Front. Comput. Sci., 2025, 19(12): 1912612
https://doi.org/10.1007/s11704-025-41116-7

REVIEW ARTICLE

Learning database optimization techniques: the state-of-the-art and prospects

Shao-Jie QIAO[1], Han-Lin FAN[1,2], Nan HAN (✉)[3], Lan DU[4], Yu-Han PENG[1], Rong-Min TANG[1,5], Xiao QIN (✉)[6]

1 School of Software Engineering, Chengdu University of Information Technology, Chengdu 610225, China
2 Key Laboratory of Cyberspace Big Data Intelligent Security, Ministry of Education, Chongqing 400065, China
3 School of Management, Chengdu University of Information Technology, Chengdu 610103, China
4 Faculty of Information Technology, Monash University, Melbourne 3800, Australia
5 Key Laboratory of Cyberspace Security, Ministry of Education of China and Henan Key Laboratory of Cyberspace Situation Awareness, Zhengzhou 450001, China
6 Guangxi Key Lab of Human-machine Interaction and Intelligent Decision, Nanning Normal University, Nanning 530100, China

Abstract Artificial intelligence-enabled database technology, known as AI4DB (Artificial Intelligence for Databases), is an active research area attracting significant attention and innovation. This survey first introduces the background of learning-based database techniques. It then reviews advanced query optimization methods for learning databases, focusing on four popular directions: cardinality/cost estimation, learning-based join order selection, learning-based end-to-end optimizers, and text-to-SQL models. Cardinality/cost estimation is classified into supervised and unsupervised methods based on learning models, with illustrative examples provided to explain the working mechanisms. Detailed descriptions of various query optimizers are also given to elucidate the working mechanisms of each component in learning query optimizers. Additionally, we discuss the challenges and development opportunities of learning query optimizers. The survey further explores text-to-SQL models, a new research area within AI4DB. Finally, we consider the future development prospects of learning databases.

Keywords AI4DB, cardinality/cost estimation, join order selection, end-to-end optimizer, Text-to-SQL

1 Introduction

1.1 Research background

Integrating modern artificial intelligence (AI) techniques into databases (DB) has significantly improved DB system performance, leading to wider applications of DB systems due to advanced AI methods. Recently, AI techniques have rapidly advanced and made breakthroughs in three major bottlenecks: big data processing, new algorithms, and powerful computing capabilities [1]. In the era of big data, traditional DB technologies (e.g., cardinality estimation, indexing, view selection) cannot meet high-performance requirements and are unable to efficiently process the increasing volume of data [2]. Additionally, complex and diverse application scenarios, varying hardware environments, and different user needs present further challenges [3] that traditional methods find difficult to solve. In particular, learning-based technologies have great potential to address these challenges [4].

1.1.1 Machine learning for database optimization

Machine learning (ML) methods can find better solutions by analyzing historical data, reducing the need for manual human work [5]. ML techniques are becoming increasingly popular and are widely applied in various fields. For optimizing DBs, training data can include samples from tables, query statements, execution plans, and parameters. Traditional learning-based models, such as linear regression [6], random forest [7,8], support vector machines [9], and ensemble learning [10], can learn from historical data and perform simple classification tasks.

Deep learning (DL) and reinforcement learning (RL) methods for DB optimization have become popular as the aforementioned three bottlenecks are gradually overcome [11]. DL simulates neurons to transmit information for learning, regulating the connections between neurons. It is suitable for complex high-dimensional projection relations [12] and uses gradient descent to minimize the loss function. While NP-hard problems are challenging for traditional methods, DL can effectively handle them, albeit requiring a large volume of training data. DB systems, with their complex components akin to a small operating system, store vast amounts of data, necessitating the integration of DL with DB optimization techniques.

Received November 2, 2024; accepted March 18, 2025

E-mail: hannan@cuit.edu.cn; 7670172@qq.com

RL, on the other hand, learns through continuous interaction with the environment, obtaining rewards to guide actions in a "trial-and-error" fashion. Its goal is to maximize the final reward through a series of actions [13]. Unlike other methods, RL does not focus on specific rewards at any given training instance but aims for the maximum overall reward. It employs a strategy to map states to actions, optimizing system performance. Additionally, deep reinforcement learning models, which combine DL and RL methods, are suitable for complex industrial application scenarios [14−16].

1.1.2 Learning-based database optimization techniques

Traditional DB systems are designed based on experience and specifications, emphasizing human involvement (e.g., database administrators) [17,18]. In query optimization, two important components are the optimizer and the executor[19]. The executor performs various physical operations using the best plan generated by the optimizer, and both components are crucial for improving DB performance [20]. The workflow of Structured Query Language (SQL) execution is depicted in Fig. 1.

Figure 2 shows three important components of the cost-based query optimizer: execution plan selection, and cardinality/cost estimation models. Execution plan selection, also called join order selection, explores the optimal cost-effective join order. In this survey, we introduce the optimizer from three aspects: end-to-end optimizer, join order selection, and cardinality/cost estimation in both traditional and learning-based methods. The cardinality/cost model performs various functions used in execution plans by generating query statements for the DB [21].

The advanced query optimization methods for learning DBs are classified into four directions: cardinality/cost estimation, learning-based join order selection, learning-based end-to-end optimizer, and text-to-SQL models. Specifically,

1. **Cardinality/cost estimation** Cardinality/cost estimation for selecting the optimal execution plan heavily depends on DB optimizers. However, traditional techniques struggle to effectively capture correlations between different columns and tables, resulting in suboptimal estimation results. Currently, enterprise DB cardinality estimators perform well for simple operations, such as single tables. But as data grows drastically and query operations become more complex, cardinality estimation errors can increase significantly, degrading join order selection performance. Regarding cost estimation models, better cardinality estimation leads to better cost estimation. However, cost estimation can be influenced by several factors, such as hardware overhead and cache size, which can increase operation costs. Recently, deep learning [22] has been utilized for estimating cardinality and cost through the capture of correlations across tables. These approaches can yield improved results.
2. **Join order selection** In complex business scenarios, obtaining the optimal query plan is crucial. Traditional DB optimizers use static plan selection, measured by the DB estimator. They apply heuristic algorithms or dynamic programming methods to sample the state space and determine the optimal query plan with the lowest cost. However, static plan selection has a significant drawback: it struggles to find good plans for tables with significant amounts of data. Exploring the vast space of query plans is highly costly, as it pertains to an NP-hard problem. Methods based on deep reinforcement learning [23] can automatically select good plans through trial-and-error and iterative

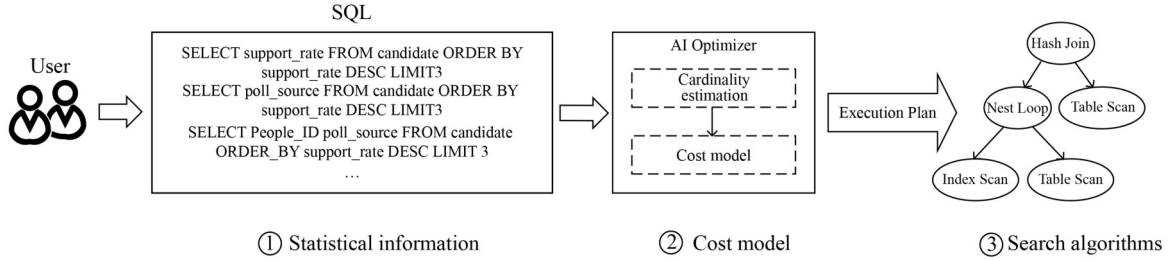


Fig. 1 The workflow of SQL execution

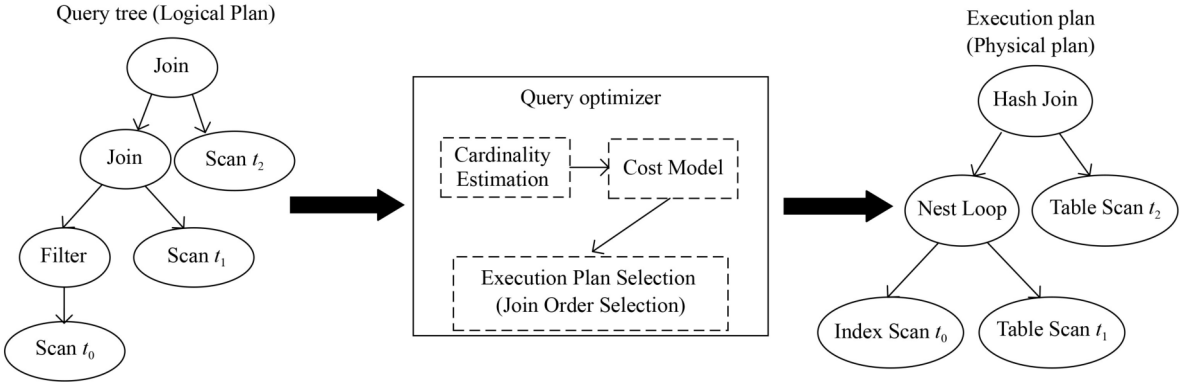


Fig. 2 Overview of a query optimizer

feedback mechanisms.

3. **End-to-end optimizer** Designing an appropriate query optimizer is essential for DB systems. Traditional optimizers do not consider complex DB environments when obtaining the optimal execution plan. However, learning-based optimizers [24,25] can leverage deep neural networks to optimize SQL queries, ultimately achieving the optimal execution plan.
4. **Text-to-SQL** Text-to-SQL enables researchers to interact with DBs without needing to master SQL. As more people utilize DBs for data security and management, not all users possess professional SQL skills. Text-to-SQL models automatically convert user-entered text into machine-executable SQL queries.

1.1.3 Application scenarios of learning database optimization techniques

Machine learning and other learning-based methods combined with DB techniques have significant advantages in disease prediction. They can provide more accurate forecasting results by analyzing a large amount of complex clinical data, thereby improving patient outcomes and the allocation of medical resources. In addition, Li et al. [26] developed a model for predicting acute kidney injury (AKI) in sepsis patients by using machine learning models, which demonstrates excellent prediction performance when combined with the XGBoost model.

In addition, the combination of DBs and machine learning models in social media platforms shows significant performance in large-scale data acquisition. It can collect and label a large number of multimedia data in an efficient and cost-effective manner, thereby improving the performance of DB systems in the real-world applications. A typical work was done by Amiriparian et al. [27]. They developed a system called Cost-efficient Audio-visual Acquisition via Social-media Small-world Targeting (CAS^2T) for rapid and efficient data collection from social media platforms, which has high capability of generating large-scale audio-visual datasets.

Furthermore, the integration of DBs and deep learning techniques can significantly enhance multi-language support, thereby boosting the robustness and accuracy of systems. Akbari et al. [28] proposed a new database system for scene text detection in multi-language environments. This database has various practical applications in the real world, such as image searching, real-time translation, robot navigation, and assisting individuals with visual impairment. It achieves good performance on both English and Arabic languages.

There are challenges in learning database optimization techniques in real-world applications, e.g., Internet of Things (IoT) (including data security and low storage efficiency).

1) Traditional DBs like SQLite store data in the format of plain texts, which can be accessed by malicious users.

2) IoT devices have limited computing resources and memory, which is challenging of encrypting their DBs.

3) IoT devices often collect time-series data at high frequencies, which requires an efficient data insertion and storage mechanism.

It is a good solution of combining DBs and DL methods to cope with the above challenges. For example, to address the issues of traditional agriculture relying on experience or basic monitoring methods, lacking real-time data support and accurate decision-making capabilities, as well as the large amount of data generated in the agricultural IoT domain that lacks effective tools and techniques of data analytic to extract knowledge and provide accurate decision support, Akhter et al. [29] utilized Spark and DL models to process and analyze the collected data. They designed models for predicting crop diseases and outputs. Particularly, the DL model for predicting the scab of apples achieves an R-Squared value of 0.788. This indicates that the proposed model can effectively explain the variations of data and have a high level of prediction accuracy.

1.2 Challenges in implementing AI techniques in databases

While advanced AI technologies have propelled DB development, integrating AI techniques to enhance DB performance presents significant challenges. This survey discusses these challenges including:

- **Algorithm adaptability and generalization** The adaptability and generalization of AI algorithms hinge on the extent to which training and testing data conform to rules derived from high-dimensional data spaces [30]. DB systems are often built on various operating systems, each with unique hardware environments, varying user counts, and diverse user requirements. These factors can limit the model's ability to generalize effectively.
- **Model complexity** Complex models require extensive training time to acquire valuable knowledge, often leading to prolonged training periods. Additionally, query optimizers must consider user requirements, concurrent user numbers, and other factors, which further extend the time needed for model convergence. Consequently, relatively simple models with fast learning speeds and robust learning capabilities are typically preferred for real-world applications.
- **Data quality** AI algorithms require high-quality training data because data quality significantly impacts model training outcomes [31]. However, there is limited availability of data for learning query optimizers, and using outdated or informal data can degrade model performance. Moreover, data correlation necessitates us to take into consideration data distribution, because adequate training data and balanced representation across different data categories play a crucial role in the accuracy of model training. Consequently, insufficient training data may lead to underfitting, while imbalanced data classes can invalidate model training.

Query optimization techniques present numerous complex challenges. Applying AI models to specific optimization tasks like environment construction and scheduling mechanisms can be even more challenging. Although there are learning-based solutions for current optimization tasks, further optimization is necessary before these models can be effectively integrated for everyday use.

1.3 Focus of this survey

Different from the existing surveys, our focus is on query optimization, providing a detailed exploration of its various aspects. We introduce in detail the key ideas behind each category of query optimization techniques. Previous studies, such as those by Li et al. [32], have outlined five directions for DB automatic tuning query, cardinality estimation, query plan selection, indexing, and view selection, but they have not compared existing methods or summarized associated challenges. Azani et al. [33] and Fantinato et al. [34] reviewed DB architecture, workload, data access, and query optimization, but they did not delve into the components of learning optimizers at a granular level. Moreover, with the rise of large models, new avenues have emerged, such as integrating DB and ML techniques like text-to-SQL models. In this survey, we explore text-to-SQL within the AI4DB domain, offering valuable insights to researchers and fostering a comprehensive understanding of these rapidly evolving techniques.

The subsequent sections are presented as follows: In Section 2, we provide a comprehensive review of cardinality and cost estimation techniques, discussing the challenges of query optimization. In Section 3, we first introduce experience-based join order selection algorithms, which are computationally intensive, followed by learning-based methods that leverage machine learning to enhance efficiency and accuracy. Section 4 covers traditional rule-based query optimization methods and transitions to learning-based approaches integrating machine learning to optimize query execution plans. In Section 5, we emphasize the role of deep learning in enhancing text-to-SQL model performance, addressing challenges, and exploring various technologies and methodologies to overcome them. Section 6 introduces technologies that enhance the performance of diverse learning-based DB optimization strategies. Finally, we conclude this review and delineate future research avenues in Section 7.

2 Cardinality/cost estimation

Cardinality estimation, a major focus of study, remains one of the most challenging issues in query optimization [35]. Cost estimation, on the other hand, predicts the I/O and CPU consumption of DB physical execution plans. Compared with cardinality estimation, cost estimation provides a more approximate and straightforward evaluation of execution plan overhead [1]. In this section, we compare the characteristics of learning-based and traditional methods as shown in Table 1, and introduce cost estimation methods.

2.1 Traditional cardinality estimation methods

Traditional models are typically categorized into three classes: sampling, data sketching, and histogram-based methods [23].

