RIB: Robust Learning-based Index Benefit Estimation

SIFAN CHEN, Fudan University, China
CHENNING WU, Fudan University, China
YINAN JING*, Fudan University, China
WENTAO WU, Microsoft Research, USA
ZHENYING HE, Fudan University, China
KAI ZHANG, Fudan University, China
X. SEAN WANG, Fudan University, China

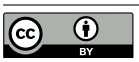
Index benefit estimation is arguably the most important step in index tuning. Existing index tuners developed for commercial and open-source database systems typically leverage the "what if" API for this purpose, which relies on query optimizer's cost estimation module and can be inaccurate due to various reasons such as cardinality estimation errors. Recent work has proposed learning-based index benefit estimators to replace the what-if API. Although learned index benefit estimators show better accuracy, they require large amounts of query execution telemetry as training data, which often contain noisy or conflicting labels that can perplex the trained ML model. There are two types of such noisy labels: *epistemic noise*, which are caused by the limitations of query plan encodings employed by existing learned estimators and *aleatoric noise*, which are caused by the inherent dynamicity of the query execution environment due to unpredictable runtime factors such as buffer pool utilization, resource contention, and sometimes adaptive query execution. In this paper, we propose **RIB** to mitigate the impact of noisy labels and therefore improve the robustness of learned index benefit estimators. **RIB** introduces two new technologies to address the challenges imposed by epistemic and aleatoric label noises: (1) a context-aware encoder based on bidirectional graph neural network (GNN) and (2) a probabilistic prediction model based on fully parameterized quantile regression (FPQR). Compared to existing work, the GNN-based encoder captures more contextual information about index-optimizable operations, such as structural changes in query plans before and after utilizing indexes, thereby reducing the chance of epistemic noise. Moreover, unlike existing work that uses a point estimate for index benefit estimation, the FPQR-based predictor considers the entire distribution of likely index benefits and provides more robust estimates by aggregating over all quantiles of the distribution, thereby reducing the impact of aleatoric noise. Extensive experiments on top of multiple benchmarks demonstrate that **RIB** significantly outperforms state-of-the-art index benefit estimators in terms of both estimation accuracy and end-to-end index recommendation quality.

CCS Concepts: • **Information systems → Query optimization**; **Autonomous database administration**.

Additional Key Words and Phrases: Index Tuning, Index Benefit Estimation

*Corresponding author.

Authors' Contact Information: Sifan Chen, sfchen23@m.fudan.edu.cn, Fudan University, College of Computer Science and Artificial Intelligence, Shanghai, China; Chenning Wu, wucn23@m.fudan.edu.cn, Fudan University, College of Computer Science and Artificial Intelligence, Shanghai, China; Yinan Jing, jingyn@fudan.edu.cn, Fudan University, College of Computer Science and Artificial Intelligence, Shanghai, China; Wentao Wu, wentao.wu@microsoft.com, Microsoft Research, Redmond, Washington, USA; Zhenying He, zhenying@fudan.edu.cn, Fudan University, College of Computer Science and Artificial Intelligence, Shanghai, China; Kai Zhang, zhangk@fudan.edu.cn, Fudan University, College of Computer Science and Artificial Intelligence, Shanghai, China; X. Sean Wang, xywangCS@fudan.edu.cn, Fudan University, College of Computer Science and Artificial Intelligence, Shanghai, China.



ACM 2836-6573/2026/2-ART77
https://doi.org/10.1145/3786691

ACM Reference Format:
Sifan Chen, Chenning Wu, Yinan Jing, Wentao Wu, Zhenying He, Kai Zhang, and X. Sean Wang. 2026. RIB: Robust Learning-based Index Benefit Estimation. *Proc. ACM Manag. Data* 4, 1 (SIGMOD), Article 77 (February 2026), 26 pages. https://doi.org/10.1145/3786691

1 Introduction

Index tuning [27] is the process of selecting an optimal subset of indexes for a given database workload, with the goal of minimizing query execution cost while satisfying constraints such as limited storage space. Index benefit estimation is arguably the most critical component of index tuning, which quantifies the performance improvement that a given set of indexes can bring to a workload and therefore guides the index tuning process to prioritize the most impactful indexes. Without accurate index benefit estimation, the index tuner may choose suboptimal or even harmful indexes, leading to poor query performance or even performance regression [42].

Existing index tuners/advisors developed for commercial and open-source database systems typically leverage the "what if" API for index benefit estimation [5], which takes dependency on the query optimizer's cost estimation module. Specifically, the what-if API allows the index tuner to specify candidate indexes as "hypothetical indexes" without materializing them, by creating only the metadata and statistics of the candidate indexes and making them visible to the query optimizer. The query optimizer can then consider these hypothetical indexes as available during its exploration of access paths and query plans. Albeit an elegant idea, the estimated index benefit based on the what-if API can be inaccurate due to various reasons such as cardinality estimation errors.

To improve the estimation accuracy of index benefit, recent work has introduced learning-based index benefit estimators [12, 32, 42, 47]. Learned index benefit estimators are trained on historical query execution telemetry data that are based on materialized rather than hypothetical indexes. As a result, learned estimators are typically more accurate than using the what-if API, which often results in better quality of the recommended indexes when integrated with the index tuner. However, the training data often contain noisy or conflicting labels that can perplex the trained ML model. There are two types of such noisy labels: (1) *epistemic noise*, which are caused by the limitations of query plan encodings employed by existing learned estimators and (2) *aleatoric noise*, which are caused by the inherent dynamicity of the query execution environment due to unpredictable runtime factors such as buffer pool utilization. Below we discuss these two types of label noises in more detail.

Epistemic Noise. The presence of indexes can significantly influence the query execution plan produced by the query optimizer. As was pointed out by [47], indexes can influence query execution in two ways: (1) *change of access path*, where the availability of new indexes modifies only the data access paths without affecting the logical structure of the query plan (e.g., by replacing a sequential table scan with an index scan); and (2) *change of plan structure*, where the availability of new indexes results in query plan restructuring (e.g., by choosing a different join order). However, existing learned index benefit estimators are largely limited to handling only the first case. They either focus on predicting operator-level costs [42, 47], estimating the impact of indexes by modeling localized changes; or attempt to represent the overall query plan by encoding only the "index-optimizable" operations [32], whose data sources contain columns that are indexed. *Failure of capturing the variability in index impact across different queries can lead to conflicting training data labels that confuse the learned index benefit estimators.* We illustrate this problem with the following example.

Example 1. As shown in Figure 1, creating an index on `orders(o_totalprice)` affects Query 1 and Query 2 differently. Query 1 experiences only a *change in access path* (from `Seq Scan` to

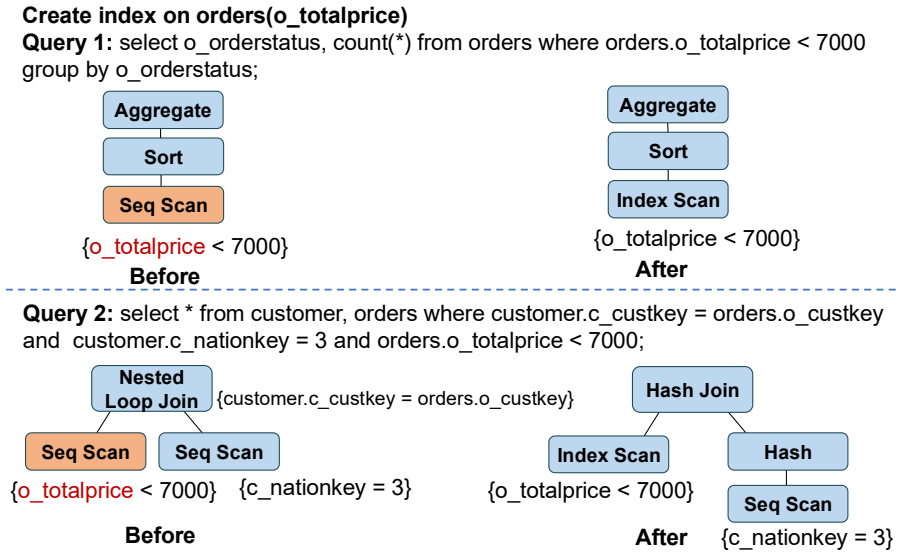


Fig. 1. Illustration of epistemic noise.

`Index Scan`), whereas Query 2 undergoes both an *access-path* change and a *structural change* of the query plan, where the query optimizer chooses a different join operator (from `NestedLoop Join` to `Hash Join`). However, such structural differences are not captured by the state-of-the-art learned index benefit estimator **LIB** [32]. Since the original plans of both queries contain a `Seq Scan` on the indexed column `o_totalprice`, they are both "*index-optimizable operators*" and **LIB** encodes them identically. In contrast, the join operators are not recognized as index-optimizable because the join key `o_custkey` does not match the indexed column `o_totalprice`. Consequently, the encodings of Query 1 and Query 2 by **LIB** are *identical* despite their distinct index benefits, producing conflicting labels that mislead **LIB**.

Aleatoric Noise. Index benefit is inherently uncertain in real-world scenarios, as it is based on query execution time that can vary significantly due to complex and unpredictable runtime factors such as buffer pool utilization, resource contention, and sometimes adaptive query execution. As a result, the benefit of a given index configuration for a specific query should be regarded as a random variable conditioned on the query and index features rather than a constant value. However, existing learned index benefit estimators typically provide only a point estimate, which is sensitive to outliers caused by fluctuation in query execution time.

Example 2. To empirically demonstrate the stochastic nature of index benefit, we executed a randomly selected pair of query and index configuration 30 times without clearing buffer pool or fix CPU binding. This setup mirrors realistic database environments where caching and OS scheduling introduce runtime variability. As Figure 2 shows, the observed index benefit exhibits substantial variance, indicating that it should be modeled as a conditional random variable rather than a deterministic value. This observation motivates the adoption of distribution-aware approaches, such as quantile regression, for more robust index benefit estimation.

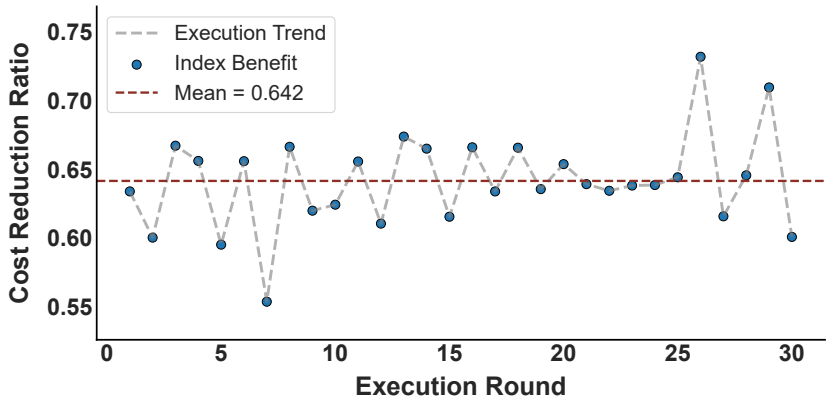


Fig. 2. Illustration of aleatoric noise.

