plaintext messages and calculates the results of the corresponding operations. `DEC` will decrypt the message back to plaintext with the secret keys.

- `MPC(OP, PARTY.DATA, ...)`. Secure Multi-Party Computation (MPC) enables multiple parties to jointly calculate a function using their individual inputs while keeping those inputs private. The `MPC` operator supports multi-party computation by taking as input the operator and data from a number of parties. It returns the final result to the users hiding all the details of the MPC protocol. We support the state-of-the-art additive secret sharing scheme (SSS) in MP-SPDZ.

- `DP(DATA)`. Differential privacy (DP) offers a rigorous privacy guarantee for protecting individuals' sensitive information. It ensures that the probability of producing a given output does not depend very much on whether a particular individual is included in the database. The `DP` operator allows the system to add controlled random noise before releasing the output, e.g., the trained model.

- `ZKP(DATA)`. Zero-knowledge proofs (ZKPs) allow one party (the prover) to prove the truth of a statement to another party (the verifier) without revealing any information other than the statement's

truthfulness. The `ZKP` operator allows users to query the data and verify the correctness of the data and operation with the proofs and operation with the proofs provided by the system.

- `TEE`. NeurDB provides options for privacy-preserving and trusted operations with the support of trusted hardware. By specifying `USING TEE`, all user queries and data will be executed and stored within the trusted execution environment (TEE), such as Intel SGX (Software Guard Extensions), so that other adversaries in the system could not exploit such private information.

- `LEDGER`. NeurDB supports distributed ledger schema with the keyword `LEDGER`. Data stored in `LEDGER` is immutable, transparent, and verifiable. `LEDGER` guarantees the correctness of the data, history, and queries. It also tracks the lineage of data. For all queries involving `LEDGER`, the system will provide integrity proof in addition to the data selected or inserted into the `LEDGER`.

6.2   In-database federated learning

To enable privacy-preserving and trusted data collaboration, NeurDB supports in-database federated learning leveraging the secure operators defined in Section 6.1. In traditional federated learning scenarios, users need to extract the datasets from their databases and preprocess the datasets before sending them to the models for training. The whole process requires multiple rounds of data processing and copying, leading to significant overhead. In contrast, data in NeurDB are ready for training due to the built-in secure operators. Data can go through pre-processing and training while being queried.

As shown in Figure 8, the training process is distributed across multiple decentralized NeurDB instances without needing to move or centralize the data. Each participating instance retains its data locally. Instead of moving data to a central location, the federated learning algorithm enables the model to be trained locally on each database and only the aggregated global model is released. The aggregator combines these parameters from all participating models to create an updated, global model. This process of training locally and aggregating globally repeats several times. Each iteration refines the global model further, without ever exposing the individual datasets. NeurDB supports both horizontal and vertical federated learning to handle various scenarios.

To protect data privacy and ensure trustworthiness for NeurDB's in-database federated learning with high efficiency and utility, we design novel protocols based on the built-in secure operators. First, we propose a hybrid technique based on MPC and PHE operators. This technique employs PHE as much as possible to facilitate each party's local computations and utilizes MPC when PHE cannot compute the functions. This technique protects the information exchanged among the parties for federated training and improves the efficiency of secure computations. Second, we incorporate DP into in-database federated learning to prevent the adversary from inferring the sensitive information of each party's local data. Specifically, we formalize a novel notion of label distributional privacy and propose a label distribution perturbation mechanism, which is a variant of stochastic gradient descent (SGD) that satisfies label distributional privacy, protecting each party's labels during federated training. Third, we present a ZKP-based federated learning algorithm that is highly efficient for secure and verifiable data collaboration, which not only protects the privacy of model updates that each party uploads to the server, but also guarantees that the model updates are well-formed such that malicious updates can be detected. Particularly, we devise an optimized ZKP generation and verification technique that significantly reduces the ZKP cost based on probabilistic integrity checks. Fourth, we introduce an efficient, verifiable distributed ledger using a two-level Merkle-like tree structure to log the training and prediction activities of in-database federated learning, which ensures an immutable and transparent collaboration history with high throughput.

