Big Data Research 27 (2022) 100304

Contents lists available at ScienceDirect

Big Data Research

www.elsevier.com/locate/bdr

Survey on Learnable Databases: A Machine Learning Perspective ☆

Benyuan Zou [a], Jinguo You [a,b,*], Quankun Wang [a], Xinxian Wen [a], Lianyin Jia [a]

[a] Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Yunnan Kunming 650500, China
[b] Yunnan Key Laboratory of Artificial Intelligence, Kunming University of Science and Technology, Kunming, Yunnan 650500, China

ARTICLE INFO

Article history:
Received 11 March 2021
Received in revised form 16 November 2021
Accepted 21 December 2021
Available online 3 January 2022

Keywords:
Learnable databases
Database optimization
Machine learning
AI4DB

ABSTRACT

During the decades of development of artificial intelligence, a spectrum of applications involving image, speech, text data, etc. are successfully powered by machine learning. The advantages are mainly derived from the learning ability from data, which most traditional databases lack. Recent years have seen a surge in approaches that explore artificial intelligence to power traditional databases, i.e. learnable databases, making databases more adaptive and intelligent. Specifically, they can be automatically optimized according to historical metric statistics and current query workload, which significantly improves the database performance and relieves the trivial routine maintenance suffering. However, one of the major issues, especially for practitioners, is the lack of consensus in their definitions as well as a lack of clear categorization from a machine learning perspective. To alleviate these problems, this paper introduces concepts and algorithms related to learnable databases and investigates the progress in learnable databases in five aspects: database parameter configuration, data storage management, query optimization, query interface, and benchmark of learnable databases. Additionally, we survey AI-empowered technique development in commercial databases and new approaches to learning-based database security. We develop a categorizing framework in terms of input features, model selection, and output results (mostly being viewed as class labels). Finally, we conclude the current work and discuss future work on learnable databases.

1. Introduction

With the growth in popularity of machine learning and artificial intelligence, a spectrum of applications that involve speech recognition, image classification, and natural language processing, etc. are powered by machine learning and are witnessed their incredible success.

Machine learning learns rules and patterns from data or experience. Usually, we train a model using data that is labeled or allow it to work on its own to discover information. As a branch of machine learning, reinforcement learning [1] recently attracts much attention. Reinforcement learning is a general framework of learning, forecasting and decision-making [2]. If a problem can be transformed into a sequential decision-making problem, which defines state, action and reward, then reinforcement learning may help to automate or optimize the strategy of manual design. With abundant data and increased computing power, deep learning mimics the workings of the human brain via building complex and deep networks. They all play essential roles behind many of the applications we use every day.

The advantages of machine learning are mainly derived from the learning ability from data, which most traditional databases lack. They are static, unadaptable in most cases despite limited tries in tuning database parameters. Certain data distributions are also presumed in advance. This means that they are hard to dynamically adapt or adjust themselves according to the historical query performance, nor can they perform specific system optimization based on the diverse data distribution and user workload. However, in the real-life world, the query workload changes over time, particularly on cloud and the data distribution varies in different application scenarios. This means that these databases cannot adapt themselves automatically, nor can they perform specific system optimization based on the real data distribution and diverse workload posted by users [3]. The databases are required to dynamically adjust themselves to obtain the optimal running status [4].

With the much development of artificial intelligence, it is natural to think about combining databases with artificial intelligence such as query interface and query optimization or vice versa [5].

☆ This work was partially supported by Opening Project of Information System (CCFIS2020-06-03) by CCF and Yunhe Enmo (Beijing).

* Corresponding author at: Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Yunnan Kunming 650500, China.
E-mail address: jgyou@126.com (J. You).

https://doi.org/10.1016/j.bdr.2021.100304
2214-5796/

The integration of the two established technologies AI and DBMS was early proposed [6], but the focus is mainly the efficient management of data as well as knowledge to facilitate information retrieval. The example is Intelligent Database, which was introduced in 1989 by the book "Intelligent Databases" [7]. Intelligent database (IDB) systems integrate resources from both relational database management systems and knowledge-based systems to manage information, making it easy to store, access and apply [6]. Different from the aims of Intelligent Database, learning database can effectively capture various characteristics of workload and data by machine learning, and select the optimum machine learning model to optimize the database system. This paper emphasizes the learning ability from data and investigates learnable models and methods applied in databases or AI4DB for short.

The possibility of integrating machine learning with databases was discussed in 2015 [8]. It characterizes the role of machine learning in the effective acquisition of the workload and data. This feature makes it increasingly become popular in database optimization [9]. Learned indexes [10] pioneered the practical AI4DB work and demonstrated effectiveness and efficiency. Gartner proposed ten trends in data development in February 2019, among which "Augmented Analytics" was listed as the top three [11]. Enhanced data analysis automates the process of data preparation, insight acquisition, and insight visualization through the hybrid of machine learning and artificial intelligence. Previous literature mainly sheds light on the path of AI4DB. Nowadays, more and more work on learnable databases has been proposed with much progress in the database community of academia and industry. They aim at self-configuring, self-optimizing, self-monitoring, self-healing via machine learning.

[12] studies the limitations of the three key components in the cost-based optimizer: cardinality estimation, cost model, and plan enumeration and further summarize the learning-based and non-learning improvement techniques. [13,14] reviews the techniques on how database and artificial intelligence benefit from each other, such as AI4DB: learning-based configuration tuning, query optimizer, index/view advisor, and security, etc.

The related surveys to ours do not sort out the work of learnable databases in terms of the input features, models, and output results (mostly in form of labels) from a machine learning perspective for practitioners. Moreover, they do not discuss the development of commercial databases empowered by AI and some benchmarks in learnable databases. This paper mainly summarizes a variety of machine learning methods used in the learnable databases and describes the different characteristics of the input and output in various models.

Fig. 1 shows the framework of this survey. It mainly divides into six parts: parameter configuration, storage management, query optimization, query interface, benchmark and database security. Specifically, parameter configuration includes workload analysis and tuning plan, storage management includes data partition and index structure, query optimization includes query plan, query size and query cost, benchmark includes data sets and related algorithms, and database security includes data security and operation security.

The contributions of this paper are as follows.

- We introduce the definition and classification and investigate the progress of learnable databases including parameter configuration, storage management, query optimization, and query interface. Specifically, they are workload analysis, tuning plan, data partition, index structure, materialized view, query plan, query size, and query cost. In particular, we develop a categorizing framework in terms of input features, model selection, and output results (mostly being viewed as class labels) from a machine learning perspective. Finally, we introduce new features of AI4DB in commercial databases (e.g., OpenGauss, Oracle Autonomous Database, Azure SQL Database, and Alibaba).
- We summarize the current main benchmark datasets and methods.
- We classify and introduce the new approaches to database security.
- The research directions of future work are concluded in this paper.

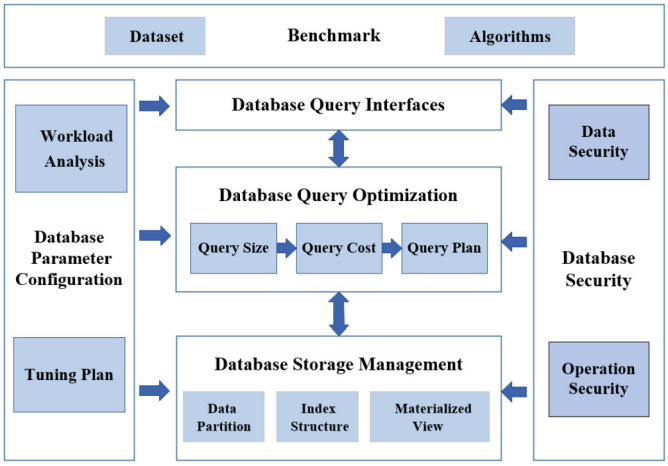


Fig. 1. Main Framework.

The remainder of this paper is structured as follows. The progress on how to automatically configure database metric parameters are introduced in Section 2. The intelligent storage management including data partition, indexing, and materialized view is described in Section 3. Section 4 investigates query optimization of the learnable database. The query interface, especially NL2SQL, is described in Section 5. The introduction of new features of AI4DB in commercial databases is described in Section 6. The various benchmark tests are described in Section 7. The introduction of database security is described in Section 8. Finally, we present the conclusion of this paper and future work in the last section.

2. Database parameter configuration

The workload of the database is changing more and more rapidly, and the database system requires to use intelligent approaches to improve the ability to respond quickly. Machine learning can be used to learn historical data to predict the unknown data so that the database system can dynamically configure the parameters under different workloads to form an optimization plan.

2.1. Workload analysis

During the decades of the development of database self-tuning, some system tuning methods have been proposed.

