Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 6118113, 11 pages
https://doi.org/10.1155/2021/6118113

Research Article

Research on Query Optimization of Classic Art Database Based on Artificial Intelligence and Edge Computing

Xiang Dong[1] **and Lijia Zeng**[2]

[1]*School of Architecture and Design, Jiangxi University of Science and Technology, Ganzhou, 341000 Jiangxi, China*
[2]*Department of Art and Design, Dong-Eui University, Busanjin-gu, 47340 Busan, Republic of Korea*

Correspondence should be addressed to Xiang Dong; 9120010035@jxust.edu.cn

Received 22 May 2021; Revised 17 June 2021; Accepted 20 July 2021; Published 10 August 2021

Academic Editor: Wenqing Wu

With the changes and development of the social era, my country's classic art is slowly being lost. In order to more effectively inherit and preserve classic art, the collection and sorting of classic art data through modern information technology has become a top priority. Database storage is a good way. However, as the amount of data grows, the requirements for computing processing power and query speed for massive amounts of data and information are also increasing day by day. Faced with this problem, this article is aimed at studying the optimization of database queries through effective algorithms to improve the efficiency of data query. Based on the traditional database query optimization algorithm, this article improves on the traditional algorithm and proposes a semi-join query optimization algorithm, which reduces the number of connection cards and the number of columns and uses the number of blocks that participate in the semi-link algorithm connection and preconnection preview and selection. And other functions reduce the size of the participating block, and the connection sent between sites reduces the cost of sending between networks. The graph data query optimization algorithm is used to optimize the graph data query in the database to reduce the extra task overhead and improve the system performance. The experimental results of this paper show that through the data query optimization algorithm of this paper, the additional task overhead is reduced by 19%, the system performance is increased by 22%, and the data query efficiency is increased by 31%.

1. Introduction

Classic art has been carrying the prosperity and replacement of the Chinese nation since ancient times and is a cultural treasure handed down by the Chinese nation. However, as the diversified development of art and the acceleration of the modernization process, coupled with the slowdown of modern attention to it, some classic art and works that have been handed down for a long time are disappearing continuously. This urgently requires us to keep up with the changes of the times, systematically collect and organize data, and establish relevant databases. To show the classic art of the Chinese clan in a new situation can better pass on the classic art. Building a database to protect relevant data more effectively is a necessary trend for modern information technology to be used in various fields. Due to the large and complex data volume of classical art inherited by China for thousands of years and a large number of data and query operation user groups, a wide range of data queries will bring great challenges to the performance of the database and the processing cost of the database query operation. It has become a huge problem that needs to be solved urgently, and optimizing database query operations has become very important and meaningful.

In order to conduct research in the field of artificial intelligence, Pigozzi et al. made a key overview of the existence and application of the concept of "preference" in artificial intelligence. Preference is the core concept of decision-making, and it has been extensively studied in disciplines such as economics, operations research, decision analysis, psychology, and philosophy. However, in recent years, in various fields from recommendation systems to automatic planning and from nonmonotonic reasoning to computational social choice and algorithmic decision theory, it has also become

an important subject of computer science research and application, especially in the field of artificial intelligence. The survey basically covers the basic knowledge of preference modeling, the use of preference in reasoning and argumentation, the problem of compact representation of preferences, preference learning, and the use of nontraditional preference models based on extended logical language. But the concept he put forward is too advanced for modern technology to be effectively implemented [1, 2]. In order to meet the urgent needs of customers, database vendors have been pushing advanced analysis techniques into the database. Most major DBMSs use user-defined aggregation (UDA) (a data-driven operator) to implement analysis techniques in parallel. In statistical algorithms, most of the work is performed by iterative transitions in larger states. UDA alone is not enough to implement statistical algorithms and cannot be naively partitioned due to data dependence. Generally, this type of statistical algorithm first needs to be preprocessed to establish a large state and needs to be postprocessed after statistical inference. Li et al. proposed General Iterative State Transition (GIST), a new database operator for parallel iterative state transitions on large states. GIST receives the state constructed by UDA and then performs several rounds of transitions on the state until convergence. The final UDA performs postprocessing and result extraction. Li et al. believe that the combination of UDA and GIST (UDA-GIST) unifies data parallel and state parallel processing in a single system and can effectively implement statistical algorithms. However, the general iterative state transition operator they proposed may cause errors in the data during the iteration process, thereby affecting the result of the operation [3]. Moreau et al. introduced the ClusterXplain method, which is a way to help users better understand their query results. These results are constructed using a clustering algorithm and described using a personal vocabulary. They proposed a clear and vague version of this method, with the goal of finding what the elements of a cluster have in common and distinguishing them from the elements of other clusters. The data that Moreau et al. consider to characterize each answer cluster is not limited to the attributes used in the query, thus revealing unexpected relevance to users. They use natural language terms to provide users with characterization to describe the obtained clusters. But his method is too simple to meet the needs of users [4].

This article uses a distributed database to manage data. The innovations of this article are as follows: (1) Commonly used distributed query optimization techniques such as semijoin and direct join algorithms still have certain defects and room for improvement, based on the basis of query optimization technology. Making some query optimization improvements can effectively reduce database query response time and improve query efficiency. (2) Based on artificial intelligence, deep learning methods are used to determine the mapping relationship between system states and behaviors. This avoids many calculations in the augmented learning process and provides a quick solution. It is possible to send as much data as possible while reducing the loss of data packets while reducing power consumption.