1. **Sampling** Qiu et al. [36] proposed a sampling method for SPJ (select-project-join) queries. SPJ can calculate the frequency of attribute occurrences in tables and use this information to obtain the query statement.
2. **Data sketching** Durand et al. [37] proposed the LogLog algorithm, which uses m auxiliary memory ("small bytes") to determine a large range of cardinality with only 1KB or 2KB of memory.
3. **Histogram-based models** Zhang et al. [38,39] used density-based multi-dimensional histograms to cluster data samples and calculate the boundaries of each dense region. The data is stored in multi-dimensional buckets, and the density of each bucket can be calculated.

Histogram and data sketching methods work well for single-column data distributions but incur significant errors for multi-column or multi-table join operations [40]. While histograms are simple and widely used, it is essential to consider the errors and the time and space complexity of these algorithms in real-world scenarios. Sampling-based methods often produce randomly generated results, which may not yield high-quality samples.

To address these limitations, learning-based estimation methods have gradually replaced traditional approaches. Learning-based estimations are more suitable for complex applications, such as multi-table joins and queries, as they can effectively capture data distribution and feature characteristics.

Table 1 Comparison of cardinality estimation models

Categary	Method	Core technique	Estimation	Encoding	Multi-column	Multi-table
Traditional methods	Histogram	Multi-dimensional histogram	Selectivity	/	✓	×
	Sketching	Bitmap and hash	Cardinality	/	×	×
	Sampling	Sampling and indexing	Cardinality	/	✓	✓
Supervised learning methods	Mixture model	Mixture method	Selectively	Predicate	✓	×
	Integrated model	XGBoost	Selectively	Predicate	✓	×
	FCN	Neural network	Cardinality	Operator	✓	✓
		LocalNN	Cardinality	Table and predicate	✓	✓
	CNN	Tree-CNN	Query delay	Execution plan	✓	✓
		Multi-set CNN	Cardinality	Query	✓	✓
	RNN	Tree-LSTM and Tree-GRU	Selectivity	Plan	✓	✓
		Plan-structure RNN	Cost	Metadata and operator	✓	✓
Unsupervised learning methods	kernel density model	KDE	Selectively	Data Sample	✓	×
	Probabilistic graphical model	BPN	Selectively	Data and predicate	✓	×
		SPN, FSPN	Selectively	Data and predicate	✓	✓
		SPN, FSPN	Selectively	Data and predicate	✓	✓
	Deep autoregressive model	Deep autoregressive	Selectively	Data and predicate	✓	✓
	Normalizing flow model	Normalized Flow	Selectively	Data and predicate	✓	✓

2.2 Learning-based cardinality estimation models

Learning-based cardinality estimation models are generally classified into supervised and unsupervised learning models, which are given in Table 1.

2.2.1 Supervised methods

The workflow of cardinality estimation based on supervised learning is depicted in Fig. 3. First, the query feature extractor retrieves features of SQL queries from the DB, using the ground-truth cardinality as the label. Then, the features are encoded by the query encoder. After that, the encoded features are utilized as input in the model during the training process to predict the cardinality. Consequently, the model parameters can be automatically optimized by the parameter optimizer. Supervised learning-based cardinality estimation methods primarily include the following categories of models:

1. **Mixture model** Few studies utilize the mixture model method for cardinality estimation. In [41], mixture models are used to efficiently compute the Euclidean distance, and the model's quality is evaluated using maximum, minimum, and multiplication operations. Mixture models can reduce the cost of multidimensional histograms, but they do not support predicates such as "LIKE", "EXISTS", and "ANY".
2. **Ensembling model** The ensembling model addresses cardinality estimation by formulating it as a regression problem and utilizing a tree-based boosting method. It corrects errors generated by previous models and recursively partitions feature dimensions with the highest gain. Compared to other ensembling models, XGBoost [42] offers shorter estimation times and higher prediction accuracy.
3. **Fully-connected neural network (FNN)** FNN divides the query into different structures and encodes them as one-hot vectors, which are then concatenated into a long one-hot vector and input into a neural network to obtain the cardinality. A model was proposed by Dutt et al. [43] to handle multidimensional predicates in join order selection. This model has small spatial and temporal overheads for join order selection and can learn the correlations of multiple columns in a single table, outperforming previous methods. However, it struggles to capture multi-table join relationships. Kipf et al. [44] considered the relationships between different predicates in the join order selection path, making it easier to learn the path with less consumption by using a simple multi-layer perceptron.
4. **Convolutional neural network (CNN)** CNN divides the query into different structures, encodes them into one-hot vectors, and inputs these one-hot vectors into the CNN for learning. Finally, CNN partitions the vectors into splices during the learning phase of cardinality estimation. Kipf et al. [45] applied a multi-set convolutional neural network called MSCN, as shown in Fig. 4, for cardinality estimation.

   MSCN partitions a query into tables, join conditions, and predicates. The involved columns are sampled, compressing samples into bitmaps and splicing them into table vectors, which are connected by a CNN and passed through a mean-pooling layer. This method cannot handle predicates with complex strings or nested queries. In recent studies on cardinality estimation using CNNs, Qiao et al. [22] proposed a vertical scanning CNN called VSCNN based on MSCN. VSCNN uses three convolutional neural networks of different widths to scan from the first word vector to the last one, generating different feature maps. These feature maps are transformed into a long feature vector, which is finally estimated by a multilayer perceptron.
5. **Recurrent neural network (RNN)** The basic concept

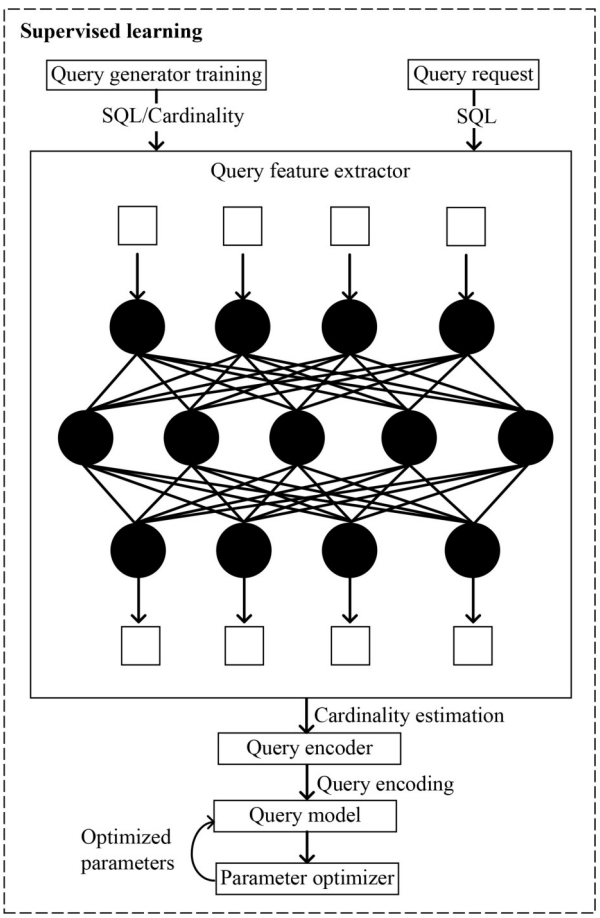


Fig. 3 Working mechanism of the supervised learning methods

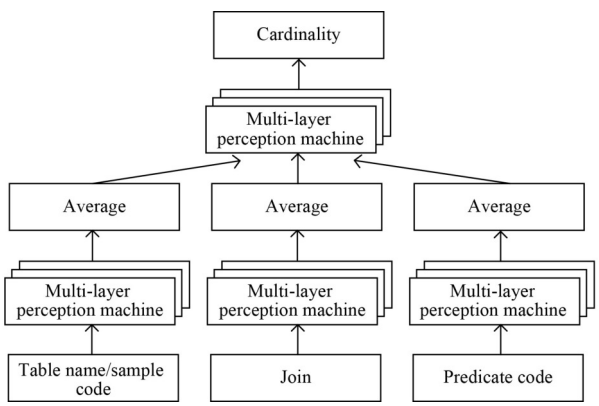


Fig. 4 An example of MSCN

involves cardinality estimation using RNNs based on the temporal relationships within SQL queries [46]. Given that execution plans are tree-structured, employing an RNN with a tree structure is particularly suitable. Each node of the execution plan serves as input to the RNN, incorporating information from lower-level nodes, which optimizes results. The architecture is illustrated in Fig. 5, showing only dashed arrows indicating relationships between nodes and corresponding recurrent units for clarity. A cost model based on RNNs was proposed in [47], which relies on cardinality estimation and statistical parameters. This model encodes execution plans as tree-structured DNNs (deep neural networks). However, encoding predicates depends heavily on accurate cardinality estimates, as inaccuracies can lead to errors in cost estimation. In [48], an end-to-end cost estimator was utilized in a tree-based model. This approach leverages operators and predicates to learn the expression of each sub-plan, deriving cardinality and cost in the output layer of an LSTM-based model. Moreover, the method incorporates word embeddings for predicates. Recent advancements in RNN-based cardinality estimation [22] enhance the tree-based LSTM model by substituting LSTM with GRU to expedite training. Addressing complex string processing, [48] employs pattern matching based on predefined rules derived from experience. In contrast, [22] introduces a rapid model for learning string patterns, significantly improving pattern matching efficiency.

2.2.2 Unsupervised methods

In unsupervised learning, understanding the joint distribution of data is crucial. The workflow of unsupervised learning is illustrated in Fig. 6. First, the SQL statement undergoes parsing by a query parser. Second, data is sampled based on the results of SQL parsing, and values or tuples are encoded accordingly. Third, the encoded results are input into a data model to derive probability values, which are then used to predict cardinality. Various methods integrate different models to capture the joint data distribution [49], including kernel density models, probabilistic graphical models, deep autoregressive models, and normalizing flow models. Specifically,

1. **Kernel density model** Heimel et al. [50] introduced a selective estimator based on Kernel Density Estimation (KDE), designed to construct and maintain a lightweight model that adapts to changing data distributions. KDE is known for its ability to quickly adjust to data distributions and its robustness to correlated data. However, it does not support multi-table queries.
2. **Probabilistic graphical model** A probabilistic graphical model relies on a graph or tree structure to represent relationships between entities [51]. In these models, nodes represent entities, and edges denote relationships between them. Commonly used types of probabilistic graphical models include the sum-product network (SPN) [52] and Bayesian network (BN) [53,54].

   SPN is a structured network model that enhances directed acyclic graphs with weighted operators, capable of representing graphical models. DeepDB [52] exemplifies SPNs by utilizing them to estimate cardinalities. This model follows three main steps: 1) gathering training data by outer joining all tables; 2) partitioning the training data to create various nodes within the network; and 3) computing node weights by fitting the joint probability distribution of the data. An application example of SPN is depicted in Fig. 7. Zhu et al. [55] introduced FSPN (Factorize-Sum-Split-Product Network), which adaptively decomposes joint probability densities based on the degree of attribute dependence. Attributes are categorized into strongly correlated, where values across different dimensions influence each other, and weakly correlated, where joint probability densities are decomposed into independent small regions.

   BN is a graphical network model used to describe uncertain causal relationships between variables, consisting of nodes, directed connections, and node probability tables, where directed connections represent causal dependency relationships between nodes. BNs share a similar structure to SPNs, both being directed acyclic graphs. Nodes in BNs represent random variables, enabling them to simulate causal relationships derived from manual reasoning processes. Chow et al.

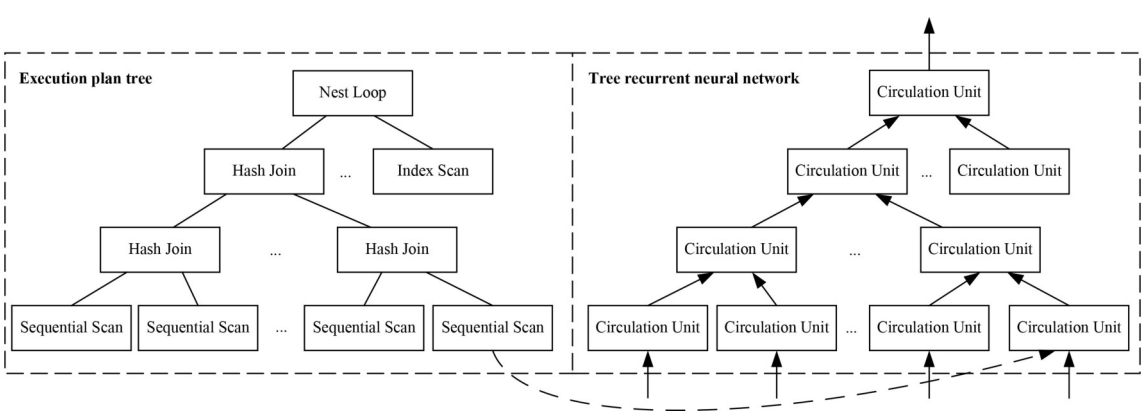


Fig. 5 An example of tree-based RNN

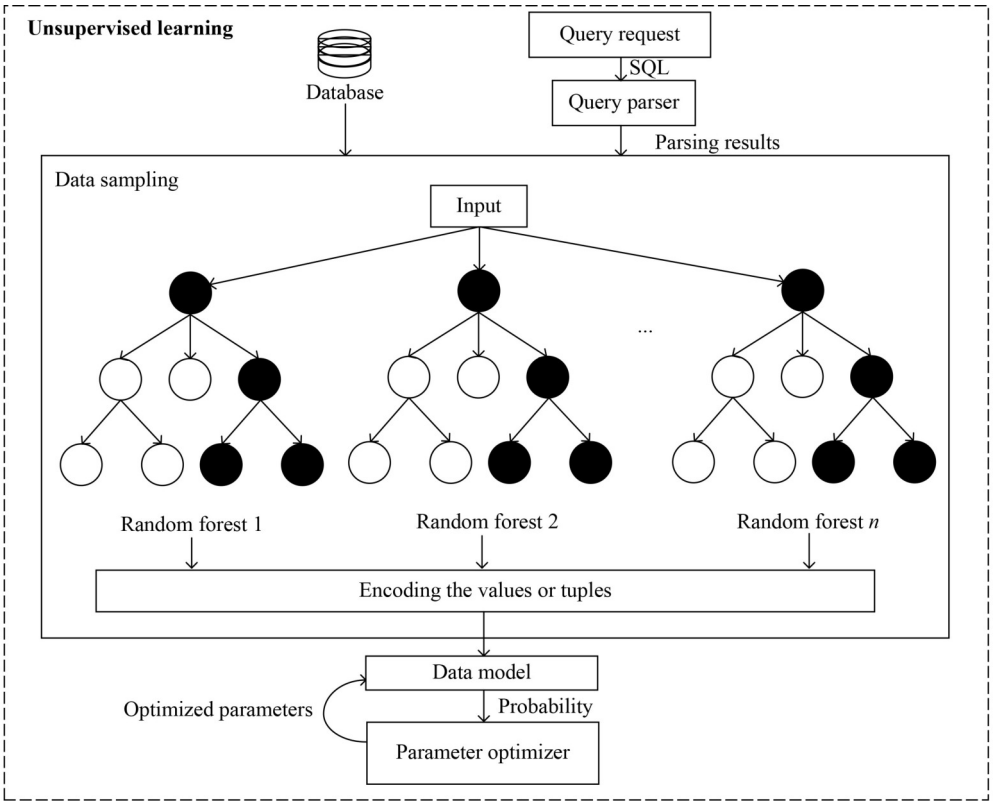


Fig. 6 Working mechanism of the unsupervised learning method

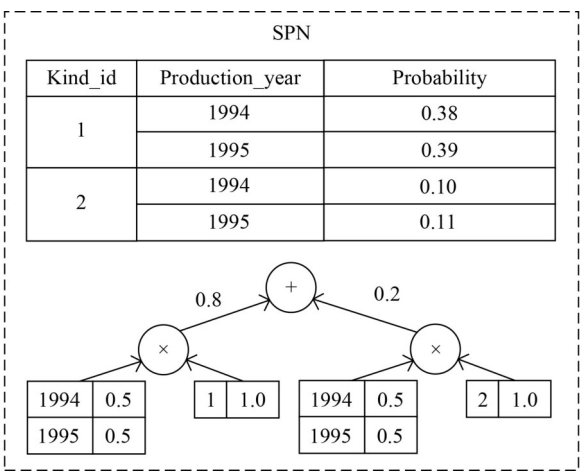


Fig. 7 An example of the application of SPN

[53] applied BNs to cardinality estimation by treating attributes as nodes and learning data distributions to predict cardinalities. Figure 8 illustrates an example of a BN for cardinality estimation. The diagram on the top left shows the structure of the BN. V1 and V2 are parent nodes that influence V3, which in turn influences V4. This indicates that V3 acts as an intermediate variable that is affected by both V1 and V2 and subsequently affects V4. The figure on the top right shows the probabilities associated with V1 taking on the values 0 or 1. In the top center figure, V1 has a 40% probability of 0 and a 60% probability of 1. The figure next to it shows the probabilities of V2. V2 has a 70% probability of 0 and a 30% probability of 1. The bottom left figure illustrates the conditional probabilities of V3 given the values of V1 and V2. There are four possible combinations of V1 and V2, each leading to different probabilities of V3 being 0 or 1. The bottom right figure shows the conditional probabilities of V4 given the value of V3.