The research on the parameter configuration of the database can be traced back to the modification of the database system design [15] in 2005. Management tasks have gradually become the dominant task in the database system. It is inefficient to adjust CPU, memory and storage resources for the current workload only by administrator. To address this issue, Narayanan et al. proposed a way to upgrade the configuration by predicting the workload. The authors propose a modification plan for the database system design so that resource consultants can answer "what-if" questions about resource upgrades. The machine learning models of this self-prediction DBMS system are buffer pool and storage models, the input feature is workload, and the output feature is buffer pool size. This model demonstrates the superiority of performance by

predicting the throughput and response time of OLTP workloads. The advantage is that it uses a modular architecture to increase the extensibility, but it still needs to reduce response time.

For many OLTP applications, it is generally used dynamic allocation to calculate how to adjust resource allocation to adapt to the workloads that are constantly evolving. Despite this method responds to workload changes, it will reconfigure and add extra burden to the system that has been overloaded. An OLTP database, called P-Store [16], can predict the workload and make the database readjust the resource configuration before the peak workload. It re-configures the database through a time series model based on dynamic programming algorithms, which can accurately predict the load of different applications, thereby reducing the resource configuration overhead of the database. Optimizing the configuration is also critical to operating modern cloud systems, but the difficulties come from the different workloads of the cloud system, the scale of large systems, and the huge parameter space.

In 2017, [17] proposed an autonomous driving database management system and introduced the architecture of the 1st-generation self-driven database management system, Peloton. This system controls and adjusts the database configuration by predicting future workload changes through an integrated planning component. Therefore, the database management system does not need to manually determine the correct deployment mode and the appropriate deployment time. The machine learning model of this self-driven database management system is deep learning model, and the input and output features are workload and response time individually. It extends an innovative method for realizing automation of different databases and reducing manual operations, but its performance will decrease with the increase of dimensions.

Li et al. [18] studied the robustness of auto-tuning in cloud systems and proposed an auto-tuning service, Metis. It is found that performance metrics such as tail latencies can be sensitive to nontrivial noises. Besides, while treating target systems as a black box promotes applicability, it complicates the goal of balancing exploitation and exploration. Metis implements customized Bayesian optimization to robustly improve auto-tuning: Diagnostic models to identify potential data outliers for re-sampling, a mixture of acquisition functions to balance exploitation, exploration and re-sampling. Metis supports different system parameter types and predicts the best performance configuration by training the workload and parameter values of the system. It obviates service interruptions caused by a configuration change. The disadvantage is that the configuration of prediction has randomness.

Different query execution sequences result in multiple disk access. SmartQueue [19] is a query scheduling system based on reinforcement learning, which maximizes cache hits to improve query performance. The author encodes the cache state and the read operation to be performed and uses them as the input of the model. The model uses deep Q-learning to obtain a scheduling strategy to improve the cache hits. This system can maintain long-term stable performance and can also adapt to new input data access patterns, but the linear combination of neural networks is limitation.

At present, the problem of high memory consumption in common occurred, even cloud database still needed to adjust the load buffer to adapt to this resource bottleneck problem. In a large cloud cluster database, the workload of each instance may change dynamically. It will be a huge task that the buffer of each database is configured and optimized manually. iBTune [20] is the system that realizes a function of automatically arranging buffers for all instances of the entire database. When the workload on each instance may change dynamically, it is found that manual optimization is not suitable for large cloud clusters. The relationship between the miss ratio and allocated memory size is used to optimize the target buffer pool size. Meanwhile, a pair of deep neural network is designed to provide a guaranteed service level protocol (SLA), the input feature is the information on workload, and the upper bound of request response time is predicted by the measurement characteristics of the instance. The target buffer pool size can only be adjusted if the predicted maximum response time is within the security limit. iBTune saves more memory resources than other systems in practical operating, but DBA is still required to verify the extension requirements of the system to ensure that the buffer pool size satisfies the effective requirements.

2.2. Tuning plan

The tuning plan in the database generally depends on the amount of cached data and the buffer pool size of the disk pages. Through a good tuning plan, the system can avoid using the cache that has an impact on performance.

Different from the previous buffer adjustment methods, PostgreSQL-TtDB [21] uses an equation to identify the buffer size limit. By associating the buffer allocation plan with the unused rate that affects the performance cache, the probability of a buffer miss is obtained. Besides, PostgreSQL-TtDB also predicts the cost of I/O. [22] proposed STMM for memory optimization of multiple versions of DB2. This is a kind of cost-benefit analysis to adjust the allocated memory and input cost-effectiveness data to get the best tuning plan. STMM can select the optimal memory adjustment scheme based on the cost-effectiveness of different memory users.

[23] provided an automated method to overcome the problem of non-standard, independent, and non-universal DBMS configuration “tuning knobs”. The combination of supervised or unsupervised machine learning methods was used to adjust the configuration; This new tool, Ottertune, extracts training data from the historical tuning plan and uses the Gaussian model to get the recommendation of parameter configuration. It is universality and matches well in various types of databases. This tool creates more load-compliant configurations in a shorter time.

It is difficult for the database management system in the cloud environment to adapt to the changes in hardware configuration and workload. CDBtune [24] uses deep reinforcement learning with a reward feedback mechanism. This cloud database automatic tuning system replaces the traditional regression mechanism, realizes end-to-end learning, accelerates the convergence speed of the model, and improves the efficiency of online adjustment. Although the recommended configuration has limited performance when the amount of data less, especially in the cloud environment, it reduces the dependence on training samples and enhances the adaptability of the system by the reinforcement learning. After it sets the knob for the cloud database, the system can have higher throughput. The input features of CDBtune and Ottertune are historical parameter configuration data. However, the reinforcement learning model of CDBtune is more effective.

Due to the impact of the rapid growth of data scale and the rapid increase of data volume in the era of big data, and it is urgent for OLAP database to provide real-time and efficient analysis services for complex temporary data processing. Zhan et al. [25] proposed a real-time OLAP database system, AnalyticDB. This kind of database is mainly aimed at the data processing process of large-scale data with large write throughput and high query concurrency. It uses the storage structure of mixed structure row and column layout, and separates the process of reading and writing access path. It has better performance in retrieving structured and complex data types. Analyticdb also adds a SQL optimizer that can save query and execution records to reduce query latency.

The comparison of learnable database parameter configuration models is divided into workload analysis and tuning plan.

Table 1
Intelligent Parameter Configuration Approaches.

Category	Approach	Input Feature	Machine Learning Model	Output Result
Workload Analysis	Peloton [17]	Workload	Deep Learning	Deployment Action
	Metis [18]	Workload	Bayesian Optimization	Performance Configuration
	SmartQueue [19]	Code Vectors	Deep Q-Learning	Scheduling Strategy
	iBTune [20]	Workload	Deep Neural Network	Response Time
Tuning Plan	OtterTune [23]	Tuning Plan	Gaussian	Configuration Parameters
	CDBtune [24]	Tuning Plan	Reinforcement Learning	Optimal Configuration

Table 2
Advantages and Disadvantages of Intelligent Parameter Configuration Approaches.

Approach	Advantages	Disadvantages
Peloton [17]	Extending an innovative method for automation	The performance decreases with increasing dimensions
Metis [18]	Supporting different system parameter types	The configuration of prediction has randomness
SmartQueue [19]	Long-term stable performance	The linear combination of neural networks is limitation
iBTune [20]	Saveing more memory resources	Requirements of manual inspection system
OtterTune [23]	Universality & matching various databases	The performance of RL model is relative low
CDBtune [24]	High throughput & RL model is more effective	Unstable configuration performance

In terms of the workload analysis, the four machine learning models we introduced are the Bayesian optimization, the deep Q-learning and the deep neural network. Their input features are workload and code vectors, and the output results of each model are deployment actions, performance configuration, scheduling strategy and response time.

Furthermore, with regard to the tuning plan, the two machine learning models we introduced are the Gaussian and the reinforcement learning. Their input features both are tuning plan, and the output results of each model are configuration parameters and optimal configuration respectively. Table 1 gives the comparison of parameter configuration models, and Table 2 gives the advantages and disadvantages of parameter configuration approaches.

2.3. Summary

The database has dozens or even hundreds of adjustable parameters, and many of them are continuous-valued tuning spaces, and the optimal parameter combination cannot be generated depending on manual experience. Intelligent data parameter configuration can predict the workload and obtain the optimal optimization plan, so that the performance of database storage management and query optimization can be improved, and the adaptability of the database to the era of big data can be further improved.

3. Database storage management

learnable database storage management is mainly divided into data partition and index. The query load and data layout of the database determines the characteristics of the partition. Using the learning prediction characteristics of the intelligent can realize the partition optimization to improve the overall performance of the database. The index is a decentralized storage structure to speed up the retrieval. The intelligent is used to replace the index or select an efficient index to improve storage management performance.