Summary of Contributions. To tackle the aforementioned challenges caused by noisy labels, we propose **RIB**, a robust learning-based index benefit estimator that includes (1) a novel combination of a bidirectional GNN encoder to mitigate *epistemic noise* caused by misleading query plan encodings and (2) a fully parameterized quantile regression (FPQR) predictor to handle *aleatoric noise* arising from stochastic query execution behavior. Although the underlying ML architectures themselves are not newly invented, our innovation lies in how they are integrated and tailored to address long-standing robustness challenges in index benefit estimation.

(GNN-based Plan Encoder) We design a context-aware encoder that employs a bidirectional GNN to represent the original query plan without relying on the what-if API. The GNN-based plan encoder allows index-optimizable operators to capture operator-level characteristics as well as contextual information from neighboring and dependent operators in the entire query plan. Compared to existing work that focuses on encoding *local* information of an index-optimizable operator, the GNN-based encoder enriches the feature representation by augmenting it with more *global* information, such as structural changes in the query plan before and after using indexes, thus reducing the chance of epistemic noise.

(FPQR-based Predictor) Unlike existing work that uses a point estimate for index benefit estimation, we develop an FPQR-based predictor that considers the entire distribution of likely index benefits and provides more robust estimates by aggregating over all quantiles of the distribution, thereby reducing the impact of aleatoric noise. Unlike classical parameter estimation methods that estimate the conditional mean of the response variable, quantile regression [46] estimates the conditional quantiles of the data. This flexibility allows quantile regression to model a complex distribution without making any a priori assumptions about the underlying distribution of the data, resulting in more accurate and robust estimations. We further design several quantile combination strategies to capture various distributional aspects.

Extensive experiments on top of multiple benchmarks demonstrate that **RIB** significantly outperforms state-of-the-art index benefit estimators in terms of both estimation accuracy and end-to-end index recommendation quality. For example, when tuning the **TPC-DS** and **JOB** (shorthand for the "join order benchmark") workloads, **RIB** achieves around 2× improvement of the total workload execution time compared to baseline approaches.

Table 1. Summary of notations

Symbol	Description	Section
$B(q, I)$	Cost reduction (index benefit) of index configuration I on query q	2.1
$C(q\|I)$	Execution cost of query q under index configuration I	2.1
$Cr(q, I)$	Cost reduction ratio (normalized index benefit) of index configuration I on query q	2.1
r_o	Feature representation for index-optimizable operator o	4.2
r_q	Unified query plan representation for query q	4.2
τ	Quantile fractions proposed by the *fraction proposal network*	5.1
$\mathbf{x}$	Input query plan representation to the prediction model	5.2
θ	Learnable parameters of the *quantile value network*	5.2
$F_\theta^{-1}(\tau\|\mathbf{x})$	Quantile value at fraction τ for input representation $\mathbf{x}$	5.2
$\text{Cost}(W\|I)$	Actual execution cost of workload W under index configuration I	6.1

2 Problem Formulation

We present an overview of the basic concepts and formulate the problem of index benefit estimation. Table 1 summarizes the notation used throughout this paper.

Workload. A workload $W = \{q_1, q_2, \ldots, q_n\}$ consists of n queries executed on relational database, where each query $q_i (q_i \in W, 1 \leq i \leq n)$ retrieves a specific set of attributes from one or more tables.

Secondary Index. A secondary index is an auxiliary data structure that optimizes query execution through accelerated data retrieval operations. A (secondary) index is characterized by an ordered set of attributes from one table.

Index Configuration. An index configuration I contains one or multiple indexes.

Index Candidates. Index candidates are potential indexes that are considered and evaluated by index tuning. Typically, they contain only columns that appear together in at least one query.

Index Tuning. Index tuning (or index selection) is about identifying an optimal index configuration I such that the execution cost of the workload under certain constraints, e.g., a limited storage budget B or the maximum number of indexes N.

The problem of index benefit estimation is to accurately and efficiently quantify the impact of a given index configuration on query execution cost without materializing the index configuration and physically executing the query.

Definition 1 (Index Benefit)**.** For a query q and an index configuration I, we define the benefit of I as the query cost reduction with and without the index configuration I:

$$B(q, I) = C(q|\emptyset) - C(q|I), \tag{1}$$

where $C(q|\emptyset)$ denotes the execution cost of the query q without any index and $C(q|I)$ denotes the execution cost of the query q with the index configuration I.

Problem Statement (Index Benefit Estimation) Given a workload $W = \{q_1, q_2, \ldots, q_n\}$ with n queries and an index configuration I, the goal of index benefit estimation is to minimize the prediction error between the predicted benefit $\hat{B}(q_i, I)$ and the actual benefit $B(q_i.I)$ for each query $q_i \in W$, formulated as

$$\min_I \sum\nolimits_{q_i \in W} \left| B(q_i, I) - \hat{B}(q_i, I) \right|. \tag{2}$$

However, since query execution cost can vary significantly, ranging from a few milliseconds to thousands of seconds, we follow the previous work [32] and estimate the cost reduction ratio, i.e., the normalized index benefit, defined as

$$Cr(q_i, I) = \frac{B(q_i, I)}{C(q_i|\emptyset)} = \frac{C(q_i|\emptyset) - C(q_i|I)}{C(q_i|\emptyset)}. \tag{3}$$

Accordingly, our model aims to minimize the squared difference between the estimated and actual cost reduction ratio, namely,

$$\min_{I} \sum\nolimits_{q_i \in W} \left(Cr(q_i, I) - \hat{Cr}(q_i, I)\right)^2. \tag{4}$$

Discussion. Our work in this paper targets OLAP (i.e., analytical) workloads and thus does not consider data and/or schema evolution. Extending our proposed framework to cover OLTP (i.e., transactional) scenarios would require incorporating index maintenance overhead caused by data and/or schema updates. In such settings, the estimation of index benefit should consider (1) query performance gain and (2) index maintenance penalty, which can be expressed as additional cost/benefit terms in our formulation and thus incorporated into our framework in an orthogonal manner. Maintenance-aware index selection has been studied in previous work [26], where index benefit is expressed as a linear combination of (1) and (2). However, this is perhaps not applicable in more complex or dynamic database query execution environments, where the relationship between (1) and (2) can be more complicated. Developing a more general approach therefore remains a challenge and we leave it for future work.

3 Overview of RIB

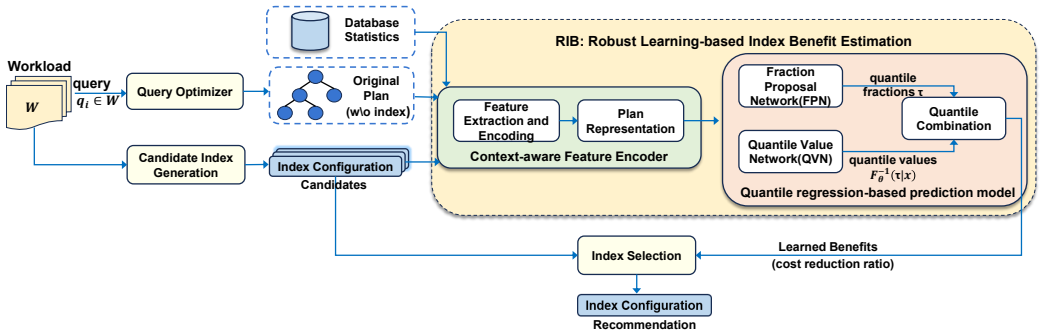


Fig. 3. An overview of the individual software components and end-to-end workflow of **RIB**.

Design Goal. In real-world database systems, query execution time exhibits two salient characteristics:

- (*Fluctuation*) Given the same query and index configuration, the query execution time can fluctuate due to transient factors such as buffer pool utilization, resource contention, and sometimes adaptive query execution.
- (*Long-tailed Distribution*) Many queries are short-running while a few incur disproportionately long delays.

Conventional regression models used by existing work ignore such inherent variance in query execution and come up with a single *point estimate* of the index benefit, which is sensitive to (the

inevitable) noisy data and outliers presented in the query execution telemetry collected for model training.

Our basic idea is to leverage *quantile regression* [46] to model index benefit as a *conditional random variable* (conditioned on the given query and index configuration), rather than viewing it as a constant value. Specifically, instead of learning an average benefit value for each pair of query and index configuration, we train the quantile regression model to predict multiple quantiles (e.g., 50th, 90th) of the index benefit distribution. This allows us to capture the inherent fluctuation in index benefit caused by unpredictable runtime factors and provide a distribution-aware, risk-sensitive estimation that is more accurate and robust under dynamic query execution conditions. For example, the model may output a 90th percentile benefit of 0.35, indicating that in 90% of the cases, the index yields no more than a 35% improvement of query performance.

Moreover, as shown in Figure 1, when encoding the original query plan, considering only index-optimizable operations may result in different query plans being encoded into the same feature vector, despite their significantly different index benefits. This can lead to conflicting labels for training data points with the same feature encodings, perplexing the model and negatively affecting its prediction accuracy. To address this issue, we employ a bidirectional graph neural network to capture more contextual information of index-optimizable operators in a given query plan.

As shown in Figure 3, the entire framework of **RIB** consists of two key modules: (1) a context-aware feature encoder and (2) a prediction model. The context-aware feature encoder is composed of two components: (1a) feature extraction and encoding, and (1b) plan representation. The prediction model consists of three components: (2a) fraction proposal network, (2b) quantile value network, and (2c) quantile combination. In the following, we present a brief overview of each of these components.

Feature Extraction and Encoding. Without relying on the what-if API, we extract *schema-agnostic* features (such as operator types, estimated costs, and output cardinalities) from the original query plan. We then encode the operators in the query plan into fixed-length feature vectors, organized as a tree based on the structure of the plan. The tree is passed to the *Plan Representation* component.

Plan Representation. We use a bidirectional graph neural network (GNN) to capture the contextual information of the operators, generating informative embeddings for the operators. The embeddings of index-optimizable operators are then selected and combined with database statistics and index information to compute the *augmented* representations. We employ an attention mechanism to capture index interactions and operator correlations, aggregating them into a single query plan representation.

Prediction Model. The prediction model takes the query plan representation as input and outputs the cost reduction ratio to measure the index benefit. Specifically, the *Fraction Proposal Network* generates quantile fractions of index benefit. The *Quantile Value Network* maps these quantile fractions to corresponding benefit values. The *Quantile Combination* employs several quantile combination strategies, such as *Approximate Expectation* and *Risk-Aware Estimations*, to provide more accurate and robust index benefit estimation.

Workflow. For offline training, given a workload, we first apply one of the common index tuning methods (e.g., DTA [3]) to generate candidate index configurations. We then materialize these index configurations and execute the workload to collect the original query plans and compute the cost reduction ratios as training data, which is encoded into sets of vectors by *Feature Extraction and Encoding*. Each training data point is stored as a tuple <operator vectors, cost reduction ratio>. We then use the cost reduction ratio as the target label to jointly train the *Plan Representation* and *Prediction Model*. We present the details of model training in Section 5.4.