3. **Normalizing flow model** Wang et al. [56] introduced the FACE cardinality estimator, utilizing a normalizing flow approach. This model employs a sequence of invertible and differentiable transformations to approximate joint distributions. FACE converts complex data distributions of continuous random variables into simpler distributions and enables the calculation of the probability density for each tuple.

2.2.3 Experimental studies

Existing research across various communities reaches a consensus that learning-based models have the potential to supplant traditional estimators. However, a critical question arises: Are learning-based cardinality estimation models suitable for real-world applications? Addressing this issue, Wang et al. [57] focused on static environments (where data does not update) and compared various learning-based methods with traditional approaches using different datasets. Experimental results demonstrate that learning-based methods generally outperform traditional methods in accuracy but often require longer training times and incur higher computational costs. Furthermore, in dynamic environments (where data frequently updates), these models struggle to adapt to changes, resulting in significant errors in cardinality predictions. More significantly, these estimators prove challenging for predicting

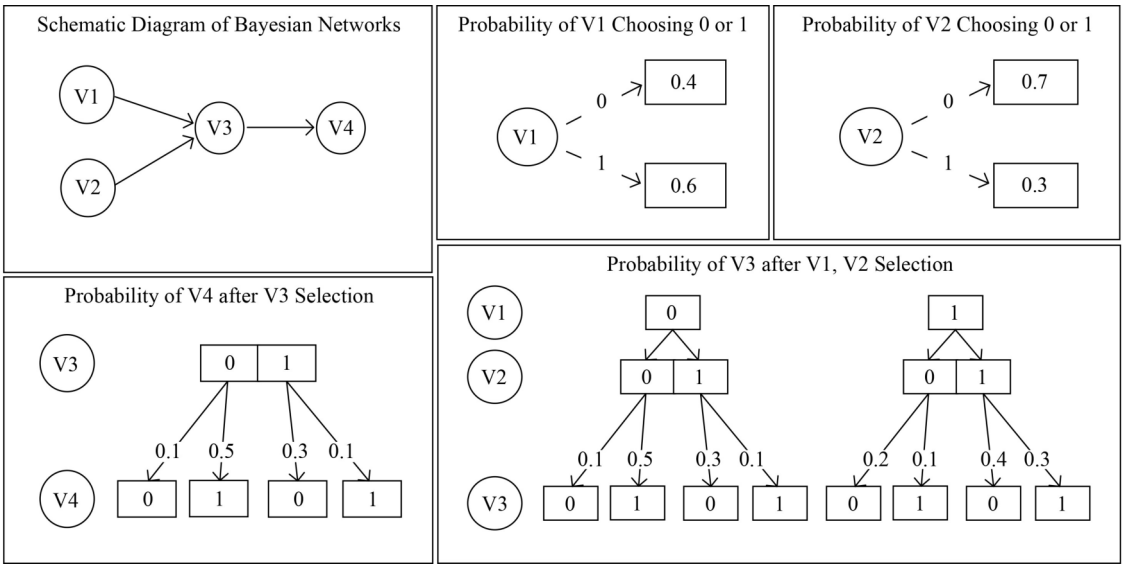


Fig. 8 An example of the Bayesian network

cardinalities reliably. Sun et al. [58] explored the design space of learning-based cardinality estimators, provided a comparison of state-of-the-art learning-based methods, and offered practical suggestions for selecting the most suitable model for diverse applications.

2.2.4 Learning-based methods comparison in Q-error

This subsection systematically compares traditional and learning-based cardinality estimation methods in Q-error which is an important metric. The traditional methods for comparison include:

1) MySQL [59]: MySQL is a widely used DB management system in various research and industrial fields. MySQL primarily estimates bases through statistics, histograms, and data sampling.

2) Postgres [60]: Postgres is an open-source relational DB. Cardinality estimation in Postgres mainly relies on traditional theoretical assumptions and sampling methods. It establishes the corresponding histograms and statistical data for each column in the DB. These statistical data, combined with established assumptions, are used to perform cardinality estimation.

3) Sample [61]: Sampling methods are an important type of cardinality estimation method, but the quality of cardinality estimation is strongly positively correlated with the quality of the samples.

4) MHIST [62]: MHIST constructs a multi-dimensional histogram over the entire dataset to achieve cardinality estimation.

The learning-based research methods for comparison include:

1) MSCN [45]: MSCN utilizes a multi-layer convolutional neural network to convert query statements into vector encodings. It uses these encodings and the ground-truth cardinalities as training data. The model also applies a summary bitmap to optimize the performance.

2) DeepDB [52]: DeepDB employs a tree-shaped non-neural network structure, modeling the data by using Sum-Product Networks (SPNs) which supports multi-table joins. It can achieves good performance in cardinality estimation as well as approximate query processing (AQP).

3) XGBoost [42]: XGBoost treats cardinality estimation as a regression problem, which is a classic query-driven cardinality estimation approach. XGBoost is particularly suitable for scenarios that require high precision and large-scale data processing. However, its model complexity is high, the parameter tuning process is complex, it requires high data quality, the training time is long, and the memory consumption is high.

The comparison experiments are conducted on the Census dataset and the results are shown in Fig. 9. Census contains national or regional census data and include individuals' socioeconomic information, including income, education level, occupation, and family status. Since these attributes are usually independent of each other in the real world, the dataset has a relatively low attribute correlation and the data distributions are independent.

The popular evaluation metric is Q-error [63], which calculates the maximum ratio of either the estimated cardinality (denoted by "estimated") to the actual cardinality (denoted by "actual") or the actual cardinality to the estimated cardinality, indicating the greatest relative discrepancy between the the estimated cardinality and the actual cardinality. Q-error is defined in Eq. (1).

$$Q\text{-}error = \max\left(\frac{C_{\text{estimated}}}{C_{\text{actual}}}, \frac{C_{\text{actual}}}{C_{\text{estimated}}}\right). \tag{1}$$

It is usually used by cardinality estimators. The closer the Q-error is to 1, the more accurate the estimation is; the further it is from 1, the greater the error of the estimation is.

There are four popular comparison metrics of Q-error, including:

1) 50th Q-error: This represents the middle evaluation value among all cardinality estimation errors. For 50% of the queries, the Q-error will be less than or equal to this value, while the other 50% will be greater than this value. It reflects the central tendency of an estimator's performance.

2) 95th Q-error: This indicates that only 5% of cardinality estimation have a Q-error greater than this value. It is typically used to measure the performance of an estimator in most cases and can be considered as the worst performance of the estimator in most circumstances.

3) 99th Q-error: It is similar to the 95th Q-error, but it is more extreme, with only 1% of the queries having a Q-error greater than this value. It is used to evaluate the extreme bad performance of the estimator.

4) Max Q-error: This represents the maximum error that an estimator might encounter in all queries. It helps to understand the performance of the estimator in the most unfavorable situations and the risks it might face with.

From Fig. 9 [57], we can draw the following conclusions:

1) Learning-based estimation methods are more accurate than traditional methods in almost all cases. The best learning-based method outperform the best traditional method by more than ten times in terms of the maximum Q-error.

2) Traditional methods generally have poor stability and have higher Q-error values in experiments. Sampling-based methods perform well on small datasets, but due to the instability of random sampling, the quality of the estimations does fluctuate to some extent.

2.3 Cost estimation

Cost estimation differs from cardinality estimation as it relies heavily on DB states and operational costs for accurate predictions. In this section, we delve into both traditional and learning-based cost models, which include:

1. **Traditional cost estimation model** Traditional cost estimation models employ heuristic functions to calculate user-induced costs when interacting with DBs. He et al. [64,65] introduced a tree-structured heuristic function that efficiently stores essential information within a defined data structure, enabling rapid data lookup and insertion. In another approach, Liu et al. [66] proposed a hash-join-based cost estimation model tailored for main-memory DBs. Initially, they employed a controlled microbenchmark procedure to assess the performance cost of each access pattern. Subsequently, the model computed the number of operations for each pattern within specific query plans, culminating in the estimation of query statement costs by weighing these operations based on their respective performance costs.
2. **Learning-based cost model** Leis et al. [67] have demonstrated the critical importance of cardinality estimation over cost estimation in single relational DBs. Their extensive experiments, conducted on DBs like PostgreSQL, aimed to analyze how accurate or inaccurate cardinality estimates impact execution plans in cost-based models. Surprisingly, correct cardinality estimation led to optimal execution plan selections by simple cost-based models, whereas inaccuracies resulted in suboptimal plans. Recent studies by Qiao et al. [22] and Sun et al. [48] have expanded beyond cardinality estimation to include cost estimation using supervised methods. They employed multi-task learning approaches, where the loss function was designed to simultaneously optimize for both cardinality and cost estimation. Despite this capability, their primary focus remained on improving cardinality estimation.

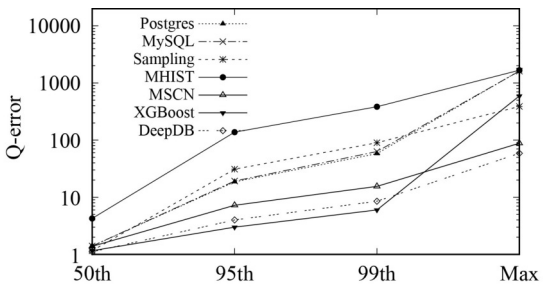


Fig. 9 Q-error comparison of cardinality estimation methods

Meanwhile, several studies focus on estimating query statement costs in big data scenarios. Siddiqui et al. [68] proposed query processing methods tailored for cloud computing environments. Given the diverse nature of cloud workloads, finding representative sets to tune query optimizers can be challenging. To address this, they developed small-scale, cost-based models trained to enhance convergence speed and accuracy. They introduced a hybrid model that automates validation and integrates predictions from multiple models, aiming to achieve stable and precise cost estimations.

3 Join order selection

The selection of the optimal join order plays a crucial role in optimizing DBs. Given the myriad possibilities for joining orders within a query, selecting the optimal execution plan becomes pivotal. This process aims to identify the plan with the lowest cost and overhead among various execution strategies. This research categorizes join order selection methods into three main groups: traditional algorithms, static learning methods, and dynamic learning methods. Table 2 provides a summary of these join order selection techniques.

3.1 Traditional methods

Traditional join order selection algorithms rely on cardinality estimators and cost models as their evaluation metrics. These components are typically designed based on practical experience. Such algorithms navigate through the state space to determine the optimal execution plan based on these metrics. Various algorithms employ distinct selection strategies to achieve effective plan selection, falling into categories such as dynamic programming methods and heuristic methods depending on their specific approach.

Dynamic programming (DP) aims to consolidate redundant states and shrink the search space. In the context of join order selection for relational DBs, DP algorithms have been widely adopted. Ioannidis et al. [69] introduced a left-deep-tree join approach where a primary path is formed for each relation to be connected, and the optimal path is chosen iteratively for each relation along this path. While DP methods typically select the plan with the lowest query cost, they often entail significant computational overhead.

Heuristic methods traverse a partial state space using predefined strategies. The gene optimizer [70] is widely

Table 2 Summary of join order selection methods

Classification	Method	Core technique	Features	Application scenarios	Performance	Adaptability
Traditional method	Dynamic programming	Leftist tree	The cost of time and space is in exchange for the best order	The estimator is accurate and the number of tables are few	Medium	Low
	Heuristic method	Genetic algorithm	Low time cost, but the order of join selection is not always optimal	The estimator is accurate and the number of tables are few	Low	High
Learning method	Static learning method	Experience feedback	Tuning based on historical experience with high cost	Estimator is not accurate and the number of tables is low	Medium	Medium
	Dynamic learning methods	Adaptive processing	High efficiency, but weak generalization	The estimator is not accurate and the number of tables are high	High	Medium

utilized in relational DBs. Bennett et al. [70] encoded the problem according to their criteria to define the state space size, evaluating whether the discovered space meets termination conditions. If not satisfied, the method iterates through its steps until completion, resulting in the optimal path. Subsequently, an execution plan tree is derived based on this optimal path. While heuristic algorithms can significantly reduce computation time, they do not always guarantee the best plan.

3.2 Learning-based models

Recent research has introduced learning-based methods for join order selection to mitigate the limitations from traditional approaches. These methods leverage machine learning techniques to understand historical join patterns and mitigate biases stemming from inaccurate cardinality and cost estimates. Learning-based approaches excel in efficiently identifying optimal solutions and are categorized as static or dynamic models. Their distinction lies in whether the selected join order can be adjusted during query execution.

3.2.1 Static models

Stillger et al. [71] introduced LEO, a static learning-based query optimizer that incorporates a feedback loop for continuous improvement based on past errors. Building on this concept of feedback, Marcus et al. [72] developed ReJoin, a reinforcement-learning-based neural network designed for optimizing join order selection. ReJoin utilizes historical execution plan costs as training data to evaluate and select optimal join orders efficiently. It outperforms PostgreSQL's optimizer in cost reduction without sacrificing performance speed. However, ReJoin's neural network architecture is simplistic and does not fully support the complex join tree structures essential for learning query plan latency effectively. Moreover, it lacks mechanisms to adapt to changes in DB schemas.

In an effort to address the limitations of ReJoin, Yu et al. [23] proposed a two-phase training strategy to enhance join order generation, leveraging a sophisticated deep learning model tailored for handling query plan optimization. Recognizing the challenge of capturing join tree structures with graph neural [73] networks in ReJoin, their model incorporates a tree-LSTM architecture to represent the hierarchical nature of join operations. This novel approach involves designing vector representations for columns, tables, and join trees. First, vector representations are generated, followed by the use of a deep Q-Network to guide the join order generation phase. The model is then pre-trained using cost data and subsequently fine-tuned using latency information. By integrating deep reinforcement learning (DRL) with a tree-structured deep learning model, their approach effectively constructs optimized join trees, as illustrated in Fig. 10.

Furthermore, Zhang et al. [74] integrated Monte Carlo search trees with an AlphaGo-inspired model to simulate various join order possibilities within a tree-based network structure. Their approach leverages Monte Carlo search trees to optimize the selection of join orders based on accurate estimations.

3.2.2 Dynamic models

Moving beyond static learning paradigms, Avnur et al. [75] introduced a dynamic learning model named Eddy, which dynamically learns and generates join order queries based on query execution and optimization. Eddy divides the query process into multiple operators, such as three relations with two join operators between them. Two routing strategies, Naive Eddy and Lottery Eddy, are employed to determine the order of execution. Naive Eddy prioritizes operators with lower costs. Trummer et al. [76] introduced Skinner-DB, a more versatile reinforcement learning model. Unlike traditional Q-Learning, Skinner-DB utilizes confidence upper bounds within the decision tree, providing a "reversal" mechanism for cumulative selection outcomes. Skinner-DB partitions query execution into multiple time slices and selects a join order to execute in each slice.