6.3   Tamper recovery and fault tolerance

Data tampering is a critical issue that can potentially cause a system to be unavailable. If data is tampered with, transactions accessing this data need to be blocked to prevent further damage. Existing blockchain systems based on Byzantine Fault Tolerance (BFT) protocols can mitigate this issue but suffer from significant performance overhead. Another line of related research, namely verifiable databases, typically restores the entire system to a previous snapshot, which can lead to the loss of recent updates. This demonstrates that correct and efficient tampering recovery remains an open problem.

Unlike existing works, we aim to prevent the system from being unavailable due to data tampering without additional overhead on normal transaction processing. To achieve this, we execute transactions on data that require protection by specifying `USING TEE`. All transaction operations are performed in

TEE so that if required data is tampered with, it can be detected when it is loaded into TEE. If data tampering is detected, we employ a log-based tampering recovery mechanism that allows transactions to continue once the tampered data is repaired. Inspired by commonly used log recovery techniques [73], we consider replaying all previous operations on the compromised data for correction. To this end, we enhance the redo log to be secure by encrypting it with a signature-based mechanism. Specifically, we generate a unique hash for each transaction's log entries, which is signed in TEE to ensure any tampering with the log can be detected. We then retrieve corresponding logs into the TEE, in which we replay the missing operations to repair the data to its correct state. With TEE, the recovery process is shielded from tampering.

Achieving efficient log replay for tampering recovery is not straightforward. Since traditional redo logs only contain information about changes, rebuilding a corrupted data item requires replaying from the first log entry that created it. This process can lead to lengthy replay times due to the large volume of logs that must be replayed. We therefore introduce anchor logs to facilitate efficient and online recovery. In particular, an anchor log is created after every n modification to a specific data item, where n is a tunable parameter. Differing from standard redo logs, anchor logs record the entire content of the data item. For example, consider a transaction that updates a data item x_i to x_i', representing the n-th modification of x_i since the last anchor log. In response, we create a new anchor log that captures the entire content of x_i'. This allows the recovery process to start from the most recent anchor log, applying subsequent changes incrementally, thereby avoiding replicating logs from the beginning. After the recovery process, the affected transactions can continue execution with the correct data.

7 Implementation of NeurDB

We shall now discuss the development of NeurDB, particularly focusing on its implementation. NeurDB is built on the PostgreSQL codebase, which we significantly enhance with our proposed AI-powered architecture and techniques. In the implementation, we are concerned with efficient interactions between the database engine and in-database AI models. Minimizing interaction with AI models is not straightforward for the following two key reasons. First, AI models are often implemented in Python, while the database is primarily written in C/C++, which incurs overhead from data copying between different programming languages. Second, model updates in NeurDB are necessary to adapt to dynamic workloads. However, frequent updates to AI models can be costly if each update requires reloading the models. To address these issues, we leverage a carefully designed shared memory system to avoid the overhead of data and model transfers. In particular, we allocate shared memory accessible in both Python and C/C++ environments. Data needed by AI models is directly retrieved into this shared memory, which allows the Python environment to access and extract data without additional copying overhead. Similarly, model updates are performed directly in the shared memory, enabling immediate access without additional model loading. Further, we introduce a pipeline technique to ensure that data retrieval can be performed concurrently with other computationally intensive operations, such as model training [12, 48, 68]. To facilitate this, we organize the shared buffer as a circular buffer. For example, a background thread is employed to regularly check the availability of cache space and load batches of data into the cache, which ensures a continuous supply of data necessary for model training.