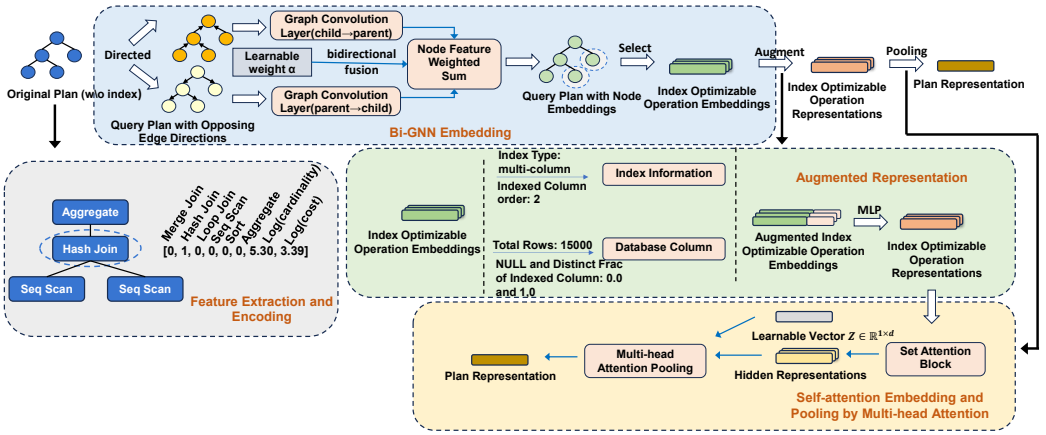


Fig. 4. An overview of the context-aware query plan feature encoder.

For online prediction, given a query in the input workload and an index configuration enumerated by the index tuner, **RIB** first utilizes *Feature Extraction and Encoding* to encode the operators of the original query plan into a set of vectors organized as a tree, without invoking the what-if API. **RIB** then applies *Plan Representation* to aggregate the vectors into a single vector to represent the query plan. The query representation is fed into the *Prediction Model* to estimate the cost reduction ratio. Notably, **RIB** can be used to replace the index benefit estimation module (which relies on the what-if API) of a typical index tuning software architecture, without modifying the other modules such as candidate index generation and index configuration enumeration. Therefore, **RIB** can easily be integrated with existing index tuners.

4 Context-aware Plan Encoder

We present the details of the GNN-based context-aware query plan encoder in this section.

4.1 Feature Extraction and Encoding

Given an original query plan p for a given query q, we first extract schema-agnostic features, which are then used to encode each operator in the plan. The operator encodings are organized into a tree based on plan structure. The operator encoding consists of two primary parts: (1) operation encoding and (2) statistics encoding.

Operation Encoding. Following [32], given candidate indexes, we focus on *index-optimizable operations*, i.e., operators that directly access indexed columns, including `Scan`, `Join`, `Sort`, and `Aggregation`. The physical operators are one-hot encoded.

Statistics Encoding. The output cardinality and estimated cost of a relational operator in the query plan are concatenated to form the statistics encoding. These values are typically generated by the query optimizer's cost model based on the database statistics. Although these estimates are not always precise, ML models can still leverage them in the training process [49].

Operator Node Encoding. The operation encoding and statistics encoding of a relational operator node in the query plan are concatenated to form the operator encoding. The node encodings are then organized into a tree based on the plan structure. The tree is further passed to *Plan Representation* to obtain the representation of the entire query plan.

4.2 Plan Representation

A key challenge of plan representation lies in how to model the potential impact of candidate indexes on the entire query plan, rather than just encoding the index-optimizable operators themselves. To address this challenge, we adopt a bidirectional graph neural network (Bi-GNN) that leverages bidirectional message passing between neighbors to effectively capture near-global structural information within the query plan. We further enhance the plan representation with signals specific to index tuning. Following [32], we also apply an attention-based mechanism to capture interactions among multiple candidate indexes.

Bi-GNN Embedding. As shown in Figure 1, relying solely on index-optimizable operators can lead to label conflicts, as it ignores their surrounding contextual dependencies. To capture structural characteristics of query plans, the directed graph neural network (GNN) [36] that represents a physical query plan as a directed graph is well suited, where nodes denote relational operators and edges capture data dependencies. However, a unidirectional GNN suffers from limited information propagation [2]: information flows only upward from children to parents, preventing parent nodes from learning from their ancestors and leading to structural information loss. Consequently, the information from the leaf nodes must traverse as many GNN layers as the depth of the tree to reach the root, which significantly increases the computational cost.

To alleviate these limitations, we employ a *bidirectional* graph neural network (Bi-GNN) that performs message passing in both upward and downward directions within each layer. Instead of simply adding an additional propagation direction, our Bi-GNN enables each node to simultaneously receive information from both its parent and child nodes in the same layer. As each parent and child further exchange information with their own neighbors subsequently, global structural information can be effectively propagated throughout the entire query plan tree within only two layers [29].

In each Bi-GNN layer, we decompose the given query plan tree into two directed tree graphs with opposing edge directions. Each direction applies the graph convolution on the tree graph constructed either from child to parent or from parent to child. The node embeddings from both directions are then fused via a learnable parameter to capture the contextual information of the nodes. This design allows the encoder to achieve *near-global context aggregation* with *minimum number of GNN layers*, improving both representational completeness and efficiency in training and inference.

Augmented Representation with Index and Statistics. Given the index-optimizable operator embeddings from Bi-GNN, we augment the representation vector by incorporating index information and database statistics. The index information includes the index type and the order of the indexed columns. Specifically, we use a one-hot vector to represent the index type (i.e., single-column or multi-column index). For multi-column index, an integer (starting from 1) denotes the order of the indexed columns, while for single-column index, we pad the order field with zeros. The database statistics include the number of rows, the fraction of `NULL`s, and the ratio of distinct values for the indexed column. We then concatenate the index-optimizable operator embeddings with these extra vectors to form the augmented representation vector. Finally, these augmented representation vectors are fed into a fully connected neural network layer with `ReLU` activation, transforming them into dense representations to capture high-level features. We adopted the previous work [32] to learn the query plan representation from the set of index-optimizable representation vectors.

Attention-based Pooling. Previous research [31] demonstrates that index interactions significantly impact index benefit estimation. Consider an index configuration with three indexes IX1, IX2, and IX3. The combined use of IX1 and IX2 can result in higher cost savings than using them separately, which is a typical example of positive index interaction. However, while combining

IX1 and IX2 can replace IX3, it may result in lower benefit than only using IX3. Furthermore, the presence of IX1 and IX2 can prevent IX3 from being included in the optimal query plan, which gives a case of negative index interaction. Therefore, accurate index benefit estimation requires accounting for index interactions. For this purpose, we employ a self-attention mechanism to process the encodings of index optimizable operators, effectively capturing pairwise and higher-order interactions between indexes [20]. Since the contributions of different index-optimizable operators to the overall index benefit may vary significantly, we use a multi-head attention pooling mechanism to aggregate the operator-level node encodings.

Following the previous work [32], we employ a self-attention mechanism to encode interactions among index-optimizable operator representations within the set, thus capturing index interactions. Given the set of N index-optimizable operator representations $C = \left\{\mathbf{r}_{o_1}, \mathbf{r}_{o_2}, \ldots, \mathbf{r}_{o_n}\right\} \in \mathbb{R}^{n \times d}$, we adopt the encoder block of the now well-known Transformer [38] architecture (without its positional encoding) as *set attention block* (SAB) [20]. We stack multiple SABs to learn the representation $\mathbf{r}'_{o_i}$, which contains information about pairwise or high-order interactions among $\mathbf{r}_{o_i} \in C, i \in [1, n]$.

To adaptively aggregate operator representations into a unified query plan representation $\mathbf{r}_q$, we apply *pooling by multi-head attention* (PMA) [20], where a learnable query vector $\mathbf{Z} \in \mathbb{R}^{1 \times d}$ attends all elements of $C' = \{\mathbf{r}'_{o_i}\}$. This allows the model to weigh the contributions of different index-optimizable operators based on their relevance to the final index benefit.

5 Prediction Model

As discussed in Section 1, index benefit, typically defined as the reduction in query execution time before and after index creation, is inherently uncertain due to variability of query performance introduced by runtime factors such as buffer pool utilization, resource contention, and sometimes adaptive query execution. Therefore, an index benefit estimation model should account for this data uncertainty [48] rather than producing a single-point prediction.

Mainstream approaches to quantifying prediction uncertainty fall into two categories: (1) Bayesian inference methods [14] and (2) parameterized distribution models [30]. These approaches are either computationally expensive or unable to capture the skewness of distributions observed in real-world query executions. Quantile regression estimates conditional quantiles without requiring a priori assumptions about the underlying data distribution. This characteristic makes it more suitable for modeling the uncertainty of index benefit. For a continuous and strictly increasing cumulative distribution function (CDF) $F_\theta(y|X = x)$, the quantile function of the response variable y (cost reduction ratio or index benefit in our context), denoted as $Q(\tau)$, is defined as the *inverse* of the CDF [28]:

$$Q(\tau) = F_\theta^{-1}(\tau|X = x) := \inf\left\{y \in \mathbb{R} : F\left(y|X = x\right) \geq \tau\right\}. \tag{5}$$

Unlike existing quantile regression methods that assume fixed quantile fractions and parameterize only the corresponding quantile values, *fully parameterized quantile function* (FPQF) [46] parametrizes *both* the quantile fractions and the corresponding quantile values (i.e., self-adjusting quantile fractions). This approach enables a more accurate approximation of the true distribution, thus enhancing the accuracy and robustness of index benefit estimation.

5.1 Fraction Proposal Network

The shape of the index benefit distribution varies significantly for different original query plans due to runtime variability and workload diversity. To approximate these diverse distributions, we avoid using fixed, uniformly spaced quantile fractions. Instead, as shown in Figure 5, we design a *fraction proposal network* that dynamically generates an adaptive set of quantile fractions, denoted as $\tau_1, \tau_2, \ldots, \tau_{N-1}$, which satisfy $\tau_{i-1} < \tau_i$ with boundary values $\tau_0 = 0$ and $\tau_N = 1$. This approach

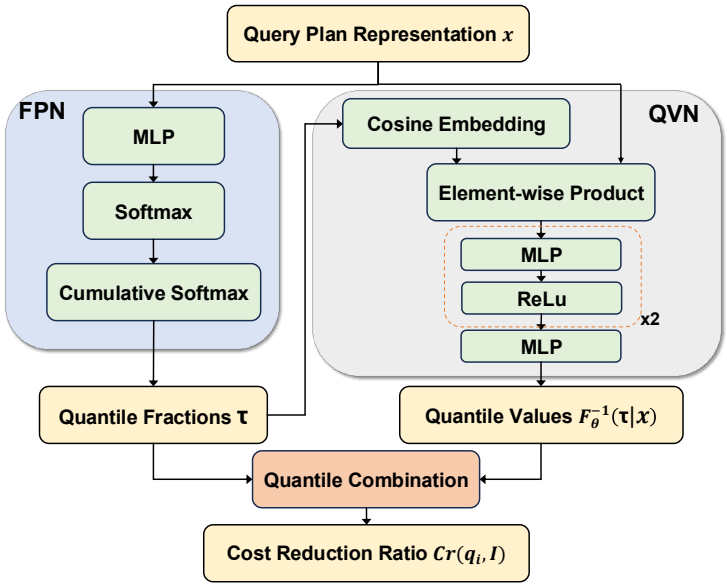


Fig. 5. An overview of the FPQR-based prediction model.

enables the model to allocate more quantiles to regions exhibiting rapid changes in the distribution and fewer quantiles to relatively flat regions, resulting in a more accurate and efficient representation of the distribution.