3.3 Learning-based and traditional methods comparison in GMRL

In this section, we compare learning-based join order selection

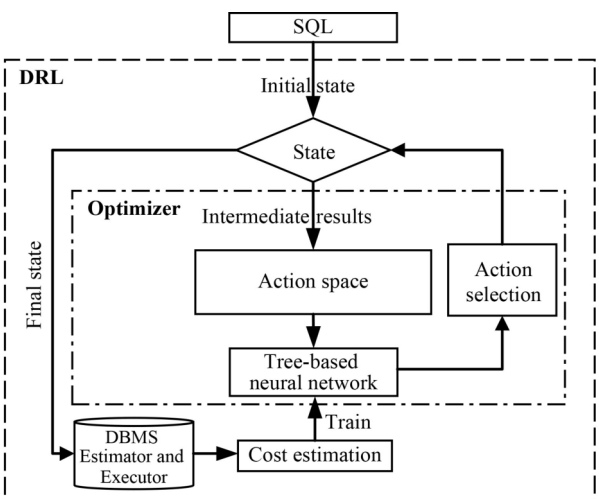


Fig. 10 An example of DRL combined with a tree-based neural network for join order selection

methods with traditional methods. The learning-based methods include RTOS [23], ReJoin [24], Skinner-C [76], and DQ [77]. For traditional optimization algorithms, QP1000 and DP are used for comparison. QP1000 is a method for selecting join order in DB query optimization. Given a query, QP1000 randomly tests 1,000 possible query execution plans and selects the one with the lowest cost as the best join order. As described in Subsection 3.1, DP selects the plan with the lowest query cost. In Fig. 11, DP is treated as a baseline method, which can be seen in Eq. (2).

These optimization algorithms are executed in the same environment. Then, they are compared based on the latency of the execution time of each algorithm by using Geometric Mean Relevant Latency (GMRL) [23] which is defined in Eq. (2).

$$\mathrm{GMRL} = \left(\Pi_{i=1}^{n} \frac{\mathrm{Latency}(q_i)}{\mathrm{Latency}_{\mathrm{DP}}(q_i)}\right)^{\frac{1}{n}}. \tag{2}$$

Figure 11 shows the comparison results of GMRL of learning-based and traditional methods, where DP is a baseline method [23]. If the GMRL value is lower than 1, which indicates that the method can obtain a better plan in terms of delay when compared to DP. We can see that: in the benchmark dataset of JOB and TPC-H, RTOS outperforms other methods. By comparing with the traditional QP1000 method, we can see the superiority of learning-based algorithms (i.e., RTOS, ReJoin, Skinner-C, and DQ) in join order selection. The advantages derived from learning techniques make them a powerful tool for handling complex queries, large-scale data, and real-time environments. However, we should also note that learning-based algorithms are not impeccable. They require appropriate training data, parameter tuning, and continuous maintenance. Therefore, applying traditional algorithms and learning-based algorithms in different real-world applications, may be an effective and practical strategy.

4 End-to-end optimizer

The query optimizer employs a cost model to automatically determine the optimal approach for a given SQL query. The accuracy of cardinality estimation plays a crucial role in the execution plan, especially in the case of intricate queries that include multiple predicates. End-to-end optimizers that integrate cardinality and cost estimation with join order selection have emerged as a vibrant and challenging research area.

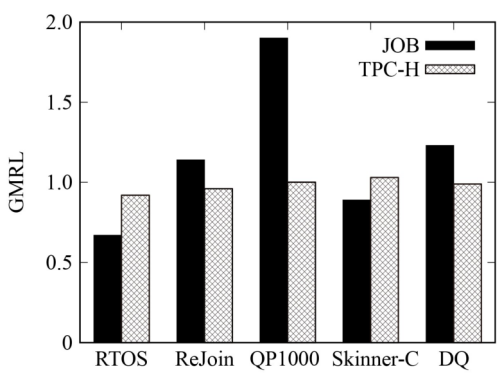


Fig. 11 GMRL comparison of join order selection methods

Markl et al. [78] introduced an autonomous query optimizer that rectifies incorrect cardinality estimations without user intervention. This optimizer monitors executed queries, compares estimated cardinalities against ground-truth values, and computes tuning factors for optimizing similar queries in the future. Moreover, it triggers re-optimization during execution upon detecting inaccuracies in estimations. Marcus et al. [24] proposed NEO, an end-to-end optimizer that eliminates the need for separate cardinality and cost estimations by directly generating the final physical plan. NEO employs a tree-CNN to capture structural information, using row vectors for predicates and one-hot vectors for selecting each physical operator or index in the neural network during plan generation. The system pre-trains on PostgreSQL plans without requiring cost model information, leveraging latency data to generate robust execution plans while mitigating estimation errors.

While replacing traditional optimizers entirely with neural networks remains challenging, researchers have introduced the Bandit optimizer (Bao) [79], depicted in Fig. 12. Bao offers multiple sets of "hints" to constrain the optimizer's search space, producing several optimizers with varying search scopes. Subsequently, deep learning selects the execution plan generated by one of these optimizers. This approach extends beyond existing optimizers, offering engineering advantages such as improved debuggability compared to black-box end-to-end learning optimizers. Bao supports all SQL types handled by the original optimizer, facilitating easier expansion with new operators and faster training. Negi et al. [80] applied a similar optimization strategy from Bao to Microsoft's big data system, SCOPE. Unlike PostgreSQL, SCOPE employs a cascade-style optimizer where optimization relies on a set of rules including rewriting rules, join order rules, and physical implementation rules to define the search space and achieve optimizations.

Bao also presents some drawbacks, such as the need to adjust the optimizer for each set of "hints" each time. While training in parallel can enhance speed, it is not well-suited for OLTP (Online Transaction Processing) scenarios where queries are typically completed within milliseconds [81]. Instead, Bao is more suitable for OLAP (Online Analytical Processing) scenarios where queries often require longer execution times.

5 Text-to-SQL

Text-to-SQL aims to develop a generator that translates natural language into SQL queries, enabling non-experts to obtain SQL queries using this tool. Traditional methods do not utilize deep learning techniques and they often work as follows: users input natural languages, and then the model processes them to generate the corresponding SQL queries. The challenges in text-to-SQL research primarily revolve around three aspects: 1) preprocessing input texts to extract the meaning of natural language, identify relevant DB keywords, column names, table names, and simplify model training; 2) converting preprocessed texts into an intermediate

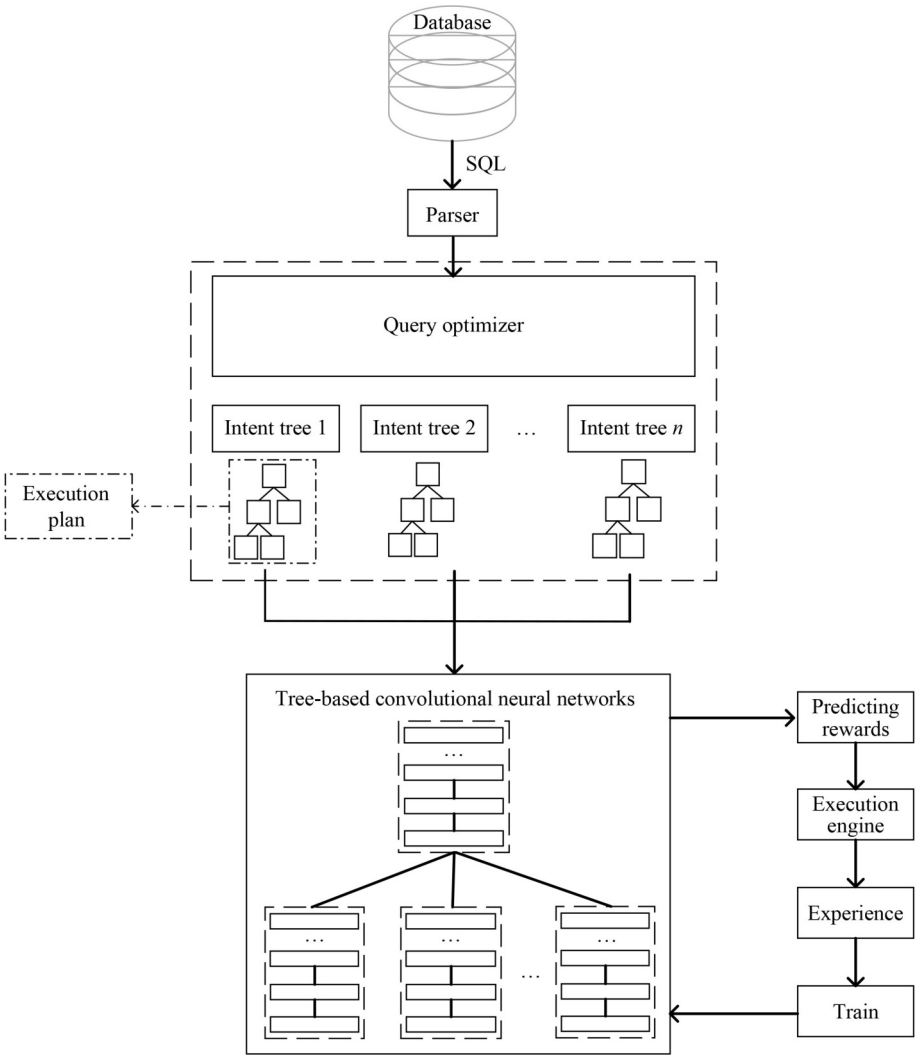


Fig. 12 The structure of the Bao model

representation. The goal is not merely comprehension of SQLs but enhancing DB management efficiency. Given the significant gap between SQL queries and natural language, establishing the relationship between the two is crucial; and 3) converting the intermediate representation into SQL queries.

While traditional text-to-SQL methods are effective, they require extensive labeling and the creation of SQL conversion templates for various scenarios. They often struggle to address the challenges above, lacking intermediate representations that streamline the conversion process. Deep learning is gradually being integrated into text-to-SQL. RNN models have proven effective in this domain, treating both text and SQL queries as sequential data that requires the integration of context from past and future information for accurate prediction.

5.1 Key techniques

Text and SQL queries are both sequential data, making sequential models ideal for training them. In this Subsection, we discuss prevalent sequential models and evaluation metrics. Figure 13 provides an illustrative example demonstrating the operational mechanism of text-to-SQL models.

The Long Short-Term Memory neural network (LSTM) [82] represents a type of recurrent neural network (RNN) enhanced with three gating units: the input gate unit, the forget gate unit, and the output gate unit. Unlike traditional RNNs, LSTM includes a memory cell to maintain long-term information and selectively forget short-term inputs.

In contrast, the Gated Recurrent Unit (GRU) [83] is a simplified variant of LSTM, reducing the number of gating units from four to two (reset and update gating units). GRU is known for its efficiency and faster training on simpler input sequences, whereas LSTM is preferred for handling more complex semantic information within input sequences.

The Transformer model [84] operates on an encoder-decoder architecture. The encoder first converts input sequences into vector representations using word and positional embeddings. It then passes contextual information to the decoder. The decoder not only receives this contextual information but also incorporates previously generated symbols and processes sequential information at each step. Figure 14 illustrates the architecture of text-to-SQL models based on the Transformer framework. BERT (Bidirectional Encoder Representations from Transformers) [85] is a variant of the Transformer model that exclusively uses the encoder part. It captures bidirectional contexts by pre-training on large

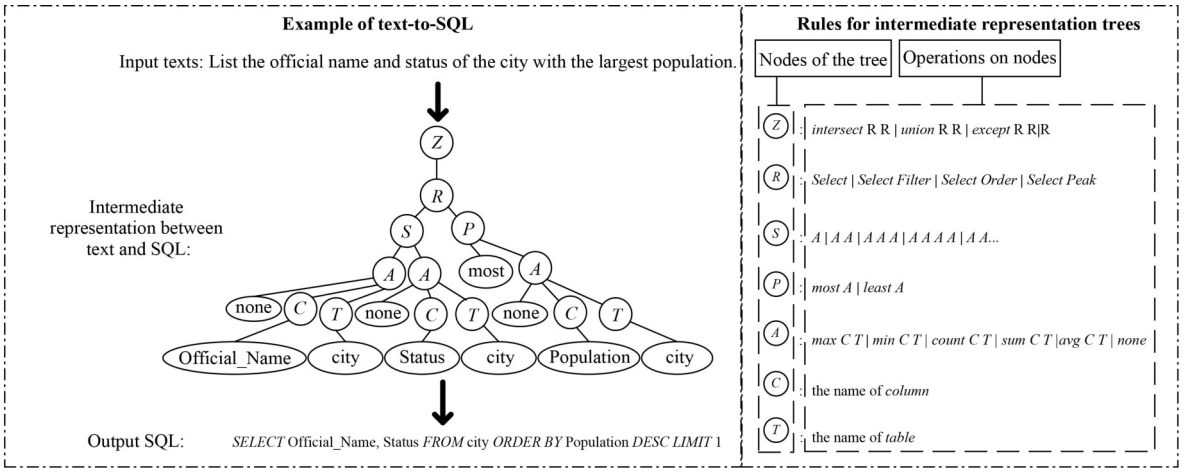


Fig. 13 An example of the working process of text-to-SQL models

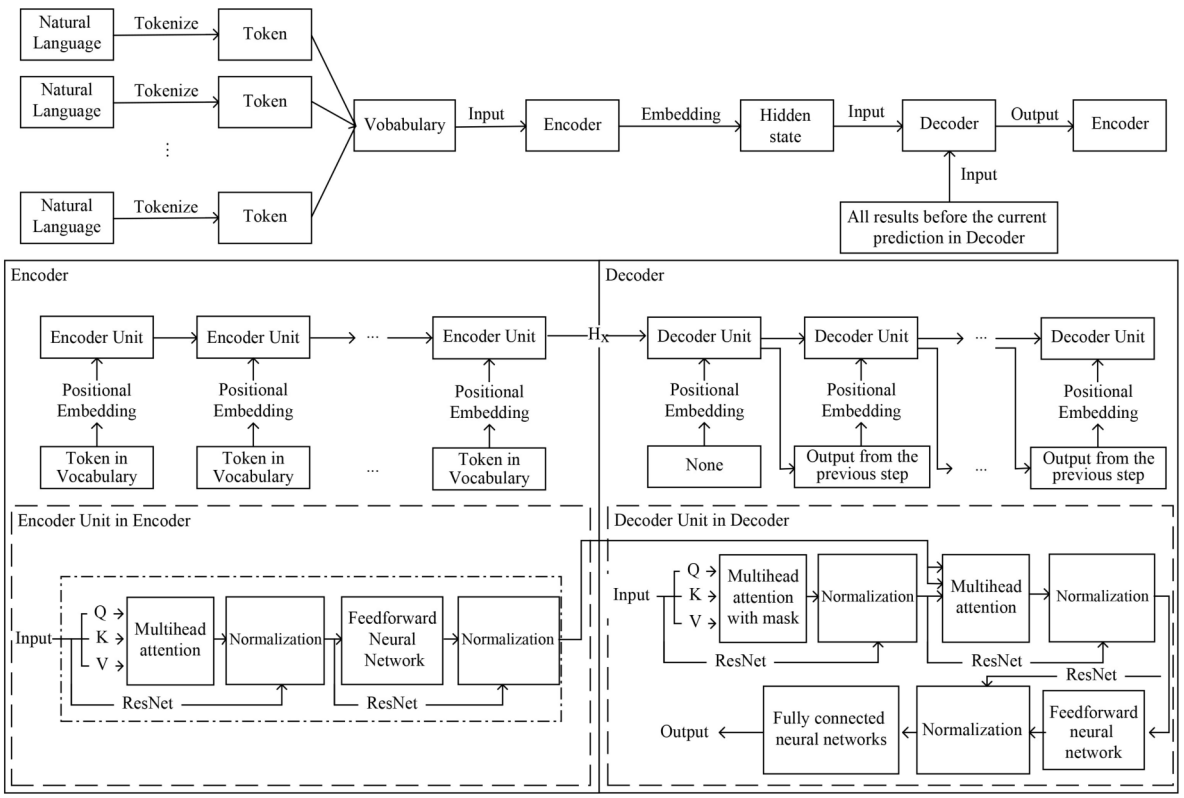


Fig. 14 An example of text-to-SQL model based on Transformer

corpora of texts and subsequently fine-tuning for specific tasks. BERT has a parameter count similar to that of a standard Transformer with comparable specifications. Its pre-training includes tasks aimed at learning robust language representations. An example of a text-to-SQL model based on BERT is depicted in Fig. 15.