We shall next describe the current implementation of AI-friendly operators in NeurDB. To ensure that users can conveniently invoke these operators, we implement them based on user-defined functions (UDFs), allowing operators to be called with SQL statements. For example, let us consider the model slicing operator. When activated, the UDF for model slicing performs four key tasks. First, it identifies the prediction target and determines the appropriate general model. Second, it customizes the model via the proposed SQL-aware dynamic model slicing technique based on the *filter* conditions in the query and then caches the sliced model in the shared memory. Third, it retrieves relevant data according to the filter conditions using the server programming interface (SPI) and stores the selected data in shared memory. Finally, it performs the model inference with the data and outputs the results. The development of NeurDB has unveiled as many questions as it has resolved. Further improvement in performance, scalability, and functionalities remain the development goals of NeurDB. We shall report new results in the near future and release the code at appropriate time [74].

8 Conclusion

New applications are often catalysts for driving technological innovations. With the immense potential of AI on all aspects of data-centric applications, data systems must now seamlessly support AI- and data-driven analytics. Naturally, AI-powered applications require next-generation data systems to efficiently support AI applications and fully leverage AI technologies. In this paper, we present the conceptual and architectural overview of our proposed AI-powered autonomous data system, called NeurDB. We also outline the innovative techniques developed for NeurDB, and envision its future trajectory in system implementation and evolution.

References

1 Wei Wang, Meihui Zhang, Gang Chen, H. V. Jagadish, Beng Chin Ooi, and Kian-Lee Tan. Database meets deep learning: Challenges and opportunities. *SIGMOD Rec.*, 45(2):17–22, 2016.

2 Andrew Pavlo and Matthew Aslett. What's really new with newsql? *SIGMOD Rec.*, 45(2):45–55, 2016.

3 Jim Gray and Andreas Reuter. *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann, 1993.

4 Michael L. Brodie. Future intelligent information systems: AI and database technologies working together. In *AAAI*, pages 844–845. AAAI Press / The MIT Press, 1988.

5 Michael R. Anderson and Michael J. Cafarella. Input selection for fast feature engineering. In *ICDE*, pages 577–588. IEEE Computer Society, 2016.

6 Shaofeng Cai, Gang Chen, Beng Chin Ooi, and Jinyang Gao. Model slicing for supporting complex analytics with elastic inference cost and resource constraints. *Proc. VLDB Endow.*, 13(2):86–99, 2019.

7 Jeffrey M. Rzeszotarski and Aniket Kittur. Kinetica: naturalistic multi-touch data visualization. In *CHI*, pages 897–906. ACM, 2014.

8 Lin Ma, William Zhang, Jie Jiao, Wuwen Wang, Matthew Butrovich, Wan Shen Lim, Prashanth Menon, and Andrew Pavlo. MB2: decomposed behavior modeling for self-driving database management systems. In *SIGMOD Conference*, pages 1248–1261. ACM, 2021.

9 Chonho Lee, Zhaojing Luo, Kee Yuan Ngiam, Meihui Zhang, Kaiping Zheng, Gang Chen, Beng Chin Ooi, and James Wei Luen Yip. Big healthcare data analytics: Challenges and applications. In *Handbook of Large-Scale Distributed Computing in Smart Healthcare*, pages 11–41. Springer, 2017.

10 Fei Xiao, Yuncheng Wu, Meihui Zhang, Gang Chen, and Beng Chin Ooi. MINT: detecting fraudulent behaviors from time-series relational data. *Proc. VLDB Endow.*, 16(12):3610–3623, 2023.

11 Apache SINGA. https://singa.apache.org/, 2024.

12 Naili Xing, Sai Ho Yeung, Chenghao Cai, Teck Khim Ng, Wei Wang, Kaiyuan Yang, Nan Yang, Meihui Zhang, Gang Chen, and Beng Chin Ooi. SINGA-Easy: An easy-to-use framework for multimodal analysis. In *ACM Multimedia*, pages 1293–1302. ACM, 2021.