Technically, the fraction proposal network is a fully connected single-layer neural network that employs the *softmax* activation function. To ensure that the output values are sorted, we adopt a cumulative softmax transformation at the output layer. Formally, let the output of the softmax layer be represented as $p \in \mathbb{R}^N$, where each element satisfies $p_i \in (0, 1)$, $i \in [0, N-1]$, and $\sum_{i=0}^{N-1} p_i = 1$. We then define the quantiles as

$$\tau_i = \sum_{j=0}^{i-1} p_j, \quad i \in [0, N]. \tag{6}$$

By construction, for any $i < j$, it follows that $\tau_i < \tau_j$ while ensuring the boundary conditions $\tau_0 = 0$ and $\tau_N = 1$.

5.2 Quantile Value Network

Given the set τ of quantile fractions generated by the fraction proposal network and a query plan representation $\mathbf{x}$, a *quantile value network* maps the quantile fractions to their corresponding quantile values $F_\theta^{-1}(\tau_i|\mathbf{x}, i \in [0, N])$, which estimate the index benefit at different quantile levels.

As shown in Figure 5, the quantile value network consists of a cosine embedding layer followed by several fully connected layers, applying the ReLU activation function with dropout. The cosine embedding layer computes the embedding $\phi(\tau)$ of the quantile fractions τ as follows. This embedding allows the model to capture the interaction of the learned quantile fractions τ when predicting the quantile values. As shown in previous work [9], simple linear embeddings of τ are insufficient to achieve good performance:

$$\phi_j(\tau) := \mathrm{ReLU}\left(\sum_{i=0}^{n-1} \cos(i\pi\tau) w_{ij} + b_j\right). \tag{7}$$

Next, we compute the element-wise product between the quantile embedding $\phi(\tau)$ and the query plan representation $\mathbf{x}$ as the *fused representation* of the quantile fractions and the query plan. This fused representation is then passed through fully connected layers to output the estimated quantile values:

$$F_\theta^{-1}(\tau_i|\mathbf{x}, i \in [0, N]) = \mathsf{ReLU}((\phi(\tau) \odot \mathbf{x}) \cdot \mathbf{W} + \mathbf{b}). \tag{8}$$

Although the output quantile values (i.e., the cost reduction ratios) should ideally lie between 0 and 1, we do not apply a sigmoid function to restrict the output, as was done in previous work [32]. Instead, we introduce a regularization term in the training loss function to guide the model to maintain outputs within a reasonable range. For rare cases where the outputs fall outside this range, we apply truncation by clipping the outputs to fit in the range $[0, 1]$.

5.3 Quantile Combination

Based on the estimated quantiles of the index benefit, we design several strategies to combine the quantiles to obtain the most suitable estimate of the index benefit, as shown in Figure 5.

Approximate Expectation. The most intuitive strategy is to approximate the expected value of the index benefit by calculating the mean of the estimated quantile values:

$$F_\theta^{-1}(\tau|\mathbf{x}) = \sum\nolimits_{i=0}^{N-1} (\tau_{i+1} - \tau_i)\, F_\theta^{-1}(\frac{\tau_{i+1} + \tau_i}{2}|\mathbf{x}). \tag{9}$$

Since the estimated quantiles are discrete, we employ a weighted interpolation method to approximate the distribution of index benefit. The expected index benefit is estimated by weighting the midpoints between adjacent quantiles. The weight of each midpoint is determined by the difference between adjacent quantiles, which dictates its contribution to the expected value.

Theoretically, this method provides a more comprehensive characterization of the distribution of the index benefit, leading to a more accurate and robust estimate. In the subsequent experimental section, we will consistently apply this method and validate its superiority in terms of accuracy and robustness, compared to point estimation methods, through extensive experiments.

Risk-aware Estimation. Creating or deleting indexes can sometimes significantly degrade query performance in production database systems. To ensure system stability, database administrators (DBAs) may want to choose indexes that offer limited benefit while having minimal impact on the overall database workload. Therefore, to accommodate this scenario, it may be more appropriate to place more emphasis on the quantiles that represent *underestimated* index benefits. One possible approach is to increase the weights of lower quantiles, namely,

$$F_\theta^{-1}(\tau|\mathbf{x}) = \sum\nolimits_{i=0}^{N-1} w_i\, (\tau_{i+1} - \tau_i)\, F_\theta^{-1}(\frac{\tau_{i+1} + \tau_i}{2}|\mathbf{x}), \tag{10}$$

where w_i is the weight of the quantile τ_i, defined as $w_i = \frac{1}{i+1}$. In addition to this simple weighting strategy, it is definitely possible to consider more sophisticated weighting functions or manual adjustments based on specific application-driven needs. However, our experiments indicate that the *approximate expectation* aggregation strategy already performs well. Therefore, we leave further study of this *risk-aware estimation* strategy for future work.

5.4 Model Training

Training Dataset. We generate training data by materializing indexes and executing workload queries. Each training data point comprises the original query plan (encoded into vectors by *Feature Extraction and Encoding* in **RIB**), the index configuration, and the actual cost reduction ratio (as the label). Specifically, given an input workload, we first follow the rule-based candidate index generation method outlined in [50] to construct potential index candidates by enumerating all permutations of columns appearing in query predicates (including join predicates) and extending

them with columns referenced in the SELECT, GROUP BY, and ORDER BY clauses. We then employ an existing index tuner to enumerate all index configurations up to size K (we set $K = 3$ in our experiments). We collect all index configurations enumerated in the tuning process. For each collected index configuration, we materialize the indexes and execute the workload queries, recording the execution time of each query and computing the corresponding cost reduction ratio. To ensure the relevance of training data, we retain only query–index pairs that lead to changes in the query execution plan.

Training Loss. When modeling the index benefit distribution via quantile regression, the goal is to ensure that the predicted quantile function closely approximates the true benefit distribution. Following prior work [13], we employ the *1-Wasserstein* distance as the primary loss to minimize the deviation between the estimated and the true quantile functions. Compared with divergence-based objectives such as *KL divergence* [19], the *1-Wasserstein* distance jointly captures both distributional differences and magnitude shifts, offering a more comprehensive measure of distributional discrepancy. As the true quantile function is unknown, the output of *quantile value network* serves as its approximation.

Moreover, to prevent the distribution from degenerating into a deterministic one, we introduce an *entropy* regularization term on the softmax outputs of *fraction proposal network*. The total loss for *fraction proposal network* thus combines the Wasserstein term with the entropy regularizer, controlled by a small coefficient λ. For *quantile value network*, we adopt the *Huber-smoothed quantile loss* [10, 16], which extends the standard asymmetric quantile loss by applying a quadratic penalty within a small error threshold and a linear one otherwise. This smoothing yields more stable gradients and better nonlinear approximation behavior.

Together, these objectives enable the two networks to jointly learn a continuous and robust quantile representation of index benefit without explicit assumptions about the underlying distribution.

Our quantile regression model does not require repeatedly collecting labels for the same query–index pair to capture aleatoric noises, but it captures the empirical variability of index benefit across diverse queries and indexes. For example, an index may be beneficial for one query but cause regression for another query with a similar feature encoding. **RIB** learns a *conditional quantile function* $F_\theta^{-1}(\tau \mid \mathbf{x})$ from a large set of input samples, where each sample $(\mathbf{x}_i, y_i)$ represents an observed index benefit with specific query and index characteristics. By minimizing the *pinball loss* over the entire training dataset, the model estimates a function that maps any input $\mathbf{x}$ to its corresponding conditional quantile of $P(y \mid \mathbf{x})$:

$$\hat{y}^*(\mathbf{x}) = \arg\min_{\hat{y}\in\mathbb{R}} \ \mathbb{E}_{y\sim P(y|\mathbf{x})}[L_\tau(y, \hat{y})] = F_\theta^{-1}(\tau \mid \mathbf{x}). \tag{11}$$

6 Experimental Evaluation

We conduct experiments on standard benchmarks and real-world workloads to answer the following key questions:

- Compared to state-of-the-art (SOTA) methods, does **RIB** achieve more accurate and robust index benefit estimation, especially for unseen queries and index configurations (Section 6.2.1)?
- When integrated into an index tuner, does **RIB** recommend better indexes and exhibit more robust query performance under varying workloads compared to baselines (Section 6.3)?
- How well do the key components of **RIB** perform (Section 6.4)?

For the last question above, we report results from the ablation study of the context-aware plan encoder and the prediction model. In the rest of this section, we refer to them as the Bi-GNN-based feature encoder and the FPQR-based prediction model, respectively, to reflect their underlying ML technologies.

Table 2. Statistics of the experimental benchmarks.

Benchmark	Dataset	Relations	Attributes	Templates	Size(GB)
TPC-H	Synthetic uniform	8	61	22	10
TPC-DS	Synthetic skew	24	429	99	10
JOB	Real-world	21	108	33	6

6.1 Experimental Setup

Datasets and Benchmarks. We study the performance of **RIB** on three publicly available benchmarks with different scales and characteristics: **TPC-H**, **TPC-DS**, and the Join Order Benchmark (**JOB**). Table 2 presents the detailed information of the benchmark workloads. The **TPC-H** benchmark is relatively simple, whereas the **TPC-DS** benchmark represents more complex OLAP scenarios. The **JOB** benchmark consists of real-world IMDB data with more realistic data dependencies and cardinality distributions.

Following the previous work [18], we exclude certain templates because the execution time of queries from these templates is orders of magnitude longer than that of queries from the other templates. Without this exclusion, queries from these templates would dominate the overall workload cost, thereby reducing the complexity of the index selection problem (because selecting an index that optimizes queries from these templates can easily outperform indexes designed for queries from the other templates).

Environment Configurations. All the experiments are performed on a workstation configured with 2× 24 Core Intel® Xeon® Gold 5318Y CPU at 2.10GHz and 1.0TB main memory, running PostgreSQL 13.18 under Ubuntu 20.04.6 LTS.

Evaluation Metrics. We use the following metrics to evaluate the performance of **RIB** in different facets:

- **Q-Error.** Following the previous work [32], we use Q-error to assess the prediction accuracy of **RIB**, defined as

$$Q\text{-error} = \max\left(\frac{Cr_{\text{actual}}}{\hat{Cr}_{\text{predicted}}}, \frac{\hat{Cr}_{\text{predicted}}}{Cr_{\text{actual}}}\right). \tag{12}$$

- **Cost Improvement(%).** We use the *relative cost improvement* of a given workload W to evaluate the quality of the index configuration I recommended by **RIB**, defined as

$$\max\{0, 1 - \frac{\text{Cost}(W|I)}{\text{Cost}(W|\emptyset)}\}. \tag{13}$$

  Here, $\text{Cost}(W|\emptyset)$ is the execution cost of the workload W without indexes and $\text{Cost}(W|I)$ is the execution cost of W with the recommended index configuration I.