The dataset used in the evaluation is divided into training, development, and testing sets. The development set monitors model performance during training, facilitating continuous optimization based on feedback. The testing set evaluates model generalization to unseen data. Out of domain (OOD) [86] data comprises samples with distributions differing from the training data, often included in the testing set to assess model performance on previously unencountered data.

BLEU Score (Bilingual Evaluation Understudy Score) [87] is the common metric assessing machine translation quality on a scale from 0 to 1. Alongside BLEU, metrics like METEOR (Metric for Evaluation of Translation with Explicit Ordering), ROUGE (Recall-Oriented Understudy for Gisting Evaluation), and PPL (Perplexity) can also evaluate text generation models. Metric selection depends on specific task requirements.

5.2 Benchmark text-to-SQL datasets

The quality of datasets significantly impacts the effectiveness of model training, particularly in the text-to-SQL domain where datasets like WikiSQL [88] and Spider [89] are widely used as benchmarks.

WikiSQL is the predominant benchmark dataset in text-to-SQL research, featuring over 25,000 Wikipedia tables and

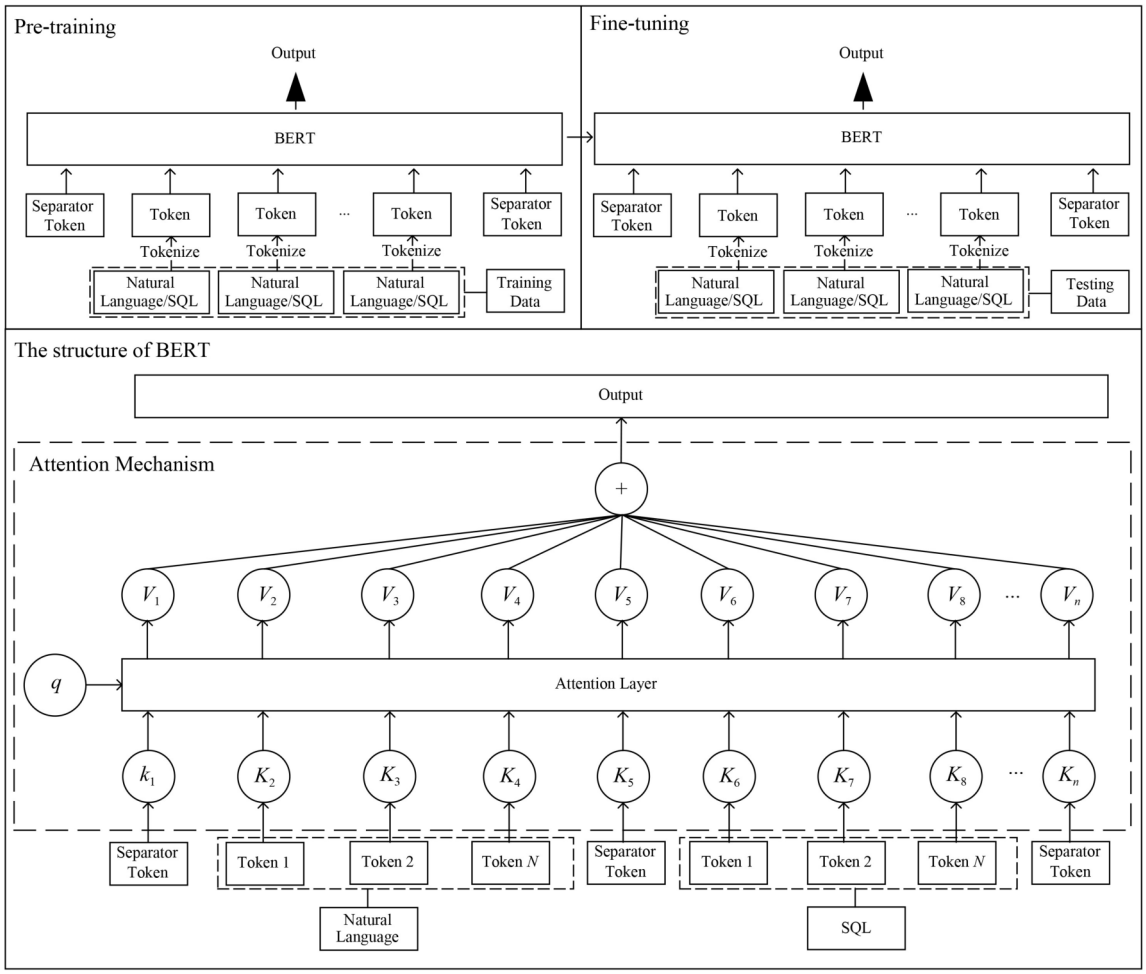


Fig. 15 An example of text-to-SQL model based on BERT

80,000 manually curated pairs of natural language questions and corresponding SQL queries [90]. Notably, WikiSQL excludes complex SQL clauses such as "INTERSECTION", "UNION", "ORDER BY", "GROUP BY", and "JOIN". Additionally, it restricts queries to single-column outputs, simplifying model training. These characteristics contribute to WikiSQL's popularity in the field.

Spider [89] is a comprehensive cross-domain dataset comprising more than 200 relational DBs across 138 diverse domains. In contrast to WikiSQL, Spider includes complex nested query clauses and a greater variety of out-of-domain (OOD) data, posing challenges for training. However, models trained on Spider tend to generalize better, making it a preferred choice for training models to handle complex SQL queries across various tasks. Researchers have extensively utilized Spider to train models capable of generating intricate SQL queries suitable for a wide array of applications. Additionally, derived datasets such as Spider-DK [91], a human-curated extension of Spider, focus on evaluating text-to-SQL models' ability to generalize to domain-specific knowledge. Spider-Syn [92], another derivative, examines the resilience of text-to-SQL models when subjected to synonym substitution, enhancing model reliability.

Furthermore, another commonly-used dataset, KaggleDBQA [93], is extracted from Kaggle, encompassing multiple domains. Although smaller in size compared to WikiSQL and Spider, it comprises a substantial amount of real-world industry data.

5.3 Model evolution

In this Subsection, we give a chronological overview of representative text-to-SQL models based on their evolutionary progression. The working mechanisms of each type of model at different evolution stages are illustrated in Fig. 16.

1. **The theoretical stage** In the field of text-to-SQL, Xu et al. [94] proposed SQLNet, an early theoretical model. SQLNet is a sketch-based synthesis approach in which the sketch comprises keywords, columns, and tables. SQLNet can transform natural language into SQL by filling in the empty spaces of the sketch with the appropriate information.
2. **The stage of theoretical development** Building on SQLNet, Min et al. [95] proposed a tree-structured model. This sequence-to-tree model employs an LSTM in the encoder and outputs a tree-structured SQL query through a decoder. The model processes tokens and transforms them into a tree structure with keywords (e.g., "SELECT", "WHERE"), table names, and column names on the tree nodes.
3. **The stage of theoretical improvement** At this stage,

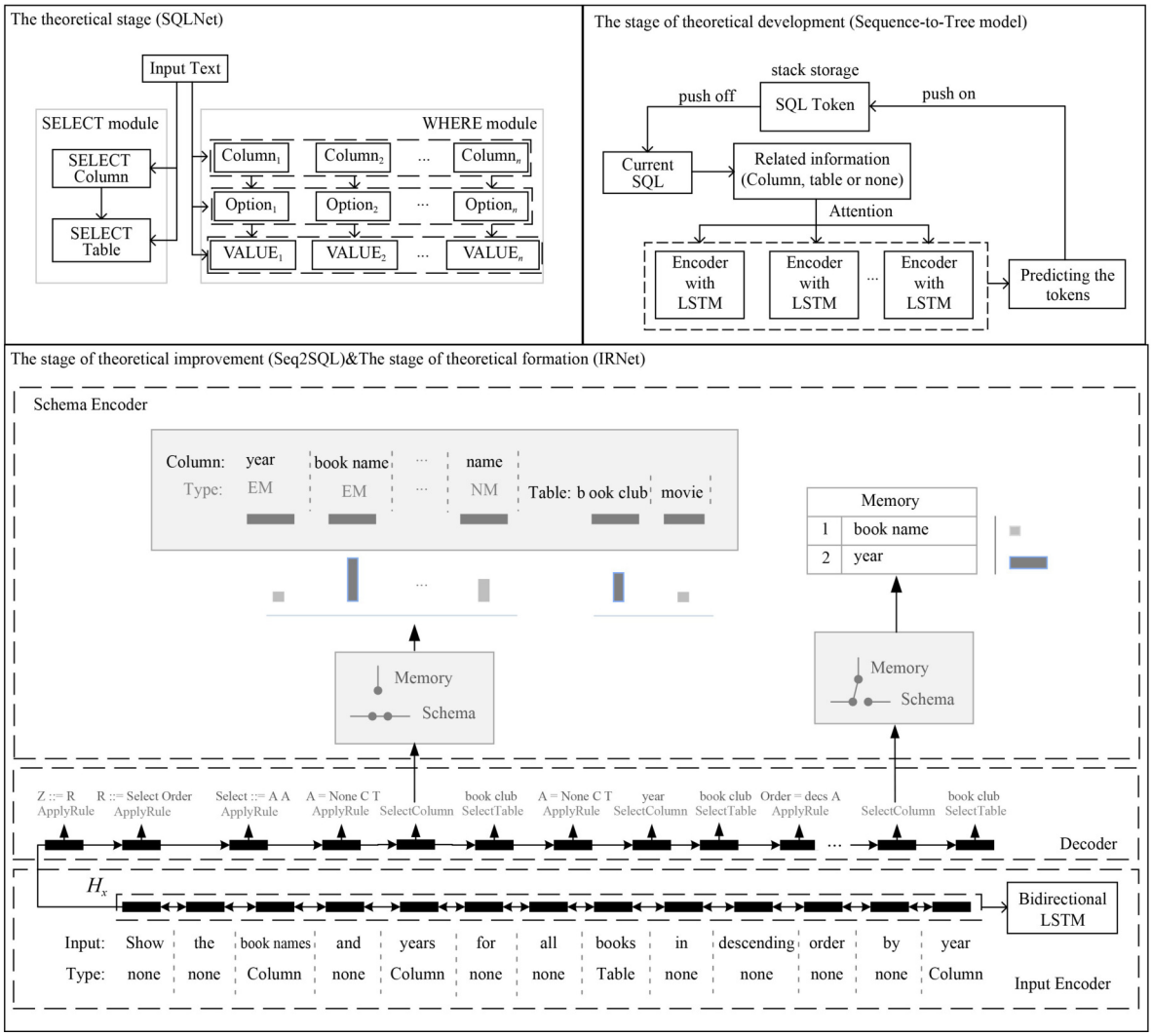


Fig. 16 The development of text-to-SQL models and their working mechanism

researchers began applying deep neural networks that build models using intermediate representations to generate a large number of queries, significantly improving computational efficiency. Kevin et al. [96] proposed the Sequence-to-SQL model, which standardizes the deep learning model. It applies a reinforcement learning approach and uses cross-entropy loss and a reward function as evaluation metrics. The model internally employs an augmented pointer network, which can change the length of the outputs, allowing for better performance. The Sequence-to-SQL model includes a classifier, a "SELECT" component, and a "WHERE" component. The classifier performs an aggregation operation on the query. An encoder-decoder architecture with a bi-directional LSTM is used inside the aggregation classifier. The "SELECT" clause distinguishes the column/table names, while the "WHERE" clause determines the SQL constraints.

4. **The stage of theoretical formation** The text-to-SQL technique has matured and now uses complex cross-domain datasets to train models. Most models are trained on the benchmark dataset WikiSQL. However, IRNet [97] uses the Spider dataset. Compared with Spider, WikiSQL contains simpler query structures without complicated nested query statements and a substantial amount of OOD data, simplifying the model training process. Conversely, IRNet is designed to handle sophisticated SQLs, making Spider a more suitable choice for training the IRNet model due to its more complex SQL statements.

IRNet consists of three key steps based on an encoder-decoder structure:

1) Schema linking. This step is intended to identify and establish connections between the columns and tables relevant to a query and the corresponding elements within the DB schema. It identifies entities such as columns, tables, and values within the input question, generating all possible n-grams ranging from one to six characters in length from the question. These n-grams are then compared to the names of columns and tables in the schema using string-matching techniques. For entities recognized as columns, there are two types of matches: an entity that exactly matches a column name in the schema is identified as an "Exact Match", while

an entity that partially matches a column name is labeled as a "Partial Match".

2) Intermediate Representation Synthesis. This step encodes the question and schema, then synthesizes a SemQL query through a neural network model based on grammar. SemQL has a set of tree structure transformation rules. The nodes in the tree use the character "Z" to indicate intersection, union, and complement operations in the DB, and if there is no such operation, it is also indicated by "Z". The character "R" indicates the "Select" keyword, where the "Select" node can be divided into one or more nodes. The "Order" node refers to descending and ascending query results. By contrast, the "Filter" node indicates filtering operations corresponding to conditional keywords such as greater than, less than, equal to, between, and not in. The character "A" represents elements that can be divided into columns and tables in the tree, with columns and tables represented by the letters "C" and "T", respectively.
3) Deterministic SQL Query Inference. This final stage involves the conversion of the SemQL query into a SQL query by mapping SemQL components to SQL and applying domain knowledge for implementation details.

STaR [98] is a model based on the attention mechanism. Compared with IRNet, STaR can handle contextually relevant textual information through two frameworks: schema state tracking (SST) and utterance dependency tracking (UDT). These frameworks enable the model to track input textual information and enrich the representation of text-to-SQL conversations, addressing contextually relevant problems effectively. SST tracks SQL queries using schema states, predicting SQL keywords for each schema slot in the current query based on the previously predicted query. This approach captures the consistency between historical and current SQL queries by introducing schema states. UDT aims to capture complex semantic dependencies in textual tasks and uses weighted learning to aggregate similar natural language (NL) while distinguishing differences in NL. It introduces two similarity measurements SQL structural similarity and SQL semantic similarity to generate appropriate pairs of NL questions. The contextual relationship evaluation model evaluates the complex semantic dependencies of sequential natural texts in each text-to-SQL sentence pair through a self-supervised approach, allowing the model to learn contextualization. It measures the similarity of SQLs using two metrics: SQL semantic similarity and SQL structural similarity. Figure 17 illustrates the structure of the STaR model.