13 Peng Jia, Shaofeng Cai, Beng Chin Ooi, Pinghui Wang, and Yiyuan Xiong. Robust and transferable log-based anomaly detection. *Proc. ACM Manag. Data*, 1(1):64:1–64:26, 2023.

14 Jim Gray, Surajit Chaudhuri, Adam Bosworth, Andrew Layman, Don Reichart, Murali Venkatrao, Frank Pellow, and Hamid Pirahesh. Data Cube: A relational aggregation operator generalizing group-by, cross-tab, and sub totals. *Data Min. Knowl. Discov.*, 1(1):29–53, 1997.

15 Wei Wang, Jinyang Gao, Meihui Zhang, Sheng Wang, Gang Chen, Teck Khim Ng, Beng Chin Ooi, Jie Shao, and Moaz Reyad. Rafiki: machine learning as an analytics service system. *Proc. VLDB Endow.*, 12(2):128–140, 2018.

16 Daniel Crankshaw, Xin Wang, Giulio Zhou, Michael J. Franklin, Joseph E. Gonzalez, and Ion Stoica. Clipper: A low-latency online prediction serving system. In *NSDI*, pages 613–627. USENIX Association, 2017.

17 Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778. IEEE Computer Society, 2016.

18 Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. In *CVPR*, pages 8697–8710. Computer Vision Foundation / IEEE Computer Society, 2018.

19 Yatong Wang, Yuncheng Wu, Xincheng Chen, Gang Feng, and Beng Chin Ooi. Incentive-aware decentralized data collaboration. *Proc. ACM Manag. Data*, 1(2):158:1–158:27, 2023.

20 Yuncheng Wu, Naili Xing, Gang Chen, Tien Tuan Anh Dinh, Zhaojing Luo, Beng Chin Ooi, Xiaokui Xiao, and Meihui Zhang. Falcon: A privacy-preserving and interpretable vertical federated learning system. *Proc. VLDB Endow.*, 16(10):2471–2484, 2023.

21 Yuncheng Wu, Shaofeng Cai, Xiaokui Xiao, Gang Chen, and Beng Chin Ooi. Privacy preserving vertical federated learning for tree-based models. *Proc. VLDB Endow.*, 13(11):2090–2103, 2020.

22 Sheng Wang, Tien Tuan Anh Dinh, Qian Lin, Zhongle Xie, Meihui Zhang, Qingchao Cai, Gang Chen, Beng Chin Ooi, and Pingcheng Ruan. ForkBase: An efficient storage engine for blockchain and forkable applications. *Proc. VLDB Endow.*, 11(10):1137–1150, 2018.

23 Kaiping Zheng, Shaofeng Cai, Horng Ruey Chua, Wei Wang, Kee Yuan Ngiam, and Beng Chin Ooi. TRACER: A framework for facilitating accurate and interpretable analytics for high stakes applications. In *SIGMOD Conference*, pages 1747–1763. ACM, 2020.

24 Qingpeng Cai, Kaiping Zheng, Beng Chin Ooi, Wei Wang, and Chang Yao. ELDA: learning explicit dual-interactions for healthcare analytics. In *ICDE*, pages 393–406. IEEE, 2022.

25 Kaiping Zheng, Gang Chen, Melanie Herschel, Kee Yuan Ngiam, Beng Chin Ooi, and Jinyang Gao. PACE: learning effective task decomposition for human-in-the-loop healthcare delivery. In *SIGMOD Conference*, pages 2156–2168. ACM, 2021.

26 Zhaojing Luo, Sai Ho Yeung, Meihui Zhang, Kaiping Zheng, Lei Zhu, Gang Chen, Feiyi Fan, Qian Lin, Kee Yuan Ngiam, and Beng Chin Ooi. MLCask: Efficient management of component evolution in collaborative data analytics pipelines. In *ICDE*, pages 1655–1666. IEEE, 2021.