Baselines. We compare **RIB** with the following SOTA and representative index benefit estimation approaches:

- **PostgreSQL.** We use the estimated cost from the query optimizer of PostgreSQL with default settings.
- **LIB.** Learned Index Benefit (LIB) is an SOTA index benefit estimator proposed in [32]. LIB encodes only index-optimizable operators in the original query plan and utilizes a self-attention mechanism to model index interaction, providing a point estimate of index benefit without considering distributional information.
- **AI-Meets-AI.** AI-Meets-AI (AMA) is proposed in previous work [12], which uses a predefined set of *feature channels*, such as the total estimated cost of each node, to train a classifier that

Table 3. Q-errors under three different train/test data splits: "No Split," "Split by Configuration," and "Split by Query."

		No Split				Split by Configuration				Split by Query			
Dataset		PostgreSQL	AMA	LIB	**RIB**	PostgreSQL	AMA	LIB	**RIB**	PostgreSQL	AMA	LIB	**RIB**
TPC-H	Mean	12.966	1.416	**1.290**	1.305	3.471	1.410	1.349	**1.333**	4.505	2.031	**1.940**	2.042
	90th	7.058	1.971	1.698	**1.654**	6.184	1.984	1.820	**1.768**	4.926	4.165	4.693	**3.961**
	95th	12.788	2.268	1.977	**1.926**	13.126	2.105	2.147	**2.041**	11.352	4.214	4.916	**4.120**
TPC-DS	Mean	5.235	1.371	1.352	**1.305**	4.270	1.382	1.367	**1.337**	4.597	1.641	1.433	**1.389**
	90th	13.891	1.878	1.819	**1.625**	10.018	1.856	1.837	**1.650**	12.785	2.633	2.134	**1.641**
	95th	16.469	2.220	2.190	**1.801**	12.039	2.182	2.224	**1.935**	14.571	2.808	2.673	**1.931**
JOB	Mean	9.541	1.943	1.843	**1.604**	7.540	1.809	1.873	**1.707**	7.265	2.100	2.023	**1.977**
	90th	23.163	3.067	3.029	**2.342**	17.689	3.042	2.877	**2.613**	5.746	3.294	3.250	**2.960**
	95th	30.450	3.537	3.948	**2.890**	26.465	3.631	3.429	**3.297**	16.616	3.556	3.553	**3.477**

compares query plans corresponding to two index configurations and returns the better one. Following [32], we replace the classification layer of AMA by an MLP with two hidden layers for index benefit estimation.

Model Training Setting. We use the mini-batch Adam optimizer for model training. Specifically, the query plan encoder and the quantile value network are trained with the Adam optimizer using an initial learning rate of 0.001. We employ a learning rate scheduler from `torch.optim` to reduce the learning rate by 0.5 when performance does not improve for at least 10 epochs. The fraction proposal network is trained separately using another Adam optimizer with a fixed learning rate of 0.0001. Additionally, we apply dropout and early stopping to prevent overfitting.

6.2 Accuracy of Index Benefit Estimation

We compare the estimation accuracy of **RIB** with baseline methods. We simulate three types of distribution shifts between the training and test data across all benchmarks:

- **No Split.** The training and test data follow the same distribution. This setting evaluates the estimation accuracy of **RIB** under ideal conditions without distributional drift.
- **Split by Configuration.** The index configurations in the training and test data are *disjoint*. This setting evaluates the ability of **RIB** to adapt to new index configurations.
- **Split by Query.** The queries in the training and test data are *disjoint*. This setting assesses the ability of **RIB** to generalize to unseen queries and adapt to workload drift.

For each participating index benefit estimator, we report the mean, the 90th percentile, and the 95th percentile of the Q-error.

6.2.1 Overall Analysis. Table 3 presents the estimation accuracy of each method when estimating the cost reduction ratio of the index configurations. We have the following observations.

RIB outperforms the other index benefit estimators in terms of estimation accuracy for different training/test data splitting scenarios. Specifically, the accuracy of the estimators follows the order: **RIB** > LIB > AMA > PostgreSQL.

Compared to learned estimation models, which adapt to data characteristics by learning from practical experience and dynamically adjusting the weights of cost features, the cost model of the query optimizer suffers from inaccurate assumptions about data characteristics and relies on static weights for cost computation. As a result, PostgreSQL exhibits the worst accuracy.

Among the learned estimation models, AMA does not explicitly encode information related to index benefit when characterizing query plan differences. Its feature extraction method is designed

primarily to identify which of the two query plans is better, making it difficult to learn the magnitude of cost reduction between plans [32], leading to poorer accuracy. In contrast, LIB directly learns the cost reduction magnitude of query plans and explicitly considers index benefit information when encoding query plans, resulting in better accuracy than AMA. However, since LIB only considers index-optimizable operators, different plans can be mapped to the same representation, causing label conflicts (i.e., epistemic noises) that confuse the ML model and thus limit its effectiveness.

RIB mitigates the impact of epistemic noise with a bidirectional GNN-based query plan encoder, which effectively captures the contextual information of index-optimizable operators, increasing the differences between query plans in terms of index benefits and resolving label conflicts. In addition, the prediction model uses FPQR to capture the entire conditional distribution of the index benefit, enabling a more comprehensive characterization. Consequently, **RIB** achieves the highest estimation accuracy.

In most cases, all participating methods achieve lower estimation accuracy on the **JOB** benchmark than the TPC benchmarks. **JOB** is built on top of a real-world dataset, which exhibits more complex data distributions, making it more challenging to estimate index benefit compared to the synthetic TPC datasets.

Compared to the mean Q-error, **RIB** achieves greater reductions in the 90th and 95th percentile Q-errors, indicating a significant decrease in extreme prediction errors. This improvement in terms of tail errors leads to more reliable and stable index benefit estimation, improving the overall robustness of the estimator. In particular, LIB achieves a lower mean Q-error on the **TPC-H** benchmark in most cases. One possible explanation is that LIB provides a point estimate of the index benefit that focuses on the central tendency of the data. In contrast, **RIB** uses quantile regression to capture a more comprehensive benefit distribution; however, it offers limited advantages for the relatively uniform data distribution of **TPC-H**.

Furthermore, **RIB** is more advantageous, in terms of prediction accuracy, with the "Split by Configuration" and "Split by Query" settings than the "No Split" setting. This indicates that **RIB** can effectively generalize to unseen queries and configurations and is more robust in the presence of workload drift.

6.2.2 Query-level Analysis. To further understand the advantage of **RIB** in terms of its estimation accuracy, we conduct a query-level comparative study between **RIB** and LIB. Specifically, we randomly select 10 queries from each benchmark. Figure 6 plots violin charts for the selected queries with the setting "No Split." The vertical span represents the range of Q-errors for each query, with larger values indicating a longer-tailed error distribution. The horizontal span reflects the probability density of the Q-errors, with wider regions indicating a higher data concentration. To highlight the tail effect, we also include the curves of the 95th percentile Q-errors.

Overall, **RIB** exhibits more compact Q-error distributions than LIB (i.e., with flatter violins) for most queries across all benchmarks, suggesting its smaller variance and more stable estimation behavior. The 95th percentile curves further confirm that **RIB** consistently achieves lower 95th percentile Q-errors, implying a substantial reduction in tail effects of estimation errors.

A closer query-level examination reveals that queries within the same benchmark exhibit significant differences in estimation difficulty. In the same benchmark, queries span various complexity levels from simple aggregations to multi-way analytical joins. LIB tends to produce heavy-tailed errors for complex multi-way joins over skewed tables (e.g., queries 5, 60, and 80 of **TPC-DS**, queries 9 and 23 of **JOB**). In contrast, **RIB** effectively reduces these extreme errors. On the other hand, for queries with simpler join conditions or on more uniformly distributed data (e.g., queries 8 and 9 of **TPC-H**), both methods show narrower Q-error ranges.

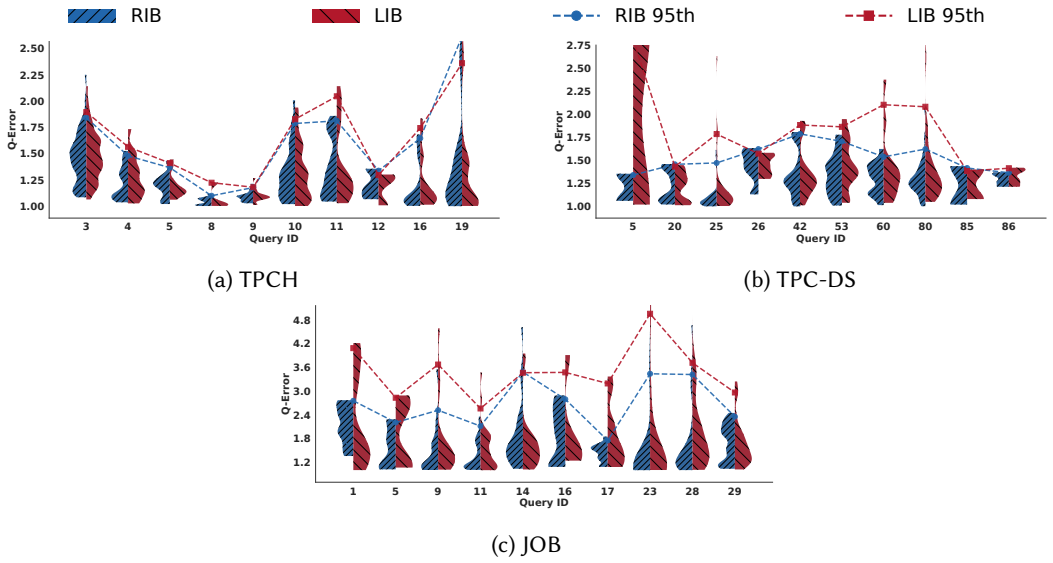


Fig. 6. A query-level comparative study between **RIB** and LIB in terms of their Q-errors for different benchmarks.

We also observe distinct patterns across benchmarks. The gaps between **RIB** and LIB in terms of the 95th percentile Q-errors are substantially larger for **TPC-DS** and **JOB** than for **TPC-H**, where the two methods perform more closely. This difference can be attributed to the complexity of the benchmarks. **TPC-DS** features a larger number of tables and more intricate query structures, while the complexity of **JOB** stems from real-world data properties, including significant data skew and extensive multi-way joins involving 5 to 10 or even more tables. **JOB** exhibits a more pronounced long-tailed distribution of Q-errors, roughly twice as wide as that of **TPC-DS** and **TPC-H**, where **RIB** shows a significant advantage. This is because the robust quantile regression estimator of **RIB** avoids selecting high-potential but high-risk indexes that tend to cause unstable execution behaviors for complex multi-join queries, which we will explain further in Section 6.3.