5.4 Advances in text-to-SQL Techniques

Octavian et al. [99] proposed an encoder-decoder architecture

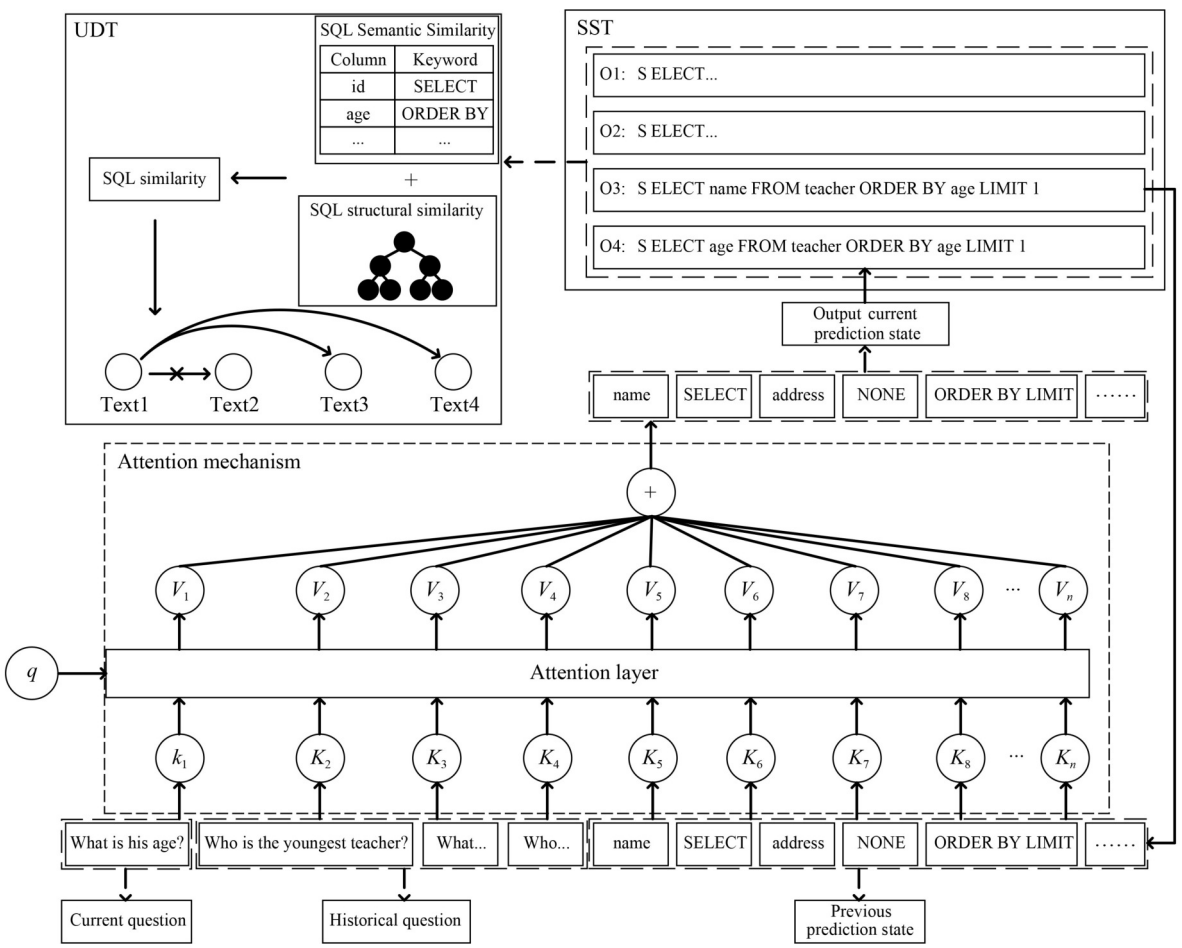


Fig. 17 The structure of STaR model

that integrates data augmentation and multi-model integration methods. This architecture extracts specific information from the input problem, linking tokens to specific tables and columns in a DB. Instead of using the commonly employed Named Entity Recognition (NER) framework in encoders, they replaced it with a Disambiguation Dictionary Module (DDM) to provide key information for the model. The system offers the following advantages: 1) the integration of a seed training data augmentation (STDA) technique to generate a larger training set, and 2) the use of different auto-generated training datasets to train distinct models, employing hybrid models to analyze results by combining multiple models.

Li et al. [100] proposed WAGG for text-to-SQL in aggregate tables, which uses a dynamic pruning strategy to eliminate irrelevant entries in multiple aggregated tables, significantly reducing the time cost of model training. Aggregation is a data batch processing operation that groups data and performs multiple batch operations on each group. An aggregate table is a table processed by aggregation, commonly used in query operations. Currently, there are relatively few studies focused on text-to-SQL for aggregate tables, which face two main challenges: 1) aggregate tables have more complex mapping relationships when converting text to DB statements, and 2) current deep learning models incur significant time costs when processing aggregate tables. WAGG allows the aggregate table to be used as input to the model, with complex DB tables fed into the model for training.

Wei et al. [101] proposed a sharing mechanism within the decoder for multi-task learning, enabling various sub-tasks to utilize the same decoder and effectively exchange knowledge during the training process. This approach allows the model to acquire knowledge of the interdependencies among sub-tasks. When a text-to-SQL model is structured on the encoder-decoder architecture, the decoder often faces limitations in its capacity to comprehend the correlation between various sub-tasks, and training costs increase with complex decoders. The proposed approach overcomes these problems, ensuring effective model training while avoiding excessive complexity. Additionally, Tomer et al. [102] designed a weakly supervised learning-based method, called Odmrs, to train a parser for text-to-SQL statements. This method does not rely on manually labeled high-quality data but uses data provided by non-specialist users for training, demonstrating the proposed model's strong generalization capabilities.

Geunyeong et al. [103] designed a hybrid decoder for SQL generation, which includes self-defined DB statement components needed for executing each query. The hybrid decoder can generate SQL query statements sequentially. Qi et al. [104] developed a Transformer-based architecture that utilizes the self-attention mechanism in Transformer to recognize table-to-table relationships. This approach introduces relational structures such as schema joins and schema encodings into the model to generate more logical SQLs. Xu et al. [105] introduced a RNN-based methodology named SeaD, which integrates an autoregressive model with a sequence-to-sequence model. It addresses the limitations of the decoding process in the model and enhances the accuracy of converting text into SQL statements. Qin et al. [106] designed a neural network called "Sun" that uses heuristic constraint rules to reduce model outputs, significantly improving generalization and stability. Peng et al. [107] addressed semantic parsing from text to SQLs by applying a transfer learning-based model called XRICL (Transfer Learning in Cross-Language). This model utilizes training results based on English datasets and applies them to other language models. Pi et al. [108] created a system called Adveta for testing the robustness of text-to-SQL models using ATP (Adversarial Table Perturbation) metrics. The system's internal structure is mainly an adversarial generative framework that monitors and improves model robustness.

Han et al. [109] developed a graph-based method called RuleSQLova, suitable for handling aggregation operators in DBs. Zheng et al. [110] proposed a schema-connected graph-based method named HIE-SQL, which enhances the connection between input texts and output SQLs. Xiao et al. [111] introduced the CQR (Conversation Question Reformulation) method based on contextually related text sequence questions. This method uses recursively augmented schemas to generate text-to-SQL intermediate representations, allowing the model to understand contextual semantics and enhance SQL parsing. Wang et al. [112] designed a method called Proton is founded on a pre-trained language model to induce the parser to process input texts. This unsupervised learning model operates without pre-processing the input texts. Abhijeet et al. [113] created a framework called REFILL for synthesizing high-quality datasets. The REFILL framework adds textual queries from existing schemas, greatly improving the efficiency of model training. Chen et al. [114] proposed a baseline modifier called SQuALL for segmentation during pre-training, which consists of schema expansion and schema pruning. During model training, it synthesizes and segments the input data. Gyubok et al. [115] designed a dataset called EHRs (Electronic Health Records) tailored for hospital application scenarios in model training. Table 3 presents the latest research and comparisons of these text-to-SQL models.

6 Future directions

6.1 Cardinality estimation

Cardinality estimation models are rapidly evolving, driven by advancements in machine learning algorithms and their integration into DB management systems. Recent studies, such as the Robust Cardinality model proposed by Praciano et al. [116], demonstrate the potential of models that adaptively update in response to data changes. These models are crucial for accurate query optimization as data characteristics fluctuate over time.

Furthermore, the application of complex models like TreeLSTM [40] indicates a significant trend toward handling intricate query structures involving multiple tables and conditions. This kind of model shows an improved capability compared to traditional methods, particularly in capturing the correlations across tables.

The BoundEst framework [117] introduced an innovative approach to optimize join cardinalities with tight upper

Table 3 Summary of the SOAT models

Categary	Orientation	Method	Key technology	Function	Training dataset	OOD	Generalization
Design model	Supervised learning	Data enhancement and model integration system	Data enhancement	Improving entity recognition accuracy	Spider	✓	✓
		Odmrs	Weakly supervised learning	Optimizing the parser	WikiSQL	✓	×
		Hybrid decoder	Combining two decoding methods	Improving the efficiency of generating SQLs	Spider	✓	✓
		Rasat	Self-attention mechanism	Generating reasonable SQLs	Spider	✓	✓
		SeaD	Autoregressive sequence-to-sequence model	Improving the accuracy of conversion	WikiSQL	✓	×
	Unsupervised learning	Sun	Model uncertainty constraints	Improving model robustness	Spider	×	×
		XRICL	Semantic parsing	Parsing reasonable contexts	XSpider&XKaggle	✓	✓
		Adveta	Against table chaos	Evaluating model robustness	Spider&WikiSQL	✓	×
	Based on graph	RuleSQLova	Setting decoding policy	Distinguishing aggregation operators	WikiSQL	✓	✓
		HIE-SQL	Pattern connection diagram	Improving pattern connection accuracy	SPARC&COSQL	×	✓
Design parser	Reinforcement learning	WAGG	Dynamic pruning	Adaptive aggregation table	SipderWAGG	✓	✓
		Model enhancer	Multi-task learning and sharing mechanism	Reducing model complexity	WikiSQL	×	×
	Deep learning	CQRSQL	Recursive enhancement	Auxiliary parsing SQLs	SPARC&COSQL	✓	✓
		Proton	Pattern recognition	Auxiliary DB schema connection	WikiSQL	×	✓
Generating datasets	Transfer learning	Refill	Data enhancement	Synthesizing high-quality parallel DB	/	✓	✓
	Deep learning	Benchmark modifier	Pattern expansion & Trimming	Synthesizing and segmenting training datasets	/	×	✓
		EHRSQL	Trustworthy semantic parsing	Constructing training datasets in medical fields	/	×	✓

bounds, which is important for enhancing the efficiency of SQL queries involving joins. This methodology reflects a broader trend towards accuracy of estimation, which is critical for the optimization of complex queries.

Additionally, the development of new frameworks like CELA, which emphasize strong generalization abilities and dimensional adaptability, reflects the requirements in the industry towards model performance that is not only accurate but also robust across different DB schemas and configurations [118]. These models are expected to work well in reducing the errors in cardinality estimation.

The research development in this field suggests that future cardinality estimation models will likely leverage techniques such as AutoML and online learning to further automate and refine the accuracy of their predictions. The shift from manual to automated design of network architectures in machine learning models could potentially influence future strategies in cardinality estimation, making it more efficient with fewer manual errors.

The continuous refinement of these technologies can significantly enhance the capabilities of DB systems, leading to more sophisticated, efficient, and adaptive query optimization strategies.

6.2 Cost estimation

Recent advancements in learning-based methods for DB cost estimation are changing the research directions on DB management systems. For instance, the Database-Agnostic Cost Estimator (DACE) [119] represents a pivotal shift towards models that can adaptively estimate costs across different DB systems without needing system-specific adjustments [120]. Similarly, the Bidirectional Compressor and Ensemble Networks (BICE) [121] framework leverages deep learning to refine cost and cardinality estimations simultaneously, demonstrating the potential of machine learning models to handle complex DB queries with improved accuracy as well as reducing computational overhead.

A promising solution to the trade-off between training time and accuracy is applying few-shot learning and transfer learning techniques [122]. Pre-trained models for cost estimation could be fine-tuned on specific query workloads with minimal data. This would allow cost estimation models to be effectively generalized across diverse workloads without requiring extensive retraining.

In the future, we may see more hybrid models that combine traditional cost estimation methods with deep learning approaches [120]. For instance, rule-based methods could handle simpler queries, while machine learning models could be selectively applied to more complex queries while traditional methods cannot be used.

6.3 Join order selection

The technique of learning-based models for join order optimization is rapidly growing, showing a significant trend toward automated, intelligent DB management systems. For instance, the ReJOOSp model (Reinforcement learning for Join Order Optimization in SPARQL) [123] demonstrates how machine learning, specifically reinforcement learning, can be effectively applied to optimize complex SPARQL queries. This approach employs historical data to train the models in order to efficiently predict join orders, thereby potentially reducing the computational overhead and accelerating the execution of queries.

Further, the Dynamic Double Deep Q-Network (called DDQN) offers a sophisticated example of how deep reinforcement learning can be tailored to address traditional

challenges in join order optimization, such as action value overestimation. This model utilizes a double Q-learning framework to enhance the stability and reliability of the learning process, enabling the optimizer to efficiently exploit the search space of possible join orders without extensive manual tuning operations [124].

Additionally, the Coral project [125] integrates a deep reinforcement learning model into federated query processing and optimizes join orders across DBs. This study focuses on the adaptability of learning-based models to different DB architectures and distributed environments, suggesting that such techniques can effectively handle the complexity introduced by diverse data sources.

The state-of-the-art indicates a promising direction for the future of DB query optimization, where learning-based models are expected to play a pivotal role. The ongoing development in this field suggests that more sophisticated, automated solutions for join order optimization will continue to be proposed, aiming to reduce manual configuration and improve the query performance of DB systems.

6.4 End-to-end optimizer

Recently, Li et al. [126] proved that effective end-to-end optimizer architectures are crucial for enhancing DB performance from different aspects. This work indicates that it is promising to explore the cost-effective optimizer architecture. Future developments focus on a new architecture that is inspired by state-of-the-art machine learning techniques, which is probably designed to achieve better performance with less computational complexity. Furthermore, Zhou et al. [127] demonstrated the potential of deep learning models to significantly improve the efficiency of query processing and optimization through deep understanding and context awareness.

The end-to-end optimizers utilize reinforcement learning and deep neural networks, suggesting a future direction in which DBs can be dynamically adapted to changing workloads and query complexity without manual intervention. The learning-based models in DB systems are expected to provide more sophisticated and integrated solutions that can manage the entire lifecycle of data operations effectively and efficiently.

As machine learning techniques develop rapidly, further innovations in DB optimizer architectures are expected. These advancements may rely on automatic methods for optimizing DB operations, which could exceed the capabilities of manually designed optimization strategies, providing better system performance with lower computational costs and higher usability.

6.5 Text-to-SQL

As large model techniques become popular, an increasing number of machine learning methods are being proposed, thereby bringing new possibilities in the AI-for-DB (combining AI with databases) research field, for example, text-to-SQL models.

Text-to-SQL models still face several challenges in real-world applications. A key issue is their lack of robustness when the input data are fluctuant. Even when the training and testing data distributions are similar, small perturbations in query phrasing or data schemas can lead to significant performance degradation. Consequently, this sensitivity to input variations limits the practical application of text-to-SQL models in scenarios where query structures and data schemas frequently change [92].

To address these challenges, a potential solution for improving the robustness of text-to-SQL models is data augmentation techniques [128]. By manually introducing variations into the training data, such as rephrasing queries or making slight modifications to the schema, the models can handle a broader range of input data. Consequently, this approach could reduce the sensitivity to minor perturbations and enhance their generalization capability across different queries.

7 Conclusion

This survey introduces learning-based DB optimization techniques, covering cardinality/cost estimation, join order selection, end-to-end optimizers, and text-to-SQL. It discusses both traditional methods and machine learning-based approaches in these areas. Additionally, it explores future research directions for each of these domains.