3.1. Data partition

After data partitioning, the table can be further divided into finer granularity to facilitate distributed data processing. [26] proposed OpenAI Gym that realized a self-adjusting function of data partition and layout. Through the reinforcement learning model, the input feature is the prediction of external workload and current physical design, and the output result is the selection of an adaptive partition. In order to use in the online environment, the agent learns n action sequences of fixed length, which maximizes the time reward of the predicted workload. Its advantage is that it can greatly reduce parallelism in training, GPU utilization, and memory footprint reductions, but the overall system has not been developed.

The DEL-based [27] that uses a DRL (deep reinforcement learning model) to solve the partition problem of a distributed database and provide a ready-made extended database solution for OLAP style workload in the cloud. DRL agents learn from experience by monitoring the returns of different workloads and partitioning schemes. Set up an online learning stage to continue learning the actual execution time of the data partition after the estimated execution time of the learning load. Through evaluation, this method not only finds better partitions than the existing automatic partition design method but also easily adapts to different deployments, it is not currently supported in the OLTP database.

3.2. Index structure

The index structure provides a query path to make data access more efficient. It can also satisfy a variety of data access modes with different requirements. The index structure can also be replaced by a machine learning model. [10] proposed an idea: the machine learning model could be used to learn the sorting order of search keys to predicting the search position. This type of learning index can predict the corresponding record location of the query when the query key is a type of integer or string. Replacing the index structure with machine learning models can save a lot of space compared with other traditional indexing techniques. Undoubtedly, the learning index is a new direction in database indexing. Although this kind of index structure has been improved in space utilization, it can not completely replace the traditional indexes in practical application.

[28] applied a new multi-dimensional index (Flood) to five core spatial partitioning techniques, namely fixed-grid, adaptive-grid, Kd tree, QuadTree and STR tree. It indexes points using a variant of the Grid-file. The current core spatial partitioning will process the query through three stages: index searching, optimization and scanning. The authors replace the learning model to the index optimization stage, that can effectively improve the query speed, especially for low-selectivity range queries.

To deal with the problem that the proposed new memory index cannot be easily integrated with the key-value system, Zhang et al. [29] proposed S3, which is an in-memory skip-list index used in expandable memory and adopts a two-layer index structure. At the top level, it used a cache-sensitive structure to maintain

some protection entries to facilitate searching in the skip list. At the bottom, a semi-sorted skip list index is constructed to support highly concurrent insertions and fast search and range queries. In order to further improve the indexing performance, it trains a neural network to intelligently select protection entries according to data distribution and query distribution. The approximation strategy trained by the neural network can deal with more complex situations. The disadvantage is that the index will sacrifice the reading performance. This skip list index uses a neural network by input key values for training. It will obtain the indexes making the better optimized query. Experiments on multiple datasets show that S3 has the same query capability as other new memory indexing schemes, and can support large amounts of data by replacing skipping lists in current memory systems such as LevelDB [30] and RocksDB [31].

The NoDBA system [32] is an automatic database management system based on deep reinforcement learning. NoDBA trains the code of the current workload and current configuration to obtain query response time using a neural network. Generally, due to space constraints and index maintenance costs, there is a ceiling on the number of indexes to be created. Therefore, it is not an option to simply index all columns, and it will create an index on a specific column that is defined as an executable operation. The system uses a deep reinforcement learning model to iterate the initial index and load state. When the number of indexes reaches the upper limit, the iteration stops, and the optimal index configuration is given. In the case of a given workload, the task of index selection is to determine the properties to create a secondary index that maximizes the benefit of workload processing. NoDBA can be optimized without query estimation, but the disadvantage is that it cannot be implemented well under non-fixed workloads.

Many kinds of machine learning systems have been widely used in database, but it is becoming increasingly difficult for users to understand the results of these machine learning models. Therefore, it is a new research direction to help users better understand these models. Users can only choose by comparing the accuracy of the model, so they cannot understand how to select the model in a specific situation. Kahng et al. [33] used data cube analysis to explore and understand the results of machine learning, and proposed MLCube, which uses a visual tool to help users observe which models could better improve the performance of data processing in practical applications. At present, such visualization tools are only applied to small-scale instances, and more work is needed to determine whether they can be extended to larger data sets or more complex systems.

3.3. Materialized view

A materialized view is a database object that includes a query result. It is used to generate a summary table based on the sum of data tables. Materialized views are essentially queries, and they are conjunctive queries that form an algebraic lattice by calculating dependency relationships (that is, scroll up, drill down, or containment).

One of the main advantages of materialized views is to precompute the query and save the results. Decision analysis usually involves complex queries with high computational costs (for example, join and aggregate calculations for tables with a size of one million or more tuples). Pre-calculating some queries and answering from these queries can effectively improve query efficiency. In query optimization, finding query rewriting in materialized views can generate a more effective query execution plan.

Reinforcement learning plays a vital role in intelligent materialized views. It is model as a Markov Decision Process (MDP), mainly including state, action, policy and reward. As shown in Fig. 2, State: M; Workload: Q; Action: $\{0, +\}$ create views or not;

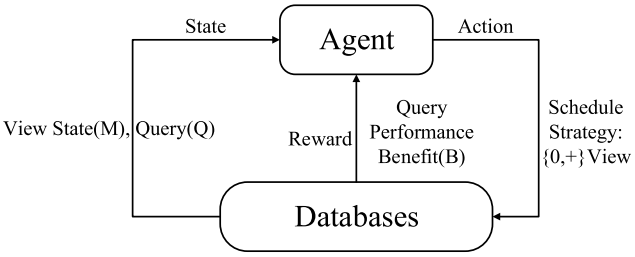


Fig. 2. Automatic View Materialization with Reinforcement Learning.

Policy: (Q, M) -> $\{0, +\}$ according to the current state, the corresponding actions are selected to generate the decision sequence; Reward: $R\ (q,v)$ the benefit B from creating views, B= $Query\ (q,v)$ − $Query\ (q,\phi)$; The agent filters the candidate views according to the input view state (M) and query workload (Q), then according the policy and benefit B, it selects the views which improves the query performance. Specifically, the reward function R is calculated by evaluating the actual time benefit B economized by querying Q under the created view compared to not creating the view.

When a quantity of queries is posed on big data, some query results have many possibilities to recur multiple times, which can be materialized to reduce these redundant calculations. Including the problems of defining the materialized view as an integer linear programming, the approaches for estimating the query cost after rewriting the materialized view by using machine learning and deep learning, and the methods to iteratively optimize the selection of the materialized view by using reinforcement learning.

3.3.1. Automatic view generation

[34] proposed the end-to-end automatic view generation method, which selects “highly beneficial” sub-queries to materialize. To achieve this method, the authors summarize the tasks that are supposed to be addressed. They include: how to estimate the benefit of using a materialized view for a query; defining the materialized view as an integer linear programming (ILP) problem; using machine learning and deep learning to estimate the query cost after the materialized view is rewritten and using reinforcement learning to iteratively optimize the selection of the materialized view.

It contains three phases: 1) Pre-process: the system extracts candidate sub-queries from the query workload. 2) Offline training: first, the system collects the training data from the query engine as input features; then it trains the cost estimation model using the actual cost of the query rewritten by the appropriate views and fine-tunes the model by the actual benefits. 3) Online recommendation: the benefits derived from the cost estimation model are used to recommend the best sub-queries to generate materialized views.

When the current queries match the results of views, the database can rewrite these views which avoids the process of redundant queries.

The features which are the input of the cost estimation model contain two parts: the query/view plans and the associated tables. The collected table statistics are converted into numerical features and the query/view plans and associated tables are split into non-numerical features. The former is normalized and input into the wide linear model to get a fixed-length vector, while the latter is input into the schema encoding model and plan sequence coding model in the depth model to convert into fixed-length vectors. Finally, a regressor is used to combine the outputs of the wide and deep models to obtain the final predicted cost.

To prove the improvement of query performance achieved by this system and how to select optimal subqueries. The author first enters the query results through a neural network and extracts fea-

Table 3
Intelligent Storage Management Approaches.

Category	Approach	Input Feature	Machine Learning Model	Output Result
Data Partition	OpenAI Gym [26]	Query Workload	Reinforcement Learning	Way of Block
	DEL-Based [27]	DRL Agent	Deep Reinforcement Learning	Query Combination
Index Structure	Learned Index [10]	Plastic/String Dataset	Neural Network	Filtered Query Records
	S3 [29]	Key Values	Neural Network	Optimization Index
	NoDBA [32]	Workload Coding	Neural Network	Query Response Time
	MLCube [33]	Feature Conditions	Deep Learning	Model Performance
Materialized View	Automatic View Generation [34]	View Plans and Associated Tables	Deep Reinforcement Learning	Predicted Query Cost
	Materialized Views Selected [36]	Query Time Benefits	Asynchronous Reinforcement Learning	Valuable Views

Table 4
Advantages and Disadvantages of Intelligent Storage Management Approaches.