27 Haotian Gao, Cong Yue, Tien Tuan Anh Dinh, Zhiyong Huang, and Beng Chin Ooi. Enabling secure and efficient data analytics pipeline evolution with trusted execution environment. *Proc. VLDB Endow.*, 16(10):2485–2498, 2023.

28 Matei Zaharia, Omar Khattab, Lingjiao Chen, Jared Quincy Davis, Heather Miller, Chris Potts, James Zou, Michael Carbin, Jonathan Frankle, Naveen Rao, and Ali Ghodsi. The shift from models to compound ai systems. https://bair.berkeley.edu/blog/2024/02/18/compound-ai-systems/, 2024.

29 Maxim Vidgof, Stefan Bachhofner, and Jan Mendling. Large language models for business process management: Opportunities and challenges. In *BPM (Forum)*, volume 490 of *Lecture Notes in Business Information Processing*, pages 107–123. Springer, 2023.

30 IP8Value. https://webapp.ip8value.com/, 2024.

31 Supun Nakandala, Yuhao Zhang, and Arun Kumar. Cerebro: A data system for optimized deep learning model selection. *Proc. VLDB Endow.*, 13(11):2159–2173, 2020.

32 Hanna Mazzawi, Xavi Gonzalvo, Aleks Kracun, Prashant Sridhar, Niranjan Subrahmanya, Ignacio López-Moreno, Hyun-Jin Park, and Patrick Violette. Improving keyword spotting and language identification via neural architecture search at scale. In *INTERSPEECH*, pages 1278–1282. ISCA, 2019.

33 Mohamed S. Abdelfattah, Abhinav Mehrotra, Lukasz Dudziak, and Nicholas Donald Lane. Zero-cost proxies for lightweight NAS. In *ICLR*. OpenReview.net, 2021.

34 Yao Shu, Zhongxiang Dai, Zhaoxuan Wu, and Bryan Kian Hsiang Low. Unifying and boosting gradient-based training-free neural architecture search. In *NeurIPS*, 2022.

35 Colin White, Arber Zela, Robin Ru, Yang Liu, and Frank Hutter. How powerful are performance predictors in neural architecture search? In *NeurIPS*, pages 28454–28469, 2021.

36 Chengliang Zhang, Minchen Yu, Wei Wang, and Feng Yan. MArk: Exploiting cloud services for cost-effective, slo-aware machine learning inference serving. In *USENIX Annual Technical Conference*, pages 1049–1062. USENIX Association, 2019.

37 Michael Armbrust, Kristal Curtis, Tim Kraska, Armando Fox, Michael J. Franklin, and David A. Patterson. PIQL: success-tolerant query processing in the cloud. *Proc. VLDB Endow.*, 5(3):181–192, 2011.

38 Naili Xing, Shaofeng Cai, Zhaojing Luo, Beng Chin Ooi, and Jian Pei. Anytime neural architecture search on tabular data. *CoRR*, abs/2403.10318, 2024.

39 Hidenori Tanaka, Daniel Kunin, Daniel L. K. Yamins, and Surya Ganguli. Pruning neural networks without any data by iteratively conserving synaptic flow. In *NeurIPS*, 2020.

40 Joe Mellor, Jack Turner, Amos J. Storkey, and Elliot J. Crowley. Neural architecture search without training. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pages 7588–7598. PMLR, 2021.

41 Naili Xing, Shaofeng Cai, Gang Chen, Zhaojing Luo, Beng Chin Ooi, and Jian Pei. Database native model selection: Harnessing deep neural networks in database systems. *Proc. VLDB Endow.*, 17(5):1020–1033, 2024.

42 Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. In *ICLR (Poster)*. OpenReview.net, 2018.

43 Xiangnan He and Tat-Seng Chua. Neural factorization machines for sparse predictive analytics. In *SIGIR*, pages 355–364. ACM, 2017.