The survey offers an in-depth analysis of the mechanisms involved in cardinality/cost estimation, join order selection, end-to-end optimizers, and text-to-SQL models. Notably, text-to-SQL is positioned within the AI-for-DB research domain, offering valuable insights to researchers aiming to deepen their understanding of this rapidly evolving field of DB optimization.

In conclusion, cardinality/cost estimation and join order selection are prime candidates for integration with machine learning techniques. Developing a coexistent method for end-to-end optimizers alongside machine learning remains a key challenge. Combining these techniques with machine learning presents a promising avenue for advancing DB efficiency.

Acknowledgments This work was partially supported by the National Natural Science Foundation of China (Grant No. 62272066); Open Research Fund of Guangxi Key Lab of Human-machine Interaction and Intelligent Decision (GXHIID2207); Sichuan Science and Technology Program (2025ZNSFSC0044, 2025YFHZ0194); Chengdu Technological Innovation Research and Development Project (2024-YF05-01217-SN); Chengdu Regional Science and Technology Innovation Cooperation Project (2025-YF11-00050-HZ); Open Foundation of Key Laboratory of Cyberspace Security, Ministry of Education of China and Henan Key Laboratory of Cyberspace Situation Awareness (KLCS20240106); Ant Group through CCF-Ant Research Fund (CCF-AFSG RF20240106); Open Research Fund of Key Laboratory of Cyberspace Big Data Intelligent Security (Chongqing University of Posts and Telecommunications), Ministry of Education of China (CBDIS202404).

Competing interests The authors declare that they have no competing interests or financial conflicts to disclose.

References

1. Zhou X, Chai C, Li G, Sun J. Database meets artificial intelligence: a survey. IEEE Transactions on Knowledge and Data Engineering, 2022, 34(3): 1096−−1116
2. Zhou X, Chai C, Li G, Sun J. Database meets artificial intelligence: a survey (extended abstract). In: Proceedings of the 39th IEEE International Conference on Data Engineering. 2023, 3901–3902
3. Li X, Zhu Y, Xu R, Wang J, Zhang Y. Indexing dynamic encrypted database in cloud for efficient secure *k*-nearest neighbor query. Frontiers of Computer Science, 2024, 18(1): 181803
4. Li G, Zhou X, Cao L. Machine learning for databases. Proceedings of the VLDB Endowment, 2021, 14(12): 3190−3193
5. Müller M, Woltmann L, Lehner W. Enhanced featurization of queries with mixed combinations of predicates for ML-based cardinality estimation. In: Proceedings of the 26th International Conference on Extending Database Technology (EDBT 2023). 2023, 273–284
6. Wu Y, Hu Z, Wang Y, Min F. Rare potential poor household identification with a focus embedded logistic regression. IEEE Access, 2022, 10: 32954−32972
7. Yu Z, Zhang C, Xiong N, Chen F. A new random forest applied to heavy metal risk assessment. Computer Systems Science and Engineering, 2022, 40(1): 207−221
8. Zhou W, Zhan S, Dai B, Guo L. SOAR: A learned join order selector with graph attention mechanism. In: Proceedings of 2022 International Joint Conference on Neural Networks. 2022, 1–8
9. Jiang J, Wen Z, Wang Z, He B, Chen J. Parallel and distributed structured SVM training. IEEE Transactions on Parallel and Distributed Systems, 2022, 33(5): 1084−1096
10. Arunkumar A, Surendran D. Autism spectrum disorder diagnosis using ensemble ML and max voting techniques. Computer Systems Science and Engineering, 2022, 41(1): 389−404
11. Cui Y, Zhang D, Zhang J, Zhang T, Cao L, Chen L. Multi-user reinforcement learning based task migration in mobile edge computing. Frontiers of Computer Science, 2024, 18(4): 184504
12. Li Y, Wang L, Wang S, Sun Y, Peng Z. A resource-aware deep cost model for big data query processing. In: Proceedings of the 38th IEEE International Conference on Data Engineering. 2022, 885–897
13. Wu J, Qu L, Yang G, Han N. Diabetes induced factors prediction based on various improved machine learning methods. Current Bioinformatics, 2022, 17(3): 254−262
14. Gallinucci E, Golfarelli M. SparkTune: Tuning spark SQL through query cost modeling. In: Proceedings of the 22nd International Conference on Extending Database Technology. 2019, 546–549
15. Ortiz J, Balazinska M, Gehrke J, Keerthi S S. Learning state representations for query optimization with deep reinforcement learning. In: Proceedings of the 2nd Workshop on Data Management for End-To-End Machine Learning. 2018, 4
16. Li G, Zhou X, Li S, Gao B. QTune: A query-aware database tuning system with deep reinforcement learning. Proceedings of the VLDB Endowment, 2019, 12(12): 2118−2130
17. Li G, Zhou X, Sun J, Yu X, Han Y, Jin L, Li W, Wang T, Li S. openGauss: An autonomous database system. Proceedings of the VLDB Endowment, 2021, 14(12): 3028−3041
18. Chen J, Chen Y, Chen Z, Ghazal A, Li G, Li S, Ou W, Sun Y, Zhang M, Zhou M. Data management at Huawei: Recent accomplishments and future challenges. In: Proceedings of the 35th IEEE International Conference on Data Engineering. 2019, 13–24
19. Zhou X, Sun J, Li G, Feng J. Query performance prediction for concurrent queries using graph embedding. Proceedings of the VLDB Endowment, 2020, 13(9): 1416−1428
20. Zhang J, Wu S, Zhao J, Xie Z, Li F, Gao Y, Chen G. A sampling-based learning framework for big databases. In: Proceedings of the ACM Web Conference 2022. 2022, 1871–1881
21. Lan H, Bao Z, Peng Y. A survey on advancing the DBMS query optimizer: Cardinality estimation, cost model, and plan enumeration. Data Science and Engineering, 2021, 6(1): 86−101
22. Qiao S J, Yang G P, Han N, Chen H, Huang F L, Yue K, Yi Y G, Yuan C A. Cardinality estimator: Processing SQL with a vertical scanning convolutional neural network. Journal of Computer Science and Technology, 2021, 36(4): 762−777
23. Yu X, Li G, Chai C, Tang N. Reinforcement learning with tree-LSTM for join order selection. In: Proceedings of the 36th IEEE International Conference on Data Engineering. 2020, 1297–1308
24. Marcus R C, Negi P, Mao H, Zhang C, Alizadeh M, Kraska T, Papaemmanouil O, Tatbul N. Neo: A learned query optimizer. Proceedings of the VLDB Endowment, 2019, 12(11): 1705−1718
25. Wu C, Jindal A, Amizadeh S, Patel H, Le W, Qiao S, Rao S. Towards a learning optimizer for shared clouds. Proceedings of the VLDB Endowment, 2018, 12(3): 210−222
26. Li L, Guan J, Peng X, Zhou L, Zhang Z, Ding L, Zheng L, Wu L, Hu Z, Liu L, Yao Y. Machine learning for the prediction of 1-year mortality in patients with sepsis-associated acute kidney injury. BMC Medical Informatics and Decision Making, 2024, 24(1): 208
27. Amiriparian S, Pugachevskiy S, Cummins N, Hantke S, Pohjalainen J, Keren G, Schuller B W. CAST a database: Rapid targeted large-scale big data acquisition via small-world modelling of social media platforms. In: Proceedings of 2017 Seventh International Conference on Affective Computing and Intelligent Interaction. 2017, 340–345
28. Akbari Y, Kunhoth J, Elharrouss O, Al-Maadeed S, Abualsaud K, Mohamed A, Khattab T. Indoor multi-lingual scene text database with different views. In: Proceedings of 2023 International Symposium on Networks, Computers and Communications. 2023, 1–6
29. Akhter R, Sofi S A. Precision agriculture using IoT data analytics and machine learning. Journal of King Saud University-Computer and Information Sciences, 2022, 34(8): 5602−5618
30. Zhou H, Qian W, Zhou X, Dong Q, Zhou A, Tan W. Scalable and adaptive log manager in distributed systems. Frontiers of Computer Science, 2023, 17(2): 172205
31. Xiong X, Qiao S, Han N, Xiong F, Bu Z, Li R, Yue K, Yuan G. Where to go: An effective point-of-interest recommendation framework for heterogeneous social networks. Neurocomputing, 2020, 373: 56−69
32. Li G L, Zhou X H, Sun J, Yu X, Yuan H-T, Liu J-B, Han Y. A survey of machine learning based database techniques. Chinese Journal of Computers, 2020, 43(11): 2019−2049
33. Al-Azani S, Sait S M, Al-Utaibi K A. A comprehensive literature review on children's databases for machine learning applications. IEEE Access, 2022, 10: 12262−12285
34. Fantinato M, Peres S M, Kafeza E, Chiu D K W, Hung P C K. A review on the integration of deep learning and service-oriented architecture. Journal of Database Management, 2021, 32(3): 95−119
35. Wu Z, Negi P, Alizadeh M, Kraska T, Madden S. FactorJoin: A new cardinality estimation framework for join queries. Proceedings of the ACM on Management of Data, 2023, 1(1): 41
36. Qiu Y, Wang Y, Yi K, Li F, Wu B, Zhan C. Weighted distinct sampling: Cardinality estimation for SPJ queries. In: Proceedings of 2021 International Conference on Management of Data. 2021,

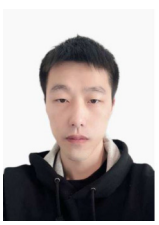
1465–1477

37. Durand M, Flajolet P. Loglog counting of large cardinalities (Extended abstract). In: Proceedings of the 11th Annual European Symposium on Algorithms. 2003, 605–617
38. Ioannidis Y E. The history of histograms (abridged). In: Proceedings of the 29th International Conference on Very Large Data Bases. 2003, 19–30
39. Zhang M, Wang H. Selectivity estimation with density-model-based multidimensional histogram. Knowledge and Information Systems, 2021, 63(4): 971−992
40. Qi K, Yu J, He Z. A cardinality estimator in complex database systems based on TreeLSTM. Sensors, 2023, 23(17): 7364
41. Park Y, Zhong S, Mozafari B. QuickSel: Quick selectivity learning with mixture models. In: Proceedings of 2020 ACM SIGMOD International Conference on Management of Data. 2020, 1017–1033
42. Dutt A, Wang C, Nazi A, Kandula S, Narasayya V, Chaudhuri S. Selectivity estimation for range predicates using lightweight models. Proceedings of the VLDB Endowment, 2019, 12(9): 1044−1057
43. Woltmann L, Hartmann C, Thiele M, Habich D, Lehner W. Cardinality estimation with local deep learning models. In: Proceedings of the 2nd International Workshop on Exploiting Artificial Intelligence Techniques for Data Management. 2019, 5
44. Kipf A, Kipf T, Radke B, Leis V, Boncz P A, Kemper A. Learned cardinalities: Estimating correlated joins with deep learning. In: Proceedings of the 9th Biennial Conference on Innovative Data Systems Research. 2019
45. Tzoumas K, Deshpande A, Jensen C S. Lightweight graphical models for selectivity estimation without independence assumptions. Proceedings of the VLDB Endowment, 2011, 4(11): 852−863
46. Sun R, Tao H, Chen Y, Liu Q. HACAN: A hierarchical answer-aware and context-aware network for question generation. Frontiers of Computer Science, 2024, 18(5): 185321
47. Marcus R, Papaemmanouil O. Plan-structured deep neural network models for query performance prediction. Proceedings of the VLDB Endowment, 2019, 12(11): 1733−1746
48. Sun J, Li G. An end-to-end learning-based cost estimator. Proceedings of the VLDB Endowment, 2019, 13(3): 307−319
49. Li P, Hua Y, Jia J, Zuo P. FINEdex: A fine-grained learned index scheme for scalable and concurrent memory systems. Proceedings of the VLDB Endowment, 2021, 15(2): 321−334
50. Heimel M, Kiefer M, Markl V. Self-tuning, GPU-accelerated kernel density models for multidimensional selectivity estimation. In: Proceedings of 2015 ACM SIGMOD International Conference on Management of Data. 2015, 1477–1492
51. Ma J, Fan A, Jiang X, Xiao G. Feature matching via motion-consistency driven probabilistic graphical model. International Journal of Computer Vision, 2022, 130(9): 2249−2264
52. Hilprecht B, Schmidt A, Kulessa M, Molina A, Kersting K, Binnig C. DeepDB: Learn from data, not from queries! Proceedings of the VLDB Endowment, 2020, 13(7): 992−1005
53. Chow C, Liu C. Approximating discrete probability distributions with dependence trees. IEEE Transactions on Information Theory, 1968, 14(3): 462−467
54. Wu T, Qian H, Liu Z, Zhou J, Zhou A. Bi-objective evolutionary Bayesian network structure learning via skeleton constraint. Frontiers of Computer Science, 2023, 17(6): 176350
55. Zhu R, Wu Z, Han Y, Zeng K, Pfadler A, Qian Z, Zhou J, Cui B. FLAT: Fast, lightweight and accurate method for cardinality estimation. Proceedings of the VLDB Endowment, 2021, 14(9): 1489−1502
56. Wang J, Chai C, Liu J, Li G. FACE: A normalizing flow based cardinality estimator. Proceedings of the VLDB Endowment, 2021, 15(1): 72−84
57. Wang X, Qu C, Wu W, Wang J, Zhou Q. Are we ready for learned cardinality estimation? Proceedings of the VLDB Endowment, 2021, 14(9): 1640−1654
58. Sun J, Zhang J, Sun Z, Li G, Tang N. Learned cardinality estimation: A design space exploration and A comparative evaluation. Proceedings of the VLDB Endowment, 2021, 15(1): 85−97
59. Sandell J, Asplund E, Ayele W Y, Duneld M. Performance comparison analysis of ArangoDB, MySQL, and Neo4j: An experimental study of querying connected data. In: Proceedings of the 57th Hawaii International Conference on System Sciences. 2024, 7760–7769
60. Lew D J, Yoo K, Nam K W. DeepVQL: Deep video queries on PostgreSQL. Proceedings of the VLDB Endowment, 2023, 16(12): 3910−3913
61. Xu Y, Zhang D, Zhang S, Wu S, Feng Z, Chen G. Predictive and near-optimal sampling for view materialization in video databases. Proceedings of the ACM on Management of Data, 2024, 2(1): 19
62. Wei J, Suriawinata A, Ren B, Liu X, Lisovsky M, Vaickus L, Brown C, Baker M, Tomita N, Torresani L, Wei J, Hassanpour S. A petri dish for histopathology image analysis. In: Proceedings of the 19th International Conference on Artificial Intelligence in Medicine. 2021, 11–24
63. Han Y, Wu Z, Wu P, Zhu R, Yang J, Tan L W, Zeng K, Cong G, Qin Y, Pfadler A, Qian Z, Zhou J, Li J, Cui B. Cardinality estimation in DBMS: A comprehensive benchmark evaluation. Proceedings of the VLDB Endowment, 2021, 15(4): 752−765
64. He Z, Lee B S, Snapp R R. Self-tuning UDF cost modeling using the memory-limited quadtree. In: Proceedings of the 9th International Conference on Extending Database Technology on Advances in Database Technology. 2004, 513–531
65. He Z, Lee B S, Snapp R. Self-tuning cost modeling of user-defined functions in an object-relational DBMS. ACM Transactions on Database Systems (TODS), 2005, 30(3): 812−853
66. Liu F, Blanas S. Forecasting the cost of processing multi-join queries via hashing for main-memory databases. In: Proceedings of the 6th ACM Symposium on Cloud Computing. 2015, 153–166
67. Leis V, Radke B, Gubichev A, Mirchev A, Boncz P, Kemper A, Neumann T. Query optimization through the looking glass, and what we found running the join order benchmark. The VLDB Journal, 2018, 27(5): 643−668
68. Siddiqui T, Jindal A, Qiao S, Patel H, Le W. Cost models for big data query processing: Learning, retrofitting, and our findings. In: Proceedings of 2020 ACM SIGMOD International Conference on Management of Data. 2020, 99–113
69. Ioannidis Y E, Kang Y C. Left-Deep vs. Bushy Trees: An analysis of strategy spaces and its implications for query optimization. In: Proceedings of 1991 ACM SIGMOD International Conference on Management of Data. 1991, 168–177
70. Bennett K, Ferris M C, Ioannidis Y E. A genetic algorithm for database query optimization. In: Proceedings of the 4th International Conference on Genetic Algorithms. 1991, 400–407
71. Stillger M, Lohman G M, Markl V, Kandil M. LEO - DB2's LEarning optimizer. In: Proceedings of the 27th International Conference on Very Large Data Bases. 2001, 19–28
72. Marcus R, Papaemmanouil O. Deep reinforcement learning for join order enumeration. In: Proceedings of the First International Workshop on Exploiting Artificial Intelligence Techniques for Data Management. 2018, 3
73. Song X, Huang H, Lian J, Jin H. XGCN: a library for large-scale graph neural network recommendations. Frontiers of Computer Science,