Approach	Advantages	Disadvantages
OpenAI Gym [26]	Greatly reduce parallelism	The overall system has not been developed
DEL-Based [27]	Adapting to different deployments	Unsupported in the OLTP database
Learned Index [10]	Increasing space utilization	Can not completely replace the traditional indexes
S3 [29]	The approximation strategy can deal with complex situations	The reading performance is relative low
NoDBA [32]	Supporting optimize without query estimation	Cannot be implemented well under non-fixed workloads
MLCube [33]	Helping users observe models better	Only applied to small-scale instances

tures from different angles, then uses an effective coding model to convert the extracted features into hidden representations; the author introduces integer linear programming to solve the problem of how to choose the optimal subquery, and iterates. The optimization process is replaced with a deep reinforcement learning model, and the performance of the obtained query plan is improved to a certain extent.

[35] further proposed an end-to-end autonomous MV management system AutoView via deep reinforcement learning, which integrates the embedding vectors of materialized views and queries to capture the correlation between them.

3.3.2. *Materialized views selection*

Aiming at the problem of using materialized views to improve the performance of OLAP workloads, Liang et al. [36] proposed a DQM system, which uses a deep reinforcement learning model to obtain an adaptive materialized view solution. RL is equivalent to a "learning by doing" process, which can observe the actual query behavior through performance indicators. The more feedback you get, the more efficient the learned behavior. In this way, the RL method can identify whether the view that can be stored currently is valuable. The OLAP system can save query results that may be useful in the future through materialized views. However, in actual applications, the system will be applied to various data and its usage mode will also change. From this, the previously stored view results may not be used in future queries and thus be discarded. Besides, maintaining a large number of views puts a burden on the query optimizer and cannot select which views to use for a given query. The experimental results of DQM under multiple load conditions of the SparkSQL dataset show the ability to adapt to the workload.

The authors quoted Opportunity Materialization (OM) [37], which makes it possible to pre-cache the results of important queries (or sub-queries) for future use. In the DQM system, the storage of the optimal view is achieved through three steps. First, an online miner is used to obtain the trace of the SQL query to determine the candidate view of the current query. Second, the RL agents are then selected from the candidate set and filtered. Third, some views used to rewrite queries will not match future query plans. In the current query, the query sequence of the database includes relational tables and materialized views, and a process of rewriting the view exists. Therefore, the additional query time can be divided into two parts: the time to create the view and the incremental cost of answering the query with the view.

The authors formalize the online view selection into a Markov decision process. First, the DQM system selects the view through an asynchronous reinforcement learning algorithm, and inputs the actual time benefits before and after each query uses the view into the decision model and quantifies the rewards of each time-step (query) to analyze the instantaneous benefits or harm in the current state. In the meantime, these queries and materialized views are filtered through certain conditions to output valuable views. Finally, these views are input into the eviction model with storage constraints. If the current storage space reaches the specified threshold, the view management will evict the views with fewer benefit gains and replace them with better views.

The intelligent storage management models are surveyed in terms of data partition, index structure and materialized view.

In data partition, the main machine learning models are reinforcement learning and deep reinforcement learning. Their input features are query workload and DRL agent and the output results of each model are the way of block and query combination respectively.

Besides, with regard to the index structure, the machine learning models are the neural network and the deep learning. Their input features are plastic, string datasets, key values, workload encoding, and feature conditions. The output results of each model are filter query records, optimization index, query response time and model performance respectively.

Finally, with respect to the materialized view, the machine learning models are deep reinforcement learning and asynchronous reinforcement learning. The input features of the first approach are view plans and associated tables; the input feature of the second approach is query time benefits, and the output results of each model are predicted query cost and valuable views. Table 3 gives the comparison of storage management models, and Table 4 gives the advantages and disadvantages of storage management approaches.

3.4. Summary

Adaptive data partition now plays an important role in storage management optimization. Intelligent methods can learn workload and data distribution to predict the adaptive partition results. The learning indexing also opens up a whole new research direction for the research field of storage management optimization. For new memory indexes that cannot be easily integrated with key-value systems, new indexing methods have also been proposed. For users to better understand machine learning models, some visual tools are proposed. In this section, we introduce two intelligent materialized view methods that efficiently improve the query performance of the database. There are many tables in the database, and there are many columns in the table. How to automatically construct indexes and views to improve the performance of the database is a problem to be considered.

4. Database query optimization

The query operation can convert the user's query and data modification operation commands into the operation sequence of the database system. The current query optimization method is transforming the query optimization module with artificial intelligence. Since the reasonable execution plan is very important for query processing, appropriate intelligent technologies can be combined at various levels of query analysis, processing, and optimization to improve the performance of the execution plan [38]. Query optimization has straight been a problem that databases are supposed to improve database performance, and it includes query size estimation, query cost estimation, join operation sequence optimization, query plan generation, and so on.

4.1. Query plan

The final execution plan of the database is based on the estimation of the query workload and the query size. Therefore, the intelligent methods can be used to obtain the optimal query plan by learning the historical queries.

At present, the main query optimization method is using machine learning, which can continuously optimize the quality of query size estimation and query cost estimation according to the performance of each query estimation and actual performance. The purpose of query optimization is to enable the calculation of relational expressions to select the most effective and least expensive query plan.

Query Performance Prediction (QPP) [39] is the core of effective resource management, query optimization and query scheduling. The current query optimizer mainly uses the analysis cost model to select the smaller written plan among all the alternative query plans. However, it has a weak ability to predict the execution delay of the query plan. In 2012, [40] proposed a query optimization model based on predictive modeling, which can learn query execution behaviors of different granularities. It used QPP to obtain efficient query plans for different types of data models, and mix the high-efficiency query plans of coarse-grained planning model and fine-grained type of complex operation model. In short, it is used this model to predict efficient query plan under static and dynamic query workloads.

It is feasible to introduce the prediction of query execution time into database management tasks. Wu et al. [41] considered the more general problem of dynamic concurrent workloads. The framework proposed in this paper is based on analytic modeling. What is worth paying attention to is that they used a combined queue model and buffer pool model based on a query optimizer. First, it uses a cost model to obtain the query requirements of I/O and CPU of each query channel. Second, a combination of the queue model and buffer pool model is used to merge the query requests. Finally, the combination model can evaluation the query time prediction of all query tasks. Through relevant experiments, it is proved that the method based on analytical modeling has stronger prediction accuracy than the method based on traditional machine learning. The general applicability and validity of these two methods are demonstrated by efficient query tasks on PostgreSQL and TPC-H datasets. However, there is uncertainty, and it is necessary to confirmed whether the performance of the database can be further improved under the combination of the two methods.

In 2018, [42] proposed a new approach for learning to optimize connection search strategies through the connection between the classic dynamic programming enumeration method and the latest research results of reinforcement learning. The RL-based DQ optimizer obtains prediction results of customized search strategies and search times, and its input features are planning spaces and specific data sets. The RL-based DQ optimizer uses three versions of DQ which were created in Apache Calcite, integrated into PostgreSQL and SParkSQL to illustrate the ease of integration into existing database management systems. The plan implemented by DQ in each system has optimization costs and query execution time that compete with the native query optimizer, but execution speeds up significantly after learning (usually by orders of magnitude), the deficiency lies in the high requirement for the universality of data.

4.2. Query size

The best query plan is based on the evaluation of query size and query cost of all plans. Query size estimation is one of the most important factors of query cost estimation. By improving the accuracy of query size estimation to optimize query cost estimation, the query plan is also optimized [43].

Lakshmi et al. [44] first used neural networks for query size estimation in 1998. It is an iterative method used to parse the metadata which is used as the feature vector by neural network, and the output result of this method is an accurate selective estimation of predicates. This method makes use of real query sizes to achieve query size estimates for user-defined data types and user-defined functions. Because the distribution of the data is not necessarily uniform, ordinary activation functions in neural networks cannot accurately fit the changes in query size. This method provides ORDBMS and extensible DBMS vendors with a practical solution that can be extended using data cartridges or extenders, but the overall optimization requires additional integrated optimizers.

MSCN [45] is a multi-set convolution network designed for representing relational query plans. It uses set semantics to capture query features and true cardinality. The input feature vector of the MSCN model is the range query threshold of numerical data in SQL statements, and it uses a neural network model integrating approximate selectivity functions [46]. MSCN is built based on sampling-based estimation, and solves the weaknesses of unsampled tuples that are not conforming to the predicate, and it captures the cross-correlation of joins.

In addition to using the threshold of the query range as the feature vector, the Deep Sketch [47] also uses the statistical histogram information of each table in the database. The Deep sketch is a cardinality estimation method based on deep learning, which can obtain the correlation between different columns(and even across tables). It uses a multi-set convolution neural network that outputs the correlation results between the tables to perform an estimate of query size. Its effect is better than the traditional cardinality estimation, but the structural information which the feature vector obtained is insufficient.

4.3. Query cost

Query cost estimation is the premise of query plan generation. By selecting an appropriate query size and a query cost as reasonable as possible, the query execution plan is finally generated.