44 Mahmoud Abo Khamis, Hung Q. Ngo, XuanLong Nguyen, Dan Olteanu, and Maximilian Schleich. Learning models over relational data using sparse tensors and functional dependencies. *ACM Trans. Database Syst.*, 45(2):7:1–7:66, 2020.

45 Lingze Zeng, Naili Xing, Shaofeng Cai, Gang Chen, Beng Chin Ooi, Jian Pei, and Yuncheng Wu. Powering in-database dynamic model slicing for structured data analytics. *CoRR*, abs/2405.00568, 2024.

46 Kaiping Zheng, Shaofeng Cai, Horng Ruey Chua, Melanie Herschel, Meihui Zhang, and Beng Chin Ooi. DyHealth: Making neural networks dynamic for effective healthcare analytics. *Proc. VLDB Endow.*, 15(12):3445–3458, 2022.

47 Wei Wang, Beng Chin Ooi, Xiaoyan Yang, Dongxiang Zhang, and Yueting Zhuang. Effective multi-modal retrieval based on stacked auto-encoders. *Proc. VLDB Endow.*, 7(8):649–660, 2014.

48 Wei Wang, Gang Chen, Tien Tuan Anh Dinh, Jinyang Gao, Beng Chin Ooi, Kian-Lee Tan, and Sheng Wang. SINGA: putting deep learning in the hands of multimedia users. In *ACM Multimedia*, pages 25–34. ACM, 2015.

49 Michael Hammer and Arvola Chan. Index selection in a self-adaptive data base management system. In *SIGMOD Conference*, pages 1–8. ACM, 1976.

50 Goetz Graefe and David J. DeWitt. The EXODUS optimizer generator. In *SIGMOD Conference*, pages 160–172. ACM Press, 1987.

51 Chetan Gupta, Abhay Mehta, and Umeshwar Dayal. PQR: predicting query execution times for autonomous workload management. In *ICAC*, pages 13–22. IEEE Computer Society, 2008.

52 Zhanhao Zhao, Hexiang Pan, Gang Chen, Xiaoyong Du, Wei Lu, and Beng Chin Ooi. VeriTxn: Verifiable transactions for cloud-native databases with storage disaggregation. *Proc. ACM Manag. Data*, 1(4):270:1–270:27, 2023.

53 Elisa Bertino and Beng Chin Ooi. The indispensability of dispensable indexes. *IEEE Trans. Knowl. Data Eng.*, 11(1):17–27, 1999.

54 H. V. Jagadish, Beng Chin Ooi, Kian-Lee Tan, Cui Yu, and Rui Zhang. iDistance: an adaptive B$^+$-tree based indexing method for nearest neighbor search. *ACM Trans. Database Syst.*, 30(2):364–397, 2005.

55 Yannis Theodoridis, Emmanuel Stefanakis, and Timos K. Sellis. Efficient cost models for spatial queries using r-trees. *IEEE Trans. Knowl. Data Eng.*, 12(1):19–32, 2000.

56 Avi Silberschatz, Henry F. Korth, and S. Sudarshan. *Database System Concepts, Seventh Edition*. McGraw-Hill Book Company, 2020.

57 Beng Chin Ooi, HweeHwa Pang, Hao Wang, Limsoon Wong, and Cui Yu. Fast filter-and-refine algorithms for subsequence selection. In *IDEAS*, pages 243–255. IEEE Computer Society, 2002.

58 Rong Zhu, Wei Chen, Bolin Ding, Xingguang Chen, Andreas Pfadler, Ziniu Wu, and Jingren Zhou. Lero: A learning-to-rank query optimizer. *Proc. VLDB Endow.*, 16(6):1466–1479, 2023.

59 Ryan Marcus, Parimarjan Negi, Hongzi Mao, Nesime Tatbul, Mohammad Alizadeh, and Tim Kraska. Bao: Making learned query optimization practical. In *SIGMOD Conference*, pages 1275–1288. ACM, 2021.