2024, 18(3): 183343
74. Zhang J. AlphaJoin: Join order selection à la AlphaGo. In: Proceedings of the VLDB 2020 PhD Workshop. 2020
75. Avnur R, Hellerstein J M. Eddies: Continuously adaptive query processing. In: Proceedings of 2000 ACM SIGMOD International Conference on Management of Data. 2000, 261–272
76. Trummer I, Wang J, Wei Z, Maram D, Moseley S, Jo S, Antonakakis J, Rayabhari A. SkinnerDB: Regret-bounded query evaluation via reinforcement learning. ACM Transactions on Database Systems (TODS), 2021, 46(3): 9
77. Krishnan S, Yang Z, Goldberg K, Hellerstein J M, Stoica I. Learning to optimize join queries with deep reinforcement learning. 2018, arXiv preprint arXiv: 1808.03196
78. Markl V, Lohman G M, Raman V. LEO: An autonomic query optimizer for DB2. IBM Systems Journal, 2003, 42(1): 98−106
79. Marcus R, Negi P, Mao H, Tatbul N, Alizadeh M, Kraska T. Bao: Making learned query optimization practical. In: Proceedings of 2021 International Conference on Management of Data. 2021, 1275–1288
80. Negi P, Interlandi M, Marcus R, Alizadeh M, Kraska T, Friedman M T, Jindal A. Steering query optimizers: A practical take on big data workloads. In: Proceedings of 2021 International Conference on Management of Data. 2021, 2557–2569
81. Zhang C, Li Y, Zhang R, Qian W, Zhou A. Scalable and quantitative contention generation for performance evaluation on OLTP databases. Frontiers of Computer Science, 2023, 17(2): 172202
82. Zhou S, Li J, Wang H, Shang S, Han P. GRLSTM: Trajectory similarity computation with graph-based residual LSTM. In: Proceedings of the 37th AAAI Conference on Artificial Intelligence. 2023, 4972–4980
83. Chen Y, Cao H, Zhou Y, Liu Z, Chen D, Zhao J, Shi J. A GCN-GRU based end-to-end LEO satellite network dynamic topology prediction method. In: Proceedings of 2023 IEEE Wireless Communications and Networking Conference. 2023, 1–6
84. Chatzianastasis M, Lutzeyer J F, Dasoulas G, Vazirgiannis M. Graph ordering attention networks. In: Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence. 2023, 7006–7014
85. Zhao W, Hu H, Zhou W, Shi J, Li H. BEST: BERT pre-training for sign language recognition with coupling tokenization. In: Proceedings of the 37th AAAI Conference on Artificial Intelligence. 2023, 3597–3605
86. Chen L, Huang H, Chen D. Join cardinality estimation by combining operator-level deep neural networks. Inf. Sci., 2021, 546: 1047−1062
87. Dey S, Vinayakarao V, Gupta M, Dechu S. Evaluating commit message generation: To BLEU or not to BLEU? In: Proceedings of the 44th ACM/IEEE International Conference on Software Engineering: New Ideas and Emerging Results. 2022, 31–35
88. Yavuz S, Gür I, Su Y, Yan X. What it takes to achieve 100% condition accuracy on WikiSQL. In: Proceedings of 2018 Conference on Empirical Methods in Natural Language Processing. 2018, 1702–1711
89. Yu T, Zhang R, Yang K, Yasunaga M, Wang D, Li Z, Ma J, Li I, Yao Q, Roman S, Zhang Z, Radev D. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In: Proceedings of 2018 Conference on Empirical Methods in Natural Language Processing. 2018, 3911–3921
90. Salimzadeh S, Gadiraju U, Hauff C, van Deursen A. Exploring the feasibility of crowd-powered decomposition of complex user questions in text-to-SQL tasks. In: Proceedings of the 33rd ACM Conference on Hypertext and Social Media. 2022, 154–165
91. Gan Y, Chen X, Purver M. Exploring underexplored limitations of cross-domain text-to-SQL generalization. In: Proceedings of 2021 Conference on Empirical Methods in Natural Language Processing. 2021, 8926–8931
92. Gan Y, Chen X, Huang Q, Purver M, Woodward J R, Xie J, Huang P. Towards robustness of text-to-SQL models against synonym substitution. In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). 2021, 2505–2515
93. Lee C H, Polozov O, Richardson M. KaggleDBQA: Realistic evaluation of text-to-SQL parsers. In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). 2021, 2261–2273
94. Xu X, Liu C, Song D. SQLNet: Generating structured queries from natural language without reinforcement learning. 2017, arXiv preprint arXiv: 1711.04436
95. Min Q, Shi Y, Zhang Y. A pilot study for Chinese SQL semantic parsing. In: Proceedings of 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). 2019, 3652–3658
96. Stower K, Krechel D. Seq2SQL - evaluating different deep learning architectures using word embeddings. In: Proceedings of Machine Learning and Data Mining in Pattern Recognition: 15th International Conference on Machine Learning and Data Mining. 2019, 343–354
97. Guo J, Zhan Z, Gao Y, Xiao Y, Lou J G, Liu T, Zhang D. Towards complex text-to-SQL in cross-domain database with intermediate representation. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. 2019, 4524–4535
98. Cai Z, Li X, Hui B, Yang M, Li B, Li B, Cao Z, Li W, Huang F, Si L, Li Y. STAR: SQL guided pre-training for context-dependent text-to-SQL parsing. In: Proceedings of Findings of the Association for Computational Linguistics: EMNLP 2022. 2022, 1235–1247
99. Popescu O, Manotas I, Vo N P A, Yeo H, Khorashani E, Sheinin V. Addressing limitations of encoder-decoder based approach to text-to-SQL. In: Proceedings of the 29th International Conference on Computational Linguistics. 2022, 1593–1603
100. Li S, Zhou K, Zhuang Z, Wang H, Ma J. Towards text-to-SQL over aggregate tables. Data Intelligence, 2023, 5(2): 457−474
101. Wei C, Huang S, Li R. Enhance text-to-SQL model performance with information sharing and reweight loss. Multimedia Tools and Applications, 2022, 81(11): 15205−15217
102. Wolfson T, Deutch D, Berant J. Weakly supervised text-to-SQL parsing through question decomposition. In: Proceedings of Findings of the Association for Computational Linguistics: NAACL 2022. 2022, 2528–2542
103. Jeong G, Han M, Kim S, Lee Y, Lee J, Park S, Kim H. Improving text-to-SQL with a hybrid decoding method. Entropy, 2023, 25(3): 513
104. Qi J, Tang J, He Z, Wan X, Cheng Y, Zhou C, Wang X, Zhang Q, Lin Z. RASAT: Integrating relational structures into pretrained Seq2Seq model for text-to-SQL. In: Proceedings of 2022 Conference on Empirical Methods in Natural Language Processing. 2022, 3215–3229
105. Xu K, Wang Y, Wang Y, Wang Z, Wen Z, Dong Y. SeaD: End-to-end text-to-SQL generation with schema-aware denoising. In: Proceedings of Findings of the Association for Computational Linguistics: NAACL 2022. 2022, 1845–1853
106. Qin B, Wang L, Hui B, Li B, Wei X, Li B, Huang F, Si L, Yang M, Li Y. SUN: Exploring intrinsic uncertainties in text-to-SQL parsers. In: Proceedings of the 29th International Conference on Computational Linguistics. 2022, 5298–5308
107. Shi P, Zhang R, Bai H, Lin J. XRICL: Cross-lingual retrieval-augmented in-context learning for cross-lingual text-to-SQL semantic parsing. In: Proceedings of Findings of the Association for Computational Linguistics: EMNLP 2022. 2022, 5248–5259

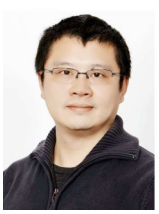
108. Pi X, Wang B, Gao Y, Guo J, Li Z, Lou J G. Towards robustness of text-to-SQL models against natural and realistic adversarial table perturbation. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2022, 2007–2022
109. Han S, Gao N, Guo X, Shan Y. RuleSQLova: Improving text-to-SQL with logic rules. In: Proceedings of 2022 International Joint Conference on Neural Networks. 2022, 1–8
110. Zheng Y, Wang H, Dong B, Wang X, Li C. HIE-SQL: History information enhanced network for context-dependent text-to-SQL semantic parsing. In: Proceedings of Findings of the Association for Computational Linguistics: ACL 2022. 2022, 2997–3007
111. Xiao D, Chai L, Zhang Q W, Yan Z, Li Z, Cao Y. CQR-SQL: Conversational question reformulation enhanced context-dependent text-to-SQL parsers. In: Proceedings of Findings of the Association for Computational Linguistics: EMNLP 2022. 2022, 2055–2068
112. Wang L, Qin B, Hui B, Li B, Yang M, Wang B, Li B, Sun J, Huang F, Si L, Li Y. Proton: Probing schema linking information from pre-trained language models for text-to-SQL parsing. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. 2022, 1889–1898
113. Awasthi A, Sathe A, Sarawagi S. Diverse parallel data synthesis for cross-database adaptation of text-to-SQL parsers. In: Proceedings of 2022 Conference on Empirical Methods in Natural Language Processing. 2022, 11548–11562
114. Zhao C, Su Y, Pauls A, Platanios E A. Bridging the generalization gap in text-to-SQL parsing with schema expansion. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2022, 5568–5578
115. Lee G, Hwang H, Bae S, Kwon Y, Shin W, Yang S, Seo M, Kim J P, Choi E. EHRSQL: A practical text-to-SQL benchmark for electronic health records. In: Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS 2022). 2022
116. Praciano F D B S, Amora P R P, Abreu I C, Pereira F L F, Machado J C. Robust cardinality: a novel approach for cardinality prediction in SQL queries. Journal of the Brazilian Computer Society, 2021, 27(1): 11
117. Yang J, Zhang Y, Wang B, Yang X. BoundEst: Estimating join cardinalities with tight upper bounds. In: Proceedings of the 7th International Joint Conference on Web and Big Data. 2023, 437–451
118. Zhou W, Zhan S, Guo L, Dai B. CELA: An accurate learned cardinality estimator with strong generalization ability and dimensional adaptability. In: Proceedings of the 22nd International Conference on Web Information Systems Engineering. 2021, 111–118
119. Liang Z, Chen X, Xia Y, Ye R, Chen H, Xie J, Zheng K. DACE: A database-agnostic cost estimator. In: Proceedings of the 40th IEEE International Conference on Data Engineering. 2024, 4925–4937
120. Huang S, Qin Y, Zhang X, Tu Y, Li Z, Cui B. Survey on performance optimization for database systems. Science China Information Sciences, 2023, 66(2): 121102
121. Liang Z, Chen X, Zhao Y, Xie J, Zeng K, Zheng K. Efficient cardinality and cost estimation with bidirectional compressor-based ensemble learning. In: Proceedings of 2023 IEEE International Conference on Data Mining. 2023, 388–397
122. Guo W, Zhuang F, Zhang X, Tong Y, Dong J. A comprehensive survey of federated transfer learning: Challenges, methods and applications. Frontiers of Computer Science, 2024, 18(6): 186356
123. Warnke B, Martens K, Winker T, Groppe S, Groppe J, Adhiyaman P, Srinivasan S, Krishnakumar S. ReJOOSp: Reinforcement learning for join order optimization in SPARQL. Big Data and Cognitive Computing, 2024, 8(7): 71
124. Ji L, Zhao R, Dang Y, Liu J, Zhang H. Query join order optimization method based on dynamic double deep Q-network. Electronics, 2023, 12(6): 1504
125. Gu R, Zhang Y, Yin L, Song L, Huang W, Yuan C, Wang Z, Zhu G, Huang Y. Coral: Federated query join order optimization based on deep reinforcement learning. World Wide Web, 2023, 26(5): 3093−3118
126. Li G, Zhou X, Cao L. AI meets database: AI4DB and DB4AI. In: Proceedings of 2021 International Conference on Management of Data. 2021, 2859–2866
127. Zhou X, Sun Z, Li G. DB-GPT: Large language model meets database. Data Science and Engineering, 2024, 9(1): 102−111
128. Chang S, Wang J, Dong M, Pan L, Zhu H, Li A H, Lan W, Zhang S, Jiang J, Lilien J, Ash S, Wang W Y, Wang Z, Castelli V, Ng P, Xiang B. Dr.Spider: A diagnostic evaluation benchmark towards text-to-SQL robustness. In: Proceedings of the 11th International Conference on Learning Representations. 2023

Shao-Jie QIAO received his BS and PhD degrees from Sichuan University, China in 2004 and 2009, respectively. From 2007 to 2008, He worked as a visiting scholar in the School of Computing at the National University of Singapore, Singapore. He is a Distinguished Young Scholars of Sichuan Province. He is currently a professor and the deputy dean with the School of Software Engineering, Chengdu University of Information Technology, China. He has authored more than 160 high quality papers, including IEEE TITS, IEEE TKDE, IEEE TNNLS, etc. His research interests include artificial intelligence for databases and spatio-temporal databases.



Han-Lin FAN received his BS degree from the Chengdu University of Information Technology, China in 2021. He is currently a master candidate with the School of Software Engineering, Chengdu University of Information Technology, China. His research interests include artificial intelligence for database.

Nan HAN received her MS and PhD degrees from Chengdu University of Traditional Chinese Medicine, China in 2009 and 2012, respectively. She is currently an associate professor with the School of Management, Chengdu University of Information Technology, China. Her research interests include artificial intelligence for databases and spatio-temporal databases.



Lan DU received his BSc and PhD degrees in computer science from the Australian National University, Australia in 2007 and 2012, respectively. He is currently an associate professor in data science and AI at Monash University, Australia. He has authored more than 100 high quality papers, including NeurIPS, ICML, ACL, AAAI, CVPR, TPAMI, and IJCV. His research interests include machine/deep learning, NLP, CV, etc.

Yu-Han PENG received his master degree from Chengdu University of Information Technology, China. His research interests include artificial intelligence for databases.

Rong-Min TANG received her BS degree from the School of Software, Tianjin Normal University, China in 2022. She is currently a master candidate with the School of Software Engineering, Chengdu University of Information Technology, China. Her research interests artificial intelligence for databases.

Xiao QIN received her MS degree in computer application technology from Nanning Normal University, China in 2009. She is currently a professor at the School of Computer and Information Engineering, Nanning Normal University, China. Her research interests include machine learning and computer vision.