In the application of the large data warehouse, the query behavior can be determined before query execution, and it will solve the practical problems. For instance, if the database management system is able to identify the query that will run for a long time, it can avoid the resource occupation problem caused by other queries; for some query tasks, if the system can determine whether it provides the required workload requirements within the task period time, it can reject or preplanned allocation these query tasks. Ganapathi et al. [48] developed a system in 2009 that uses the KCCA (Kernel Canonical Correlation Analysis) to predict whether the database can meet the performance index of the query. KCCA uses a neural network to train the number of connection operations and the size of the query in the query load and obtains various performance indicators of the database query, such as query execution time and the number of access tuples. This method can not only accurately predict queries with different timeliness, but also compare the accuracy of using different techniques to predict query indexes. This method is still tested its performance in Map-reduce, and it is supposed to complete the operational mode in the parallel computing environment.

Aiming at the query cost prediction problem of the query optimizer, Bi et al. [49] used LSTM to predict the query cost. This method uses the operation behavior and actual running time in the query plan as the source of neural network feature extraction. To reduce the complexity of load management, a sophisticated model based on a recurrent neural network is proposed to predict the query overhead, the operation behavior, and actual running time in the query plan are used as the source of feature extraction. When a specific query plan is given, the model can generate the predicted execution time interval before the plan is executed. This will be more referenced than the cost estimation results produced by the query optimizer of the existing database, and it will be preferred than the query progress indicator that predicts after execution starts. This model outputs the predicted cost sequence of the query plan, and before the query plan is executed, it can generate a prediction of the actual running time of the plan. The drawback is that this model has not been extended to other types of databases other than PostgreSQL database.

In order to estimate the selectivity of different queries on a single table, [50] proposed a data-driven cardinality estimation method. First, the authors use the deep autoregressive model (DAR) to learn the data features, that is, learn the joint probability distribution of the data column. The input feature is a set of n-dimensional tuples, and the output features are the point density estimates which are used to obtain the detection indicators through a selective estimator. Second, they realize a way that supports arbitrary range queries, that is, they select sample rows in the query range of the first column based on the probability estimated by the DAR model, and then the DAR model iteratively derives the selectivity of the remaining range queries on these rows. Furthermore, to generate a truly usable estimator, the authors develop a Monte Carlo integration scheme based on the autoregressive model, which can effectively deal with range queries of dozens of dimensions or more.

The previous learning-based cost estimation methods have some shortcomings: they only focus on the estimation of SQL statements, which is not suitable for the estimation of different query sub-plans in query optimization. To efficiently estimate the cardinality and query cost under multi-table connection, [51] proposed an end-to-end cost estimation framework based on Tree-structured Model. First, the authors use a training data generator to collect the fully connected graph of the database; they randomly generate the connected table and connected conditions and randomly generate the corresponding type of query conditions for each column, then the query optimizer obtains the execution plan and costs through the optimizer. Second, they use a feature extractor to extract tree structure encoding of the execution plan, structure encoding of query condition, and string encoding. The nodes in each execution plan are encoded into a vector. As the value of the string is discrete and sparse, for instance, a LIKE query will query various substrings which enumerate too many queries to be learned. The authors reduce this problem to a set-covering problem so that one set of rules can be used to generate strings that are covered in the workload. Finally, the authors design a tree-structured model to capture the structural and semantic features of the encoding execution plan, then it estimates the cardinality and cost on this basis. The model uses a two-layer fully connected neural network to transform query representation into cost and cardinality, and it gets loss compared with real cost and cardinality, then returns it to an upstream neural network for parameter updating. The advantage of this model is that it uses some shared operators to obviate some redundant calculations, but the cost of operators is high.

The comparison of intelligent query optimization models is classified into query plan, query size and query cost.

In terms of the query plan, the machine learning model we introduced is reinforcement learning. The input features are specific datasets, and the output result is the search strategy.

Besides, with respect to the query size, the machine learning models we introduced are neural network and multi-set convolution network. Their input features are the parsed metadata, the range query threshold of numerical data and tables, and the output results of each model are the accurate selective estimation of predicates, filter query records and table correlation respectively.

Finally, with regard to the query cost, the machine learning models we introduced are neural network, LSTM neural network, deep autoregressive and tree-structured model, their input features are workload code, query plan, a set of n-dimensional tuples, and structural and semantic features of the encoding execution plan, and the output features are query response time, query cost, point density estimates and cost and cardinality of the plan. Table 5 shows the comparison of query optimization models, and Table 6 shows the advantages and disadvantages of query optimization approaches.

4.4. Summary

Intelligent query optimization mainly includes query plan, query size and query cost. The query plan depends on the results of query size and query cost estimation; the query accuracy and the estimation accuracy of the query cost are continuously improved to formulate a better execution plan. Databases systems are prone to dynamic changes. Intelligent query processing and optimization can monitor the hardware state of the database, detect performance fluctuations, and use transfer learning to improve the adaptive ability of the model. In addition, end-to-end learning query optimizer may provide more insights into AI-enabled query processing and optimization problems.

5. Database query interface

In database optimization, query interface (query language) is recently paid much attention to.

5.1. Query interface

[52] aimed at the tedious problem of information extraction caused by the complex attributes of data and researched the ex-

Table 5
Intelligent Query Optimization Approaches.

Category	Approach	Input Feature	Machine Learning Model	Output Result
Query Plan	RLDQ [42]	Specific Datasets	Reinforcement Learning	Search Strategy
Query Size	Iterative [44]	Parsed Metadata	Neural Network	Selective Estimation
	MSCN [45]	Query Threshold	Neural Network	Filtered Query Records
	Deep sketch [47]	Tables	Multi-Set Convolutional Network	Table Correlation
Query Cost	KCCA [48]	Query Load Parameter	Neural Network	Query Response Time
	LSTM [49]	Query Plan	LSTM Neural Network	Query Cost
	Data-Driven [50]	Tuples	DAR	Point Density Estimate
	End-to-End [51]	Structural and Semantic Features	Tree-structured	Cost and Cardinality of Plans

Table 6
Advantages and Disadvantages of Intelligent Query Optimization Approaches.

Approach	Advantages	Disadvantages
RLDQ [42]	Execution speeds up significantly	High requirement for the universality of data
Iterative [44]	Providing practical solution	Overall optimization requires additional integrated optimizers
MSCN [45]	Captured the cross-correlation of joins	The input features are limited
Deep-sketch [47]	Obtaining the correlation of various columns	The structural information obtained is insufficient
KCCA [48]	Compare the accuracy of predicted indexes	Still test in Map-reduce and parallel computing environment
LSTM [49]	Generating the predicted execution time interval before the plan is executed	Limited to PostgreSQL database
Data-Driven [50]	Generating truly usable estimator	Limitations on the number of dimensions
End-to-End [51]	Obviating some redundant calculations	The cost of operators is high

ploratory query that can skip query language and avoid mechanism complexity. When the data set is more complex or difficult to be parsed by the user, the attributes and certain characteristics of the data can be used to infer the query results required by the user. The authors introduce this example-based exploratory analysis method, which may be a new research direction in the query interface.

A method that represents SQL queries as vectors named Query2Vec [53] which converts natural language to database query language. Its input features are the query plan and query template, and it trains the LSTM neural network to obtain a recommended query vector which is used to represent the semantics of the query. Query2Vec applies this recommended query vector to index recommendations to indicate that the SQL query has achieved good results. The main advantage is that the knowledge learned in the model of large SQL workload can be transferred to other workloads and applications, and it is currently being extended to query optimization and data integration.

The CODE-NN model [54] is an LSTM neural network model with an attention mechanism. This model is superior to competitive benchmarks and achieves state-of-the-art performance on automatic indicators (METEOR and BLEU) and manual evaluation studies. At present, the authors plan to develop better models for capturing the structure of the input. Its input feature is the SQL Statement, and its output feature is a description of the functions that can be implemented. CODE-NN answers programming questions by retrieving the most appropriate code snippet from the corpus and breaks previous benchmarks for this task based on MRR.

[55] proposed a cognitive database that introduces artificial intelligence into query statements and query interfaces in relational databases. First, it converts structured data into meaningful unstructured text data. Second, an unsupervised neural network model for input query is constructed by using word embedding to generate text. Finally, this model is combined with SQL query structure to generate a new SQL-based analysis query. This kind of query is called cognitive intelligence (CI) query, which can be combined with the knowledge provided by external knowledge, and it obtains the semantic relation vector used to code context in database tokens of different types. CI query uses this semantic relation vector to realize complex query, such as semantic matching and inductive reasoning query. The improvements to this cognitive database are mainly new approaches for accelerating model training and developing incremental vector training.