60 Zongheng Yang, Wei-Lin Chiang, Sifei Luan, Gautam Mittal, Michael Luo, and Ion Stoica. Balsa: Learning a query optimizer

without expert demonstrations. In *SIGMOD Conference*, pages 931–944. ACM, 2022.

61 Lyric Doshi, Vincent Zhuang, Gaurav Jain, Ryan Marcus, Haoyu Huang, Deniz Altinbüken, Eugene Brevdo, and Campbell Fraser. Kepler: Robust learning for parametric query optimization. *Proc. ACM Manag. Data*, 1(1):109:1–109:25, 2023.

62 Ryan Marcus, Parimarjan Negi, Hongzi Mao, Chi Zhang, Mohammad Alizadeh, Tim Kraska, Olga Papaemmanouil, and Nesime Tatbul. Neo: A learned query optimizer. *Proc. VLDB Endow.*, 12(11):1705–1718, 2019.

63 Mengdi Xu, Yuchen Lu, Yikang Shen, Shun Zhang, Ding Zhao, and Chuang Gan. Hyper-decision transformer for efficient online policy adaptation. In *ICLR*. OpenReview.net, 2023.

64 Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision Transformer: Reinforcement learning via sequence modeling. In *NeurIPS*, pages 15084–15097, 2021.

65 Jia-Chen Wang, Ding Ding, Huan Wang, Conrad Christensen, Zhaoguo Wang, Haibo Chen, and Jinyang Li. Polyjuice: High-performance transactions via learned concurrency control. In *OSDI*, pages 198–216. USENIX Association, 2021.

66 Chunzhi Su, Natacha Crooks, Cong Ding, Lorenzo Alvisi, and Chao Xie. Bringing modular concurrency control to the next level. In *SIGMOD Conference*, pages 283–297. ACM, 2017.

67 Dixin Tang and Aaron J. Elmore. Toward coordination-free and reconfigurable mixed concurrency control. In *USENIX Annual Technical Conference*, pages 809–822. USENIX Association, 2018.

68 Beng Chin Ooi, Kian-Lee Tan, Sheng Wang, Wei Wang, Qingchao Cai, Gang Chen, Jinyang Gao, Zhaojing Luo, Anthony K. H. Tung, Yuan Wang, Zhongle Xie, Meihui Zhang, and Kaiping Zheng. SINGA: A distributed deep learning platform. In *ACM Multimedia*, pages 685–688. ACM, 2015.

69 Yizheng Zhu, Yuncheng Wu, Zhaojing Luo, Beng Chin Ooi, and Xiaokui Xiao. Secure and verifiable data collaboration with low-cost zero-knowledge proofs. *Proc. VLDB Endow.*, 2024.

70 Ergute Bao, Dawei Gao, Xiaokui Xiao, and Yaliang Li. Communication efficient and differentially private logistic regression under the distributed setting. In *KDD*, pages 69–79. ACM, 2023.

71 Haipei Sun, Xiaokui Xiao, Issa Khalil, Yin Yang, Zhan Qin, Wendy Hui Wang, and Ting Yu. Analyzing subgraph statistics from extended local views with decentralized differential privacy. In *CCS*, pages 703–717. ACM, 2019.

72 Cong Yue, Tien Tuan Anh Dinh, Zhongle Xie, Meihui Zhang, Gang Chen, Beng Chin Ooi, and Xiaokui Xiao. GlassDB: An efficient verifiable ledger database system through transparency. *Proc. VLDB Endow.*, 16(6):1359–1371, 2023.

73 Arvind Arasu, Ken Eguro, Raghav Kaushik, Donald Kossmann, Pingfan Meng, Vineet Pandey, and Ravi Ramamurthy. Concerto: A high concurrency key-value store with integrity. In *SIGMOD Conference*, pages 251–266. ACM, 2017.

74 NeurDB. https://neurdb.com, 2024.