6.3 Index Recommendation Quality

We evaluate the impact of integrating **RIB** with an SOTA index tuner on the quality of end-to-end index recommendation. For each benchmark, we construct 30 workloads by randomly sampling queries that were excluded from the training data. Each sampled workload consists of 10 query templates for **TPC-H** and **TPC-DS**, and 8 query templates for **JOB**, with 10 query instances per template. Candidate indexes are determined by generating all syntactically relevant indexes. We employ the anytime index tuning algorithm presented in [3] to enumerate candidate index configurations, which are then evaluated using each index benefit estimation method. Finally, we materialize the recommended indexes, execute workload queries, and calculate the relative cost improvement of the workload to assess the quality of the recommended indexes.

6.3.1 End-to-end Analysis. Table 4 reports the end-to-end index recommendation quality in terms of workload execution cost, including total cost savings and average relative improvement. The slight variations in the original runtime (i.e., without indexes) across different methods occur because the baseline workloads were collected at different times in a shared environment where system states

Table 4. Comparison of **RIB** with baseline approaches in terms of end-to-end index recommendation quality.

Method	Cost saving (minutes)			Original runtime (minutes)			Improvement (%)		
	TPC-H	**TPC-DS**	**JOB**	**TPC-H**	**TPC-DS**	**JOB**	**TPC-H**	**TPC-DS**	**JOB**
PostgreSQL	18.53	4.99	13.89	194.35	390.37	87.98	9.2	1.7	16.3
AMA	24.38	12.07	6.36	204.25	397.77	89.04	11.9	3.5	4.5
LIB	24.73	26.79	16.20	209.86	382.26	90.01	12.5	8.1	17.8
RIB	**30.86**	**52.67**	**31.33**	202.13	386.16	90.73	**16.2**	**14.5**	**31.7**

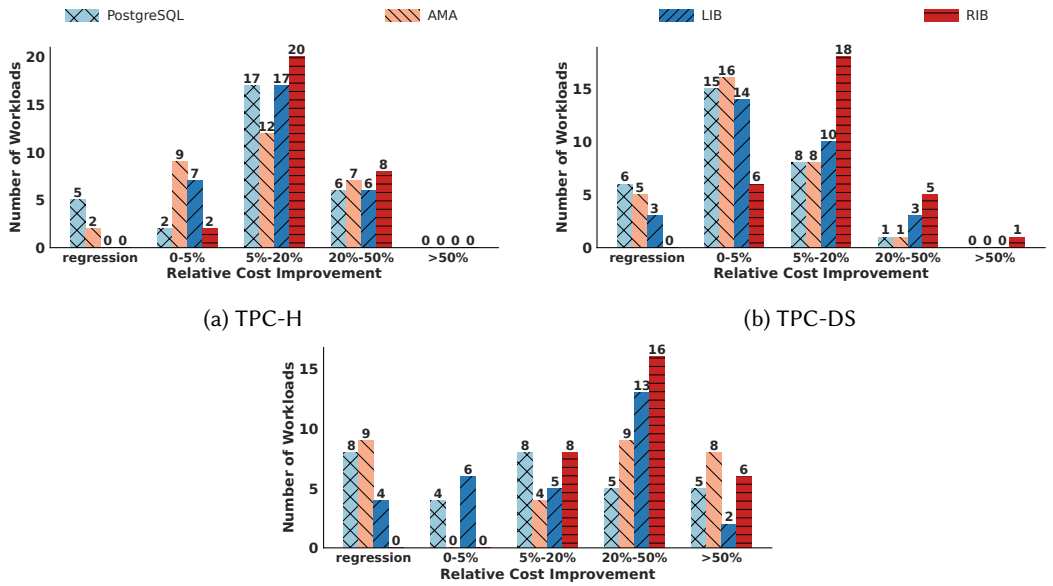
can fluctuate. However, for each method, both the execution time without and with indexes were measured consecutively under similar system conditions, ensuring fair comparison and reliable cost-savings evaluation. We observe that **RIB** achieves the best overall index recommendation quality, surpassing the SOTA index benefit estimator, LIB. On **TPC-H**, the indexes recommended by **RIB** yield approximately 25% and 30% higher total cost reduction and average workload improvement, respectively, compared to LIB. The advantage becomes more pronounced on **TPC-DS** and **JOB**, where **RIB** achieves approximately 2× improvement compared to LIB.

One reason for this is that **RIB** quantifies the index benefit for the workload more accurately than LIB by resolving label conflicts and effectively capturing the distribution characteristics of the index benefit. On the other hand, AMA and PostgreSQL exhibit relatively poor performance compared to the other methods. AMA can identify which index from a pair of indexes is more beneficial for a single query but cannot *quantify* this benefit, making it challenging to evaluate the overall impact of the index on the workload. The cost model used by PostgreSQL frequently introduces index estimation errors due to incorrect assumptions about data characteristics.

6.3.2 Workload-level Analysis. Figure 7 further presents the distribution of workload improvements across benchmarks, grouped by the number of workloads within each improvement range. **RIB** effectively identifies indexes that lead to higher cost improvements across various benchmarks while effectively preventing query performance regression. Both **RIB** and LIB successfully avoid regressions for **TPC-H**, but **RIB** results in the largest number of workloads with cost improvements between 20% and 50%, whereas AMA and PostgreSQL result in several regressions. **RIB** entirely eliminates regressions for **TPC-DS**, while LIB, AMA, and PostgreSQL experience varying degrees of performance degradation. Moreover, **RIB** results in more workloads with $20-50\%$ cost reductions and is the only method with an improvement of more than 50% for a single workload. A similar pattern is observed for **JOB**, where **RIB** is more robust, preventing regressions that are common with AMA and PostgreSQL. Again, **RIB** results in the largest number of workloads with an improvement of 20% to 50%.

A key reason behind the above observations is that **RIB** comprehensively models the conditional distribution of index benefit by jointly learning quantile fractions and their corresponding values. Since each quantile prediction channel captures signals from different parts of the index benefit distribution, their aggregation effectively reduces output variance, resulting in more robust index benefit estimation.

Compared with the synthetic **TPC** benchmarks, the real-world IMDB dataset underlying **JOB** features stronger attribute correlations and heavier data skew, making accurate index benefit estimation more challenging and increasing the risk of query performance regression. Although **RIB** produces slightly fewer workloads with over 50% improvement than AMA, AMA suffers from the highest number of workload performance regressions and performs significantly worse than other baselines overall (see Table 4). These results highlight the importance and necessity



(a) TPC-H (b) TPC-DS

(c) JOB

Fig. 7. Analysis of workload execution time improvements for different benchmarks. Each plot shows the number of workloads falling into the corresponding ranges of percentage improvements.

of mitigating query performance regressions in index benefit estimation and demonstrate the robustness and superiority of **RIB** in this aspect.

6.3.3 Case Study. To better understand the superiority of **RIB** in terms of the index recommendation quality, we conduct a case study with representative workloads from two benchmarks: (1) **TPC-H**, which is relatively simple with uniform data distribution, and (2) **JOB**, which contains more complex queries.

On the workload 12 from **TPC-H**, baseline methods exhibit significant regressions (up to 50%) on the query 7 due to recommendation of the index `lineitem(l_suppkey,l_shipdate)`, which results in a query plan with range scans on the column `l_shipdate` over a time interval that is less selective. It further leads to a poor join order selected by the query optimizer. In contrast, **RIB** deliberately avoids this index and selects a different index `customer(c_nationkey,c_custkey)`, which exploits the highly selective filter on the column `c_nationkey`. With the GNN-based modeling of complex plan-index interactions, **RIB** can better speculate how an index influences the decisions made by the query optimizer on its access path selection, such as join order and table access strategy, leading to more accurate index benefit estimation and reduced query performance regressions.

The performance regressions caused by favoring high-potential but high-risk indexes on large tables become more pronounced for **JOB**. We again choose the workload 12 as an example. The query 19 of **JOB**, an analytical query that joins ten tables with sparse filtering predicates, suffers significant regression with baseline approaches. The regression is caused by indexes on `cast_info(movie_id)`. Since `movie_id` appears in multiple join predicates of the query, the query optimizer is misled by choosing plans with a `NestedLoop` join that requires excessive amount of index lookups with random IOs. As a result, the execution time increases from 4 seconds to 20 seconds. A better strategy is to build indexes on small and highly selective tables (e.g., `aka_name(person_id)` as

Table 5. Evaluation of index tuning with different constraints, such as the number of indexes allowed and storage budget.

Constraints	Index creation time (min)			Execution time (min)			Index storage (MB)		
	AMA	LIB	RIB	AMA	LIB	RIB	AMA	LIB	RIB
# of indexes ($K = 2$)	5.96	3.04	4.80	82.68	73.81	59.40	686	507	334
Storage bound ($B = 1000$MB)	5.82	2.32	1.98	78.08	69.96	57.29	566	178	185

recommended by **RIB**), which effectively restricts joins between large tables. The objective function of quantile regression used by **RIB** reduces the bias toward high-potential but high-risk indexes on large tables and instead emphasizes indexes on more selective and stable attributes, effectively lowering the risk of regression.

Across all benchmarks, **RIB** consistently achieves both higher quantitative gains and greater qualitative stability in index recommendation. On simple benchmarks such as **TPC-H**, it mitigates epistemic noise by modeling complete plan-index interactions through Bi-GNN, which enables accurate prediction of structural plan changes and prevents indexes that lead to risky join orders. On more complex benchmarks such as **JOB**, it effectively handles aleatoric uncertainty through distribution-aware quantile regression. Together, our case studies demonstrate that **RIB** can deliver more robust index tuning results.

6.3.4 Index Tuning with Constraints. Table 5 presents the index creation time and overall query execution time under different tuning constraints (e.g., the number of indexes or storage budget allowed) for **JOB**. Across both constraint settings, **RIB** consistently demonstrates superior end-to-end performance compared to baselines, achieving the lowest overall query execution time *as well as* the lowest index creation time. This advantage can be attributed to the robust selection strategy used by **RIB**, which avoids choosing indexes with high uncertainty and thus reduces the overhead associated with creating risky indexes that can lead to query performance regression. Robust index selection is particularly valuable for online index tuning, where index creation overhead is less affordable and stability is critical for production workloads.

Table 6. Q-errors of **RIB** and its variants.

Dataset		LIB	**RIB**-w/o-FPQR	**RIB**-w/o-BiGNN	**RIB**
TPC-H	Mean	1.639	1.587	1.661	**1.538**
	90th	1.731	1.593	1.717	**1.565**
	95th	2.444	2.434	2.317	**2.258**
TPC-DS	Mean	1.346	1.337	1.339	**1.306**
	90th	1.810	1.713	1.709	**1.622**
	95th	2.109	2.013	1.910	**1.808**
JOB	Mean	1.866	1.760	1.767	**1.662**
	90th	3.220	2.851	2.773	**2.536**
	95th	3.970	3.561	3.551	**3.213**

6.4 Ablation Study

We conduct a detailed analysis of the contribution from each of the two major components, namely, (1) the Bi-GNN-based encoder and (2) the FPQR-based predictor, to index benefit estimation. We compare the performance of **RIB** and its following *variants*:

- **RIB-w/o-BiGNN**, which replaces the Bi-GNN-based feature encoder with the encoder from LIB. The encoder from LIB focuses only on index-optimizable operators without considering their contextual information and models index interactions using an attention-based mechanism.
- **RIB-w/o-FPQR**, which replaces the FPQR-based prediction model with the prediction model from LIB. The prediction model from LIB is composed of a two-layer MLP with the ReLU activator and a sigmoid function as the output layer.