In terms of the query interface, the machine learning models we proposed are neural network and LSTM. The input features of the first approach are query plan and query template, and the output result is recommended query vector. The input feature of the second approach is SQL statement, and the output result is summaries of SQL codes. The input feature of the third approach is the model of vector, and the output result is predictive query. Table 7 shows the comparison of query interface models, and Table 8 gives the advantages and disadvantages of query interface approaches.

5.2. Summary

Natural language query interfaces, say NL2SQL, are becoming a new trend for database interface.

6. AI4DB in commercial databases

At present, intelligent database presents a variety of new features. This paper summarizes the new features of AI4DB in different commercial databases, which are OpenGauss Database, Oracle Autonomous Database, Azure SQL Database, and Alibaba Database respectively; They introduced functions or components optimized for their own environment.

6.1. Opengauss database

Three intelligent technologies are mainly used in the Opengauss database, they include X-Tuner (database parameter tuning framework), SQLDiag (SQL statement intelligent recognition), and Intelligent Optimizer (database execution cost prediction).

X-Tuner has the following characteristics. 1) strong robustness and certain fault tolerance. 2) Flexible deployment and convenient operation. 3) Easy to understand and facilitate secondary development. In practical application, relying on X-Tuner related technology, Huawei cloud database DAS service can intelligently recommend parameters for the historical load of user database. Through

Table 7
Intelligent Query Interface Approaches.

Category	Approach	Input Feature	Machine Learning Model	Output Result
Query Interface	Query2Vec [53]	Query Plan and Template	Neural Network	Recommended Query Vector
	CODE-NN [54]	SQL Statement	LSTM Neural Network	Summaries of SQL codes
	Cognitive Database [55]	The Model of Vector	Neural Network	Predictive Query

Table 8
Advantages and Disadvantages of Intelligent Query Interface Approaches.

Approach	Advantages	Disadvantages
Query2Vec [53]	Know ledges can be transferred to other workloads	Expanding and integrating
CODE-NN [54]	Superior to other competitive benchmarks	Developing better models for capturing the structure of the input
Cognitive Database [55]	It can combine with the knowledge provided by external knowledge	Improvements to new approaches for accelerating model training and developing incremental vector training

the actual test, the overall performance is improved by about 20%, which can greatly save cloud computing resources for users and reduce production costs.

SQLDiag focuses on the historical SQL statements of the database. Since there will not be much difference in the execution time of database SQL statements in a short time, SQLDiag can detect similar SQL statements from historical data, and predict the execution time of SQL statements based on SQL vectorization technology and timing prediction algorithm, and then identify potential slow SQL. This framework has the following advantages. 1) It does not require an execution plan for SQL statements and has no impact on database performance. 2) At present, many algorithms in the industry are highly targeted, such as only applicable to OLTP or OLAP. SQLDiag can even be used in NoSQL after modification. 3) The framework is robust and easy to understand. Users can design their prediction model only by simple modification.

The intelligent optimizer is convenient operation, has finer granularity, and is easy to locate the performance bottleneck of the plan. Besides, the deployment is flexible to minimize the impact on database performance. It also has an open interface that enables users to place a custom machine learning operation in a database function to implement simply calling.

6.2. Oracle autonomous database

Oracle autonomous database has the following new features: 1) Automatic database configuration, optimization and expansion. It can configure and adjust to specific workloads and expand computing resources when needed, all of which is done automatically. 2) Automated data protection and security. Oracle autonomous databases automatically protect sensitive and regulated data, repair database security vulnerabilities and prevent unauthorized access. 3) Automatic fault detection, fault transfer and repair. Oracle autonomous databases can automatically detect and prevent system failures and user errors.

6.3. Azure SQL database

Azure SQL database automatically optimizes and uses the recommendations of CREATE INDEX, DROP INDEX, and FORCE LAST GOOD PLAN to optimize database performance. CREATE INDEX is used to identify indexes that can improve workload performance, and automatically verify whether query performance is improved. DROP INDEX deletes unused (in the past 90 days) and duplicate indexes, but does not delete unique indexes. This option may automatically be disabled when there are queries with index prompts in the workload, or when the workload performs partition switching. FORCE LAST GOOD PLAN can automatically correct the query plan.

6.4. Alibaba database

Alibaba's Database Autonomy Service is a cloud service based on machine learning and expert experience to realize database self-perception, self-healing, self-optimization, self-operation, and self-security. This service can help users eliminate service failures caused by manual operation, and it effectively ensures the stability, security and efficiency of database services.

7. Benchmark

7.1. Datasets

Here we investigate the benchmarks most on datasets for AI4DB, specifically NL2SQL.

In terms of datasets, the current more popular English datasets include WikiSQL, WikiTableQuestions, ATIS, etc. each data set has its characteristics. A brief introduction to these datasets as follows.

7.1.1. WikiSQL

This dataset is a large-scale labeled NL2SQL dataset proposed by Salesforce in 2017, and it is the largest NL2SQL dataset at present. It is a large crowd-sourced dataset for developing natural language interfaces for relational databases. It contains 24,241 tables, 80,645 natural language questions and corresponding SQL statements. Currently prediction accuracy rate of up to 91.8% of the academic community [56].

7.1.2. Spider

The Spider dataset [57] is a larger NL2SQL dataset newly proposed by Yale University in 2018. The dataset contains 10,181 natural language questions and 5,693 SQLs distributed in 200 independent databases, covering 138 different fields. Although it is not as good as WikiSQL in terms of the amount of data, Spider introduces more SQL usage, such as higher-level operations such as Group By, Order By and Having. It even needs to join different tables, that are closer to the real scene, so it is more difficult. Currently, the highest accuracy rate is only 54.7%.

7.1.3. WikiTable questions

This data set is proposed by Stanford University in 2015 for those semi-structured question and answer forms in Wikipedia [58]. It contains 22,033 real questions and 2,108 tables. Since the source of the data is Wikipedia, the data in the table is true but not normalized. A cell may contain multiple entities or meanings, such as

"Beijing, China" or "200 km". To generalize the data in other fields properly, the relationship between the table topics and entities in the test set of this dataset is not seen in the training set.

7.1.4. The air travel information: system (ATIS)

It is a relatively old classic data set, which is proposed by Texas Instruments in 1990. The data set is taken from the relational database Official Airline Guide (OAG, 1990) and contains 27 tables and less than 2,000 queries. Each inquiry has an average of 7 rounds. In 93% of the cases, more than 3 tables need to be combined to get answers. The contents of the inquiry include flight, cost, city, and ground service information [59].

7.1.5. The Tianyi Chinese dataset

The Tianyi Chinese data set is the data set published by Tianyi Technology Limited Corporation for the Tianchi competition, including 40,000 labeled data as the training set and 10,000 unlabeled data as the test set. The accuracy of the first place in the competition has reached 92%.

7.2. Benchmark

When benchmarking the WikiSQL dataset in 2017, the logical form accuracy and execution accuracy are 23.3% and 37% in the development environment. Compared with a relatively stable test environment, the accuracy rate of the logic form is 23.4%, and the execution accuracy rate is 35.9%. The author of this data set also proposes a seq2sql model [60], which is based on a neural network. Compared with baseline, the execution accuracy of this model is increased from 35.9% to 59.4%, and the logical form of accuracy is increased from 23.4% to 48.3%. The execution accuracy of the WikiSQL dataset has increased from the original 59.4% to 91.8% in recent years, and progress has been rapid. Its method has also been transformed from simple seq2seq to multitasking, transfer learning, and pre-training.

In the same year, technicians from the University of Washington and Microsoft made some improvements to the Seq2SQL model. They used supervised learning for training. The accuracy of logical form and execution of the model is 59.5% & 65.1% respectively, which are a significant improvement compared with the previous version.

Xu et al. [61] proposed SQLNet. The test standard increases the query matching accuracy rate in addition to the logical form accuracy rate and execution accuracy rate. Its meaning is to convert the synthetic query SQL and ground truth query SQL into the standard form, and see if it can still match the previous exact match. In the test, the query matching accuracy is 61.3% and the execution accuracy is 68%, which proves that SQLNet can be 9% to 13% higher than the prior in WikiSQL tasks.

In Microsoft's latest research, they proposed X-SQL [62], a new network architecture for parsing natural language into SQL queries. The evaluation of X-SQL on the WikiSQL dataset has an accuracy rate of 86% in logic form and an execution accuracy rate of 91.8%.

8. Database security

As early as 2004, there has been the introduction of artificial intelligence to database security. Bai et al. [63] proposed a formal artificial intelligence method in object-oriented database system security, which combined the specification of object-oriented databases with security policy and provided formal syntax and semantics. In recent years, some learning-based algorithms [64,65] have been used in database security, these algorithms are proposed to discover sensitive data, conduct access control, and avoid SQL injection.

Aurum [66] is a learning-based sensitive data discovery system, it provides flexible queries to search data sets and plays the role of automatic detection, so it timely and accurately monitors confidential data. Since the data will be accessed by unauthorized users, [67] is a purpose-based access control model. It formulates relevant access control strategies to constrain the data requests of different users, which can well protect the database from random access. SQL injection is currently the most common and harmful method affecting database security. Classification tree [68] and the fuzzy neural network [69] are proposed to detect SQL injection to prevent interference with SQL statements to modify or view higher priority data in the database.