2. Based on Artificial Intelligence Database Query Optimization Algorithm

2.1. Artificial Intelligence. Artificial intelligence is an emerging technological science, which researches and develops theories, methods, technologies, and application systems for simulating and expanding human intelligence [5]. Artificial intelligence is a branch of computer science that attempts to create a new type of intelligent machine that can understand the essence of intelligence and can react in the same way as human intelligence [6, 7]. Research areas include robotics, language recognition, image recognition, natural language processing, and expert systems [8–10]. Since the birth of artificial intelligence, theory and technology have become more and more mature, and its scope is also expanding. The technological products that artificial intelligence will bring in the future can become "containers" of human wisdom. It can simulate the information processing of human consciousness and thought, not only limited to logical thinking, but also imaginative and inspirational thinking can be considered to promote the pioneering development of artificial intelligence [11]. The main content of artificial intelligence research includes knowledge representation, automatic methods of reasoning and retrieval, machine learning and knowledge acquisition, knowledge processing systems, natural language understanding, computer vision, intelligent robots, and automatic programming [6, 12]. The focus of artificial intelligence is to enable computers to perform tasks that must be completed by human advanced intellectual activities [9, 13].

2.2. Edge Computing. Edge computing refers to the "sinking" of cloud computing functions at the network edge, which is a new computer model used with network edge devices [14]. The "end" of edge computing is the data source generated by the end of the network terminal [15, 16]. For the resources between the data paths in the cloud data center, the basic idea is to perform computer work on the computer resources near the data source. Edge computing and cloud computing are complementary to each other [17, 18]. Portable computing chip is a computing chip. The traditional wireless access network is located between the wireless access point and the wired network, which can provide end users with higher bandwidth and waiting for data service, which can save time and reduce bandwidth, data service requirements for network search [19, 20].

2.3. Deep Learning Algorithms Based on Artificial Intelligence

2.3.1. Adaptive Grouping Algorithm. In the cognitive network, users access and send data according to the state of the channel. When the channel status is better, more data streams will be transmitted on it. Therefore, the amount of data sent on the channel will maintain a continuous and steady trend. On the other hand, a channel with a more general state will only be accessed by users when other channels are occupied, and its traffic value will show indirectness and hopping. According to the above two characteristics, the data stream is divided into two groups [21, 22]. Using adaptive grouping (AG) method to group different channel traffics,

data with similar characteristics can be grouped together. The algorithm is as follows:

$$\bar{s}^{t} = \frac{1}{R}\sum_{n=1}^{R} s_n^t. \tag{1}$$

Calculate the average value of the traffic on t communication channels, respectively, where t is the data on the t-th communication channel and R is the number of data on the communication channel. Calculate the variance of the traffic data on each communication channel separately and use the variance to express the volatility of the data:

$$\phi_t^2 = \frac{1}{R}\sum_{n=1}^{R} \left(s_n^t - s^t\right)^2. \tag{2}$$

According to the variance, each data stream is grouped, and the volatility is used to indicate the stability and fluctuation of the data stream. Calculate the mean variance of t data streams and use this as the threshold for grouping.

$$\bar{\phi}^{2} = \frac{1}{R}\sum_{m=1}^{t} \phi_t^2. \tag{3}$$

2.3.2. Deep Learning Algorithm. When it comes to wireless data transmission, full consideration will be given to packet loss, power consumption, and system performance. In an actual wireless network, the logical assumption is that the environmental information is unknown; that is, the retransmission node cannot know in advance that environmental conditions may occur. Therefore, the system model modeled as MDP uses the amplification method of deep learning algorithms to train relay nodes to learn environmental state transition information and guide node operations [23, 24]. The strategy selection method has been improved to obtain better state behavior data to allow for the balance between exploration and use when retrieving state behavior. In addition, based on behavioral data from learning enhancements, deep learning methods are used to establish a mapping relationship between situations and behaviors to achieve the goal of rapid resolution.

$$\pi^*(e_n, v_n) = \max \sum_{e_n+1} T_{e_n e_n+1}^{v_n}(c^{\pi}(e_n+1)), \tag{4}$$

where the T value in the learning algorithm is the evaluation value of the state and behavior, and the goal is to maximize the utility of the system.

When the scale of the system is large, how to realize the exploration and utilization of the strategy is the key issue in the deep learning algorithm. How to effectively choose behavior will directly affect the convergence speed of the algorithm and the performance of the system. This article introduces the behavior evaluation based on behavior and increases the index value to select the behavior that maximizes revenue, namely,

$$\text{Index}(e_n, v) = \ell\sqrt{\frac{2\text{In}}{P_n(m)}\ \min\left\{\frac{1}{6}, c_n(m)\right\}}, \tag{5}$$

where $P_n(m)$ represents the number of times the e_n behavior is selected after m behavior selections, ℓ is a constant greater than 0, and $c_n(m)$ represents the deviation factor to reflect the volatility of the value.

2.4. Query Optimization Based on Semi-Join Algorithm

2.4.1. Cost Function. Query optimization refers to the implementation of specific optimization measures to minimize the total time and cost of the query. There are generally two basic types: one is to optimize the query response time during the query process and the other is to optimize the execution cost during the query process. Optimizing the query response means that the processing time from the delivery of the query statement to the feedback of the query result to the user should be as short as possible. The focus of cost-based optimization is to minimize the use of system resources in the query process.

The cost of a distributed execution strategy can be expressed as the total time it takes to complete the query or as the response time from the start of the query to the completion of the query. The formula for calculating the total time spent is as follows:

$$D