Table 6 reports the estimation accuracy of **RIB** and its variants. **RIB** consistently achieves the best over all performance, compared to its variants. This validates the joint effectiveness of the Bi-GNN feature encoder and the FPQR-based predictor. We further observe that the variant **RIB-w/o-BiGNN** outperforms **RIB-w/o-FPQR** on complex, long-tailed workloads (e.g., with smaller 90th and 95th Q-errors for **TPC-DS** and **JOB**), indicating that the FPQR-based predictor plays a more critical role in addressing long-tail effects. The FPQR-based predictor combines multiple quantile predictions of the index benefit. During this process, each quantile prediction channel captures different aspects of the index benefit distribution, which helps to reduce the variance of the output, leading to a more accurate and robust estimation.

On the other hand, the Bi-GNN encoder (i.e., variant **RIB-w/o-FPQR**) yields greater improvement on simpler benchmarks like **TPC-H** or in terms of mean accuracy, as it mitigates label conflicts in the training data by incorporating the contextual information of index-optimizable operators. This enables structurally distinct query plans to be represented differently, thereby improving estimation accuracy and overall model robustness. Both variants surpass the state-of-the-art index benefit estimator LIB, indicating that each component contributes substantial and complementary value to the overall framework.

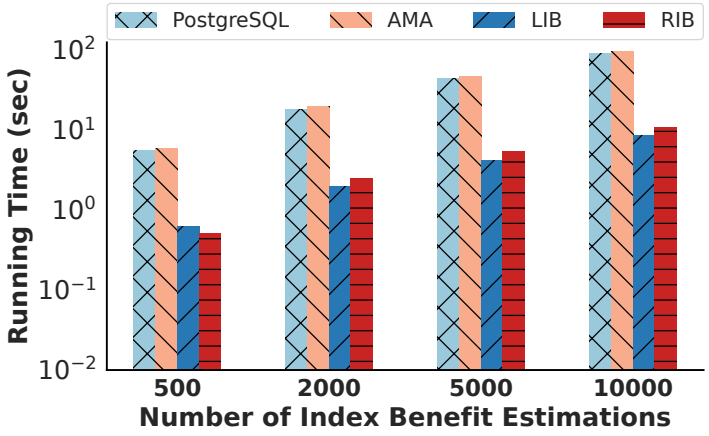


Fig. 8. Comparison of inference time in index tuning.

Table 7. Comparison of offline training overhead.

Training time (min)			Collection time (min)		Storage (MB)	
RIB	LIB	AMA	Query	Index	AVG	MAX
90	52	1.5	717.83	48.91	473	2831

6.5 Efficiency of Model Training and Inference

Online Inference Time. We compare the total inference time of **RIB** during index tuning with other baseline methods. We randomly select a fixed number (from 500 to 10,000) of query and index configuration pairs and compute the average inference time spent on index benefit estimation. All experiments are conducted without GPU acceleration for fairness. As shown in Figure 8, the estimation efficiency of different methods follows the following order: LIB $\approx$ **RIB** $\gg$ AMA $\approx$ PostgreSQL. **RIB** maintains high estimation efficiency and scales well with the number of estimations, matching the most efficient baseline, LIB. Both **RIB** and LIB completely replace the time-consuming what-if calls with learning-based estimators and enable batch evaluation, therefore achieving strong scalability on large workloads. In contrast, PostgreSQL and AMA invoke a what-if call for each pair of query and index configuration, incurring significant time overhead on large workloads [24, 39, 40].

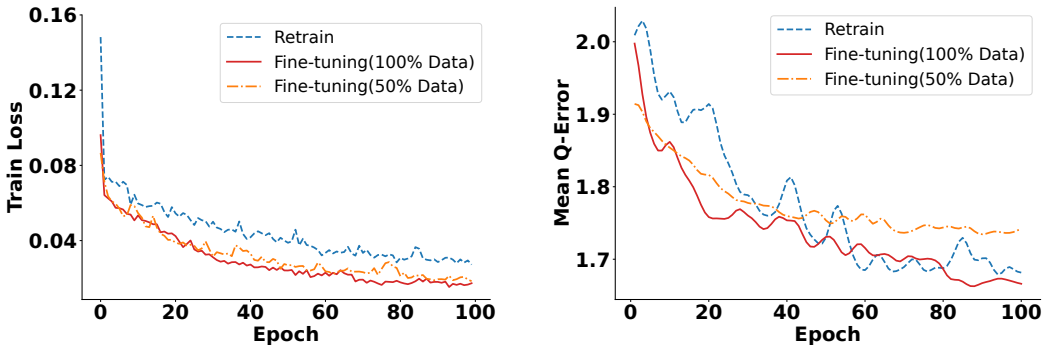
Offline Training Overhead. Table 7 presents the offline training overhead of the models for **TPC-DS**, including the data collection time, storage consumption, and model training time. All learning-based models are trained on the same 6,781 query–index pairs, therefore having identical data collection time and storage consumption. We observe that the data collection time dominates the model training time, accounting for the majority of the total time cost. **RIB** requires more model training time compared to LIB and AMA due to its higher model complexity. AMA relies on what-if calls to obtain the impact of indexes on queries, thereby avoiding complex attention-based modeling and having shorter training time, at the cost of significantly longer inference time (see Figure 8). Since all models are trained offline, the extra training time of **RIB** is well justified by its improved accuracy, faster inference, and superior end-to-end index recommendation quality.

6.6 Adaptation to Database Drifts

While our primary focus of **RIB** in this paper has been on OLAP workloads without data or schema drift, **RIB** can generalize to new data distribution or schema change with suitable adaptation. We further design experiments to evaluate its generalization capability, providing supporting evidence for its applicability to more complex workloads and more dynamic real-world scenarios.

To examine the adaptability of **RIB** to new database environments, we evaluate its cross-schema transfer learning capability from **TPC-DS** (with scaling factor 10) to **JOB**, which differ substantially in both database schema and data distribution. We consider three adaptation strategies: (1) training the model from scratch using only **JOB** data, (2) fine-tuning the pretrained model (initialized with parameters learned from **TPC-DS**) with 50% of **JOB** data, and (3) fine-tuning the same pretrained model with 100% of **JOB** data. The latter two strategies simulate realistic scenarios where only partial or full data from a new database is available for incremental model updating, allowing us to assess whether lightweight fine-tuning can effectively replace full retraining. All models are trained for 100 epochs under identical optimization configurations.

Figure 9 shows the convergence behavior and prediction accuracy of different adaptation strategies. As illustrated in Figure 9a, both fine-tuning strategies converge faster and achieve lower training losses than full retraining. In Figure 9b, the mean Q-errors of the fine-tuned models



(a) Training loss convergence

(b) Mean Q-error on test set

Fig. 9. Comparison of different adaptation strategies.

decrease more rapidly and stabilize at a similar level compared to full retraining. In particular, fine-tuning with 100% of the new data produces the best overall accuracy, while fine-tuning with 50% of the new data remains competitive despite a slightly higher Q-error, demonstrating that **RIB** can effectively adapt to a new database using only a fraction of the training data.

These findings demonstrate that the context-aware encoder and the distribution-aware predictor of **RIB** facilitate efficient knowledge transfer across heterogeneous database schema and data distribution. Rather than relearning schema-specific patterns from scratch, **RIB** effectively reuses generalizable representations learned from **TPC-DS** and adapts them to **JOB** with lightweight fine-tuning. This suggests that **RIB** can accommodate schema evolution, data distribution shift, and data scale variation without the need for prohibitively expensive full retraining.

7 Related Work

Index Tuning. Index tuning has been a long-standing research focus in the database community, with the aim of selecting an optimal subset of indexes to minimize the execution cost of a given workload under constraints such as a storage budget. The workflow of index tuning typically includes *candidate index generation*, *index configuration enumeration* or *index selection*, and *index benefit estimation* [34, 45, 50]. Candidate index generation [33, 35] identifies promising index candidates based on the given workload by looking for *indexable columns* contained by queries with relevant relational expressions such as filter and join predicates. Index selection [4, 25, 37, 41, 44] iterates over the candidate indexes using a predefined mechanism to select the final configuration of recommended indexes. Index benefit estimation [12, 32, 35, 47] quantifies the impact of a given index configuration on query execution cost, which helps filter out useless candidate indexes and guides the index selection process. Consequently, index benefit estimation is crucial to ensuring that index tuning identifies the most effective indexes for a given workload.

Learning-based Index Benefit Estimation. Learning-based index benefit estimation methods have shown excellent performance in recent years. These methods employ ML models trained using historical query execution telemetry data on top of materialized indexes. Existing learning-based index benefit estimators use either classification or regression models [50]. A classification model [12] predicts which of two given index configurations (and the corresponding query plans) is more effective. Although it can directly minimize the plan comparison error, it cannot quantify index benefit for each individual query, and therefore cannot be used for workload-level index benefit estimation that requires further aggregating query-level index benefit estimates. In contrast,

a regression model [32] can directly predict the query execution cost or the cost reduction ratio given an index configuration, and therefore avoids the problem of aggregating index benefit estimates at the workload level. Among existing state-of-the-art index benefit estimators that leverage regression models, Shi et al. [32] characterize index-optimizable operations in the original query plan and utilize an attention mechanism to model index interaction. Yu et al. [47] focus on predicting operator-level costs, estimating the effect of indexes by modeling localized changes. Both methods did not account for the impact of the indexes on the structural change of the query plan, and they solely relied on a single-point benefit prediction while ignoring the uncertainty arising from the inherent variability in query execution time.

Uncertainty Quantification. Uncertainty quantification is important in machine learning (ML) and has been a key element of ML methodologies [17]. Predictive uncertainty in ML can be classified into two categories: *model uncertainty* and *data uncertainty*. Model uncertainty typically arises from training errors, inadequate model structure, or insufficient knowledge due to unknown samples or limited coverage of the training dataset [11], and can be reduced by providing enough representative training data. Data uncertainty describes the variance of the conditional distribution of the target variable given the input features [21], generally stemming from the inherent stochastic nature of the target variable, such as noise or measurement errors. Many probabilistic deep learning methods have been proposed in the literature for quantifying predictive uncertainty [23]. One category of such methods is Bayesian approaches, such as Bayesian Neural Networks (BNNs) [15, 22] and Monte Carlo Dropout (MCD) [14], which approximate the distribution of model output by sampling from the posterior over model parameters. Another category is parametric distribution models [6, 30], which assume that model output follows predefined parametrized distributions, such as Gaussian or mixture distributions. In contrast, quantile regression (QR) is a nonparametric approach that directly learns multiple quantiles of the conditional distribution without assuming any a priori distribution, resulting in well-calibrated uncertainty estimates [8, 28]. Uncertainty quantification has also been extensively studied in robust query optimization [1, 7, 43].