9. Conclusion and future work

This paper investigates the progress in learnable databases from the perspective of machine learning, including parameter configuration, storage management, query optimization, query interface, and benchmark test.

The current main related work is summarized as follows. 1) self-tuning system parameter configuration based on historical workload and query data via reinforcement learning. 2) automatically data partitioning and index recommending or optimizing by using certain machine learning models. 3) self-optimizing query plan by estimating the query size and query cost via deep learning or reinforcement learning. 4) providing query interface by combining natural language and database SQL language. 5) many commercial databases have also introduced AI, and they have developed different new features. 6) database security generally refers to the security of the data and system operation.

Recently, graph algorithms based on graph embedding representation have proved the great potential of AI in graph processing, e.g. complex coupling and interactive network topological structure and attribute information are modeled via deep autoencoder [70], and the storage management and performance optimization of graph databases empowered by AI will be a new research direction. Many deep learning models have been applied to space and time series modeling, which provides an opportunity for intelligent data access to extend to space-time data. Furthermore, deep learning is also applied to more popular spatio-temporal prediction problems, and spatio-temporal data query processing and optimization will also be one of the promising research directions in the future.

Although deep learning is applied in some scenarios such as query optimization and query interface etc. in databases, the lack of interpretation and the frangibility prohibit its further use in practice. To integrate both knowledge-driven and data-driven models will mitigate the challenges. Data distribution drift may make the model ineffective. Thus another promising direction is online learning or continuous intelligence [11] that uses real-time context data to improve decision making in databases.

Declaration of competing interest

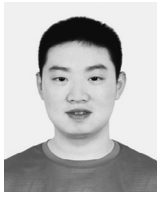
The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, The MIT Press, 2016.
[2] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, 2018.
[3] L.-M. Sun, S.-M. Zhang, T. Ji, C.-P. Li, H. Chen, Survey of data management techniques powered by artificial intelligence, J. Softw. 31 (03) (2020) 600–619.
[4] X.-F. Meng, C.-H. Ma, C. Yang, A survey of machine learning database system, J. Comput. Res. Dev. 56 (9) (2019).

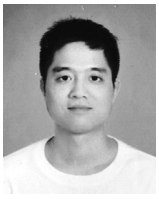
[5] W. Wang, M. Zhang, G. Chen, H.V. Jagadish, B.C. Ooi, K.L. Tan, Database meets deep learning: Challenges and opportunities, 2019.
[6] N. Nihalani, S. Silakari, M. Motwani, Integration of artificial intelligence and database management system: an inventive approach for intelligent databases, in: 2009 First International Conference on Computational Intelligence, Communication Systems and Networks, IEEE, 2009, pp. 35–40.
[7] Parsaye, Kamran, Intelligent Databases, Wiley, 1989.
[8] C. Ré, D. Agrawal, M. Balazinska, M. Cafarella, M. Jordan, T. Kraska, R. Ramakrishnan, Machine learning and databases: the sound of things to come or a cacophony of hype?, in: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, 2015, pp. 283–284.
[9] M.-K. Chai, J. Fan, X.-Y. Du, Learnable database systems: challenges and opportunities, J. Softw. 31 (3) (2020) 806–830.
[10] T. Kraska, A. Beutel, E.H. Chi, J. Dean, N. Polyzotis, The case for learned index structures, in: Proceedings of the 2018 International Conference on Management of Data, 2018, pp. 489–504.
[11] S. Moore, Gartner identifies top 10 data and analytics technology trends for 2019, 2019.
[12] H. Lan, Z. Bao, Y. Peng, A survey on advancing the dbms query optimizer: cardinality estimation, cost model, and plan enumeration, Data Sci. Eng. 6 (1) (2021) 86–101.
[13] G. Li, X. Zhou, L. Cao, AI meets database: AI4DB and DB4AI, in: Proceedings of the 2021 International Conference on Management of Data, 2021, pp. 2859–2866.
[14] X. Zhou, C. Chai, G. Li, J. Sun, Database meets artificial intelligence: a survey, IEEE Trans. Knowl. Data Eng. (2020).
[15] D. Narayanan, E. Thereska, A. Ailamaki, Continuous resource monitoring for self-predicting dbms, in: 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, IEEE, 2005, pp. 239–248.
[16] R. Taft, N. El-Sayed, M. Serafini, Y. Lu, A. Aboulnaga, M. Stonebraker, R. Mayerhofer, F. Andrade, P-store: an elastic database system with predictive provisioning, in: Proceedings of the 2018 International Conference on Management of Data, 2018, pp. 205–219.
[17] A. Pavlo, G. Angulo, J. Arulraj, H. Lin, J. Lin, L. Ma, P. Menon, T.C. Mowry, M. Perron, I. Quah, et al., Self-driving database management systems, in: CIDR, vol. 4, 2017, p. 1.
[18] Z.L. Li, C.-J.M. Liang, W. He, L. Zhu, W. Dai, J. Jiang, G. Sun, Metis: robustly tuning tail latencies of cloud systems, in: 2018 {USENIX} Annual Technical Conference ({USENIX}{ATC} 18), 2018, pp. 981–992.
[19] C. Zhang, R. Marcus, A. Kleiman, O. Papaemmanouil, Buffer pool aware query scheduling via deep reinforcement learning, preprint, arXiv:2007.10568, 2020.
[20] J. Tan, T. Zhang, F. Li, J. Chen, Q. Zheng, P. Zhang, H. Qiao, Y. Shi, W. Cao, R. Zhang, ibtune: individualized buffer tuning for large-scale cloud databases, Proc. VLDB Endow. 12 (10) (2019) 1221–1234.
[21] D.N. Tran, P.C. Huynh, Y.C. Tay, A.K. Tung, A new approach to dynamic self-tuning of database buffers, ACM Transactions on Storage 4 (1) (2008) 1–25.
[22] A.J. Storm, C. Garcia-Arellano, S.S. Lightstone, Y. Diao, M. Surendra, Adaptive self-tuning memory in db2, in: Proceedings of the 32nd International Conference on Very Large Data Bases, 2006, pp. 1081–1092.
[23] D. Van Aken, A. Pavlo, G.J. Gordon, B. Zhang, Automatic database management system tuning through large-scale machine learning, in: Proceedings of the 2017 ACM International Conference on Management of Data, 2017, pp. 1009–1024.
[24] J. Zhang, Y. Liu, K. Zhou, G. Li, Z. Xiao, B. Cheng, J. Xing, Y. Wang, T. Cheng, L. Liu, et al., An end-to-end automatic cloud database tuning system using deep reinforcement learning, in: Proceedings of the 2019 International Conference on Management of Data, 2019, pp. 415–432.
[25] C. Zhan, M. Su, C. Wei, X. Peng, L. Lin, S. Wang, Z. Chen, F. Li, Y. Pan, F. Zheng, et al., Analyticdb: real-time olap database system at alibaba cloud, Proc. VLDB Endow. 12 (12) (2019) 2059–2070.
[26] G.C. Durand, M. Pinnecke, R. Piriyev, M. Mohsen, D. Broneske, G. Saake, M.S. Sekeran, F. Rodriguez, L. Balami, Gridformation: towards self-driven online data partitioning using reinforcement learning, in: Proceedings of the First International Workshop on Exploiting Artificial Intelligence Techniques for Data Management, 2018, pp. 1–7.
[27] B. Hilprecht, C. Binnig, U. Roehm, Learning a partitioning advisor with deep reinforcement learning, preprint, arXiv:1904.01279, 2019.
[28] V. Pandey, A. van Renen, A. Kipf, I. Sabek, J. Ding, A. Kemper, The case for learned spatial indexes, preprint, arXiv:2008.10349, 2020.
[29] J. Zhang, S. Wu, Z. Tan, G. Chen, Z. Cheng, W. Cao, Y. Gao, X. Feng, S3: a scalable in-memory skip-list index for key-value store, Proc. VLDB Endow. 12 (12) (2019) 2183–2194.
[30] Wikipedia, LevelDB - Wikipedia|LevelDB, https://en.wikipedia.org/wiki/LevelDB.
[31] RocksDB, A persistent key-value store|RocksDB, https://rocksdb.org/.
[32] A. Sharma, F.M. Schuhknecht, J. Dittrich, The case for automatic database administration using deep reinforcement learning, preprint, arXiv:1801.05643, 2018.
[33] M. Kahng, D. Fang, D.H. Chau, Visual exploration of machine learning results using data cube analysis, in: Proceedings of the Workshop on Human-in-the-Loop Data Analytics, 2016, pp. 1–6.
[34] H. Yuan, G. Li, L. Feng, J. Sun, Y. Han, Automatic view generation with deep learning and reinforcement learning, in: 2020 IEEE 36th International Conference on Data Engineering (ICDE), IEEE, 2020, pp. 1501–1512.
[35] Y. Han, G. Li, H. Yuan, J. Sun, An autonomous materialized view management system with deep reinforcement learning, in: 2021 IEEE 37th International Conference on Data Engineering (ICDE), IEEE, 2021, pp. 2159–2164.
[36] X. Liang, A.J. Elmore, S. Krishnan, Opportunistic view materialization with deep reinforcement learning, preprint, arXiv:1903.01363, 2019.
[37] J. LeFevre, J. Sankaranarayanan, H. Hacigumus, J. Tatemura, N. Polyzotis, M.J. Carey, Opportunistic physical design for big data analytics, in: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, 2014, pp. 851–862.
[38] G. Li, X. Zhou, S. Li, Xuanyuan: an ai-native database, IEEE Data Eng. Bull. 42 (2) (2019) 70–81.
[39] B. He, I. Ounis, Query performance prediction, Inf. Sci. 31 (7) (2006) 585–594.
[40] M. Akdere, U. Çetintemel, M. Riondato, E. Upfal, S.B. Zdonik, Learning-based query performance modeling and prediction, in: 2012 IEEE 28th International Conference on Data Engineering, IEEE, 2012, pp. 390–401.
[41] W. Wu, Y. Chi, H. Hacígümüş, J.F. Naughton, Towards predicting query execution time for concurrent and dynamic database workloads, Proc. VLDB Endow. 6 (10) (2013) 925–936.
[42] S. Krishnan, Z. Yang, K. Goldberg, J. Hellerstein, I. Stoica, Learning to optimize join queries with deep reinforcement learning, preprint, arXiv:1808.03196, 2018.
[43] M.V. Mannino, P. Chu, T. Sager, Statistical profile estimation in database systems, ACM Comput. Surv. 20 (3) (1988) 191–221.
[44] S. Lakshmi, S. Zhou, Selectivity estimation in extensible databases-a neural network approach, in: VLDB, vol. 98, 1998, pp. 24–27.
[45] H. Liu, M. Xu, Z. Yu, V. Corvinelli, C. Zuzarte, Cardinality estimation using neural networks, in: Proceedings of the 25th Annual International Conference on Computer Science and Software Engineering, 2015, pp. 53–59.
[46] R.R. Selmic, F.L. Lewis, Neural-network approximation of piecewise continuous functions: application to friction compensation, IEEE Trans. Neural Netw. 13 (3) (2002) 745–751.
[47] A. Kipf, D. Vorona, J. Müller, T. Kipf, B. Radke, V. Leis, P. Boncz, T. Neumann, A. Kemper, Estimating cardinalities with deep sketches, in: Proceedings of the 2019 International Conference on Management of Data, 2019, pp. 1937–1940.
[48] A. Ganapathi, H. Kuno, U. Dayal, J.L. Wiener, A. Fox, M. Jordan, D. Patterson, Predicting multiple metrics for queries: better decisions enabled by machine learning, in: 2009 IEEE 25th International Conference on Data Engineering, IEEE, 2009, pp. 592–603.
[49] L.-Y. Bi, S. Wu, G. Chen, L.-D. Shou, T.-L. Hu, Database query cost prediction using recurrent neural network, J. Softw. 29 (3) (2018) 799–810.
[50] Z. Yang, E. Liang, A. Kamsetty, C. Wu, Y. Duan, X. Chen, P. Abbeel, J.M. Hellerstein, S. Krishnan, I. Stoica, Deep unsupervised cardinality estimation, in: Proceedings of the Vldb Endowment, 2019.
[51] J. Sun, G. Li, An end-to-end learning-based cost estimator, Proc. VLDB Endow. 13 (3) (2019) 307–319.
[52] D. Mottin, M. Lissandrini, Y. Velegrakis, T. Palpanas, New trends on exploratory methods for data analytics, Proc. VLDB Endow. 10 (12) (2017) 1977–1980.
[53] S. Jain, B. Howe, J. Yan, T. Cruanes, Query2vec: an evaluation of nlp techniques for generalized workload analytics, preprint, arXiv:1801.05613, 2018.
[54] S. Iyer, I. Konstas, A. Cheung, L. Zettlemoyer, Summarizing source code using a neural attention model, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2016, pp. 2073–2083.
[55] R. Bordawekar, B. Bandyopadhyay, O. Shmueli, Cognitive database: a step towards endowing relational databases with artificial intelligence capabilities, preprint, arXiv:1712.07199, 2017.
[56] S.I. Wiki, Sql injection cheat sheet, 2013.
[57] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, et al., Spider: a large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task, preprint, arXiv:1809.08887, 2018.
[58] C. Liang, M. Norouzi, J. Berant, Q.V. Le, N. Lao, Memory augmented policy optimization for program synthesis and semantic parsing, in: Advances in Neural Information Processing Systems, 2018, pp. 9994–10006.
[59] C.T. Hemphill, J.J. Godfrey, G.R. Doddington, The atis spoken language systems pilot corpus, in: Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990, 1990.
[60] V. Zhong, C. Xiong, R. Socher, Seq2sql: generating structured queries from natural language using reinforcement learning, preprint, arXiv:1709.00103, 2017.
[61] X. Xu, C. Liu, D. Song, Sqlnet: generating structured queries from natural language without reinforcement learning, preprint, arXiv:1711.04436, 2017.
[62] P. He, Y. Mao, K. Chakrabarti, W. Chen, X-sql: reinforce schema representation with context, preprint, arXiv:1908.08113, 2019.
[63] Y. Bai, Y. Zhang, Artificial intelligence in database security, Int. J. Pattern Recognit. Artif. Intell. 18 (01) (2004) 3–17.
[64] P. Tang, W. Qiu, Z. Huang, H. Lian, G. Liu, Sql injection behavior mining based deep learning, in: International Conference on Advanced Data Mining and Applications, Springer, 2018, pp. 445–454.

[65] Z. Lin, X. Li, X. Kuang, Machine learning in vulnerability databases, in: 2017 10th International Symposium on Computational Intelligence and Design (ISCID), vol. 1, IEEE, 2017, pp. 108–113.
[66] R.C. Fernandez, Z. Abedjan, F. Koko, G. Yuan, S. Madden, M. Stonebraker, Aurum: a data discovery system, in: 2018 IEEE 34th International Conference on Data Engineering (ICDE), IEEE, 2018, pp. 1001–1012.
[67] P. Colombo, E. Ferrari, Efficient enforcement of action-aware purpose-based access control within relational database management systems, IEEE Trans. Knowl. Data Eng. 27 (8) (2015) 2134–2147.
[68] N.M. Sheykhkanloo, A learning-based neural network model for the detection and classification of sql injection attacks, Int. J. Cyber Warf. Terror. 7 (2) (2017) 16–41.
[69] L.O. Batista, G.A. de Silva, V.S. Araújo, V.J.S. Araújo, T.S. Rezende, A.J. Guimarães, P.V.d.C. Souza, Fuzzy neural networks to create an expert system for detecting attacks by sql injection, preprint, arXiv:1901.02868, 2019.
[70] Z. Li, X. Wang, J. Li, Q. Zhang, Deep attributed network representation learning of complex coupling and interaction, Knowl.-Based Syst. 212 (2021) 106618.



Benyuan Zou received the B. E. degree in network engineering from Shandong University of Science and Technology, Taian, China. He is currently pursuing the M. E. degree with Kunming University of Science and Technology, Kunming China. His current research interests include database and machine learning.

Jinguo You received the M.Sc. degree from Kunming University of Science and Technology, Kunming, China, in 2005, and the Ph.D. degree from South China University of Technology, Guangzhou, China, in 2008. He is currently an Associate Professor with the Faculty of Information Engineering and Automation, Kunming University of Science and Technology, China. His research interests mainly include big data, data warehousing, data mining. He is a senior member of China Computer Federation (CCF), a member of the Technical Committee of Information System, CCF and a communication member of the Technical Committee of Database, CCF.



Quankun Wang received the B. E. degree in software engineering from Nanyang Institute of Technology, Nangyang, China. He is currently pursuing the M. E. degree with Kunming University of Science and Technology, Kunming, China. His current research interests include online analytical processing and cloud computing.

Xinxian Wen received the B. E. degree in Electronic Information Engineering from Nanjing University of Post and Telecommunications, Nanjing, China. She is currently pursuing the M. E. degree with Kunming University of Science and Technology, Kunming China. Her current research interests include machine learning and data mining.

Lianyin Jia received the Ph.D. degree from South China University of Technology, Guangzhou, China, in 2013. He is currently an Associate Professor in the Faculty of Information Engineering and Automation, Kunming University of Science and Technology, Kunming, China. His current research interests include databases, data mining, information retrieval, and parallel computing.