8 Conclusion

We have proposed **RIB**, a novel learning-based index benefit estimator that is, to the best of our knowledge, the first work to mitigate the impact of label noise presented in the training data collected from historical query execution telemetry and therefore improve the robustness of learning-based index benefit estimation. **RIB** employs two novel technologies, namely, (1) a context-aware query plan feature encoder that utilizes a bidirectional graph neural network to capture more contextual information of index-optimizable operators in the query plan, thus resolving label conflicts caused by insufficient query plan encoding (i.e. epistemic noise); and (2) a fully parameterized quantile regression model to account for the entire conditional distribution (instead of a single point estimate as was done by existing work) of the index benefit, thus characterizing label uncertainties caused by the inherent dynamicity and unpredictability of query execution (i.e. aleatoric noise). Extensive experimental evaluation on top of the TPC-H, TPC-DS, and JOB benchmarks demonstrates the effectiveness and superiority of **RIB** compared to state-of-the-art learned index benefit estimators.

Acknowledgments

We would like to thank all the anonymous reviewers for their insightful comments and suggestions. This work is supported by the National Key R&D Program of China (2024YFC3307800), NSFC (U24A20232, 62072113, and 62272106) and Xiaohongshu–Fudan Collaborative Project (PHT10120241108010).

References

[1] Brian Babcock and Surajit Chaudhuri. 2005. Towards a Robust Query Optimizer: A Principled and Practical Approach. In *SIGMOD*. 119–130.

[2] Baoming Chang, Amin Kamali, and Verena Kantere. 2024. A Novel Technique for Query Plan Representation Based on Graph Neural Nets. In *International Conference on Big Data Analytics and Knowledge Discovery*. Springer, 299–314.

[3] Surajit Chaudhuri and Vivek Narasayya. 2020. Anytime algorithm of database tuning advisor for microsoft sql server.

[4] Surajit Chaudhuri and Vivek R. Narasayya. 1997. An Efficient Cost-Driven Index Selection Tool for Microsoft SQL Server. In *VLDB*. 146–155.

[5] Surajit Chaudhuri and Vivek R. Narasayya. 1998. AutoAdmin 'What-if' Index Analysis Utility. In *SIGMOD*. 367–378.

[6] Sungjoon Choi, Sanghoon Hong, Kyungjae Lee, and Sungbin Lim. 2020. Task agnostic robust learning on corrupt outputs by correlation-guided mixture density networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3872–3881.

[7] Francis C. Chu, Joseph Y. Halpern, and Johannes Gehrke. 2002. Least Expected Cost Query Optimization: What Can We Expect?. In *PODS*. 293–302.

[8] Youngseog Chung, Willie Neiswanger, Ian Char, and Jeff Schneider. 2021. Beyond pinball loss: Quantile methods for calibrated uncertainty quantification. *Advances in Neural Information Processing Systems* 34 (2021), 10971–10984.

[9] Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. 2018. Implicit quantile networks for distributional reinforcement learning. In *International conference on machine learning*. PMLR, 1096–1105.

[10] Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. 2018. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI conference on artificial intelligence*. AAAI Press, Article 353, 10 pages.

[11] Armen Der Kiureghian and Ove Ditlevsen. 2009. Aleatory or epistemic? Does it matter? *Structural safety* 31, 2 (2009), 105–112.

[12] 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.

[13] Roland L Dobrushin. 1970. Prescribing a system of random variables by conditional distributions. *Theory of Probability & Its Applications* 15, 3 (1970), 458–486.

[14] Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*. PMLR, 1050–1059.

[15] José Miguel Hernández-Lobato and Ryan Adams. 2015. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International conference on machine learning*. PMLR, 1861–1869.

[16] Peter J Huber. 1973. Robust regression: asymptotics, conjectures and Monte Carlo. *The annals of statistics* 1, 5 (1973), 799–821.

[17] Eyke Hüllermeier and Willem Waegeman. 2021. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine learning* 110, 3 (2021), 457–506.

[18] Jan Kossmann, Stefan Halfpap, Marcel Jankrift, and Rainer Schlosser. 2020. Magic mirror in my hand, which is the best in the land? an experimental evaluation of index selection algorithms. *Proceedings of the VLDB Endowment* 13, 12 (2020), 2382–2395.

[19] Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics* 22, 1 (1951), 79–86.

[20] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*. PMLR, 3744–3753.

[21] Antonio Loquercio, Mattia Segu, and Davide Scaramuzza. 2020. A general framework for uncertainty estimation in deep learning. *IEEE Robotics and Automation Letters* 5, 2 (2020), 3153–3160.

[22] David John Cameron Mackay. 1992. *Bayesian methods for adaptive models*. California Institute of Technology.

[23] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. 2019. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in neural information processing systems* 32 (2019).

[24] Stratos Papadomanolakis, Debabrata Dash, and Anastassia Ailamaki. 2007. Efficient Use of the Query Optimizer for Automated Database Design. In *VLDB*. 1093–1104.

[25] R Malinga Perera, Bastian Oetomo, Benjamin IP Rubinstein, and Renata Borovica-Gajic. 2021. DBA bandits: Self-driving index tuning under ad-hoc, analytical workloads with safety guarantees. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 600–611.

[26] R Malinga Perera, Bastian Oetomo, Benjamin IP Rubinstein, and Renata Borovica-Gajic. 2023. No DBA? No regret! Multi-armed bandits for index tuning of analytical and HTAP workloads with provable guarantees. *IEEE Transactions on Knowledge and Data Engineering* 35, 12 (2023), 12855–12872.

[27] Gregory Piatetsky-Shapiro. 1983. The optimal selection of secondary indices is NP-complete. *ACM SIGMOD Record* 13, 2 (1983), 72–75.
[28] Yaniv Romano, Evan Patterson, and Emmanuel Candes. 2019. Conformalized quantile regression. *Advances in neural information processing systems* 32 (2019).
[29] Emanuele Rossi, Bertrand Charpentier, Francesco Di Giovanni, Fabrizio Frasca, Stephan Günnemann, and Michael M Bronstein. 2024. Edge directionality improves learning on heterophilic graphs. In *Learning on graphs conference*. PMLR, 25–1.
[30] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. 2020. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International journal of forecasting* 36, 3 (2020), 1181–1191.
[31] Karl Schnaitter, Neoklis Polyzotis, and Lise Getoor. 2009. Index interactions in physical design tuning: modeling, analysis, and applications. *Proceedings of the VLDB Endowment* 2, 1 (2009), 1234–1245.
[32] Jiachen Shi, Gao Cong, and Xiao-Li Li. 2022. Learned index benefits: Machine learning based index performance estimation. *Proceedings of the VLDB Endowment* 15, 13 (2022), 3950–3962.
[33] Tarique Siddiqui, Saehan Jo, Wentao Wu, Chi Wang, Vivek Narasayya, and Surajit Chaudhuri. 2022. ISUM: Efficiently compressing large and complex workloads for scalable index tuning. In *Proceedings of the 2022 International Conference on Management of Data*. 660–673.
[34] Tarique Siddiqui and Wentao Wu. 2023. ML-Powered Index Tuning: An Overview of Recent Progress and Open Challenges. *SIGMOD Rec.* 52, 4 (2023), 19–30.
[35] Tarique Siddiqui, Wentao Wu, Vivek Narasayya, and Surajit Chaudhuri. 2022. DISTILL: low-overhead data-driven techniques for filtering and costing indexes for scalable index tuning. *Proceedings of the VLDB Endowment* 15, 10 (2022), 2019–2031.
[36] Zekun Tong, Yuxuan Liang, Changsheng Sun, Xinke Li, David Rosenblum, and Andrew Lim. 2020. Digraph inception convolutional networks. *Advances in neural information processing systems* 33 (2020), 17907–17918.
[37] Gary Valentin et al. 2000. DB2 Advisor: An Optimizer Smart Enough to Recommend Its Own Indexes. In *ICDE*. 101–110.
[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
[39] Xiaoying Wang, Wentao Wu, Vivek R. Narasayya, and Surajit Chaudhuri. 2025. Esc: An Early-Stopping Checker for Budget-aware Index Tuning. *Proc. VLDB Endow.* 18, 5 (2025), 1278–1290.
[40] Xiaoying Wang, Wentao Wu, Chi Wang, Vivek R. Narasayya, and Surajit Chaudhuri. 2024. Wii: Dynamic Budget Reallocation In Index Tuning. *Proc. ACM Manag. Data* 2, 3 (2024), 182.
[41] Zijia Wang, Haoran Liu, Chen Lin, Zhifeng Bao, Guoliang Li, and Tianqing Wang. 2024. Leveraging dynamic and heterogeneous workload knowledge to boost the performance of index advisors. *Proceedings of the VLDB Endowment* 17, 7 (2024), 1642–1654.
[42] Wentao Wu. 2024. Hybrid Cost Modeling for Reducing Query Performance Regression in Index Tuning. *IEEE Transactions on Knowledge and Data Engineering* (2024).
[43] Wentao Wu et al. 2014. Uncertainty Aware Query Execution Time Prediction. *Proc. VLDB Endow.* 7, 14 (2014).
[44] Wentao Wu, Chi Wang, Tarique Siddiqui, Junxiong Wang, Vivek R. Narasayya, Surajit Chaudhuri, and Philip A. Bernstein. 2022. Budget-aware Index Tuning with Reinforcement Learning. In *SIGMOD*. 1528–1541.
[45] Yang Wu, Xuanhe Zhou, Yong Zhang, and Guoliang Li. 2024. Automatic Index Tuning: A Survey. *IEEE Trans. Knowl. Data Eng.* 36, 12 (2024), 7657–7676.
[46] Derek Yang, Li Zhao, Zichuan Lin, Tao Qin, Jiang Bian, and Tie-Yan Liu. 2019. Fully parameterized quantile function for distributional reinforcement learning. *Advances in neural information processing systems* 32 (2019).
[47] Tao Yu, Zhaonian Zou, Weihua Sun, and Yu Yan. 2024. Refactoring index tuning process with benefit estimation. *Proceedings of the VLDB Endowment* 17, 7 (2024), 1528–1541.
[48] Tao Yu, Zhaonian Zou, and Hao Xiong. 2024. Can Uncertainty Quantification Enable Better Learning-based Index Tuning? *arXiv preprint arXiv:2410.17748* (2024).
[49] Yue Zhao, Zhaodonghui Li, and Gao Cong. 2023. A comparative study and component analysis of query plan representation techniques in ML4DB studies. *Proceedings of the VLDB Endowment* 17, 4 (2023), 823–835.
[50] Wei Zhou, Chen Lin, Xuanhe Zhou, and Guoliang Li. 2024. Breaking It Down: An In-Depth Study of Index Advisors. *Proceedings of the VLDB Endowment* 17, 10 (2024), 2405–2418.

Received July 2025; revised October 2025; accepted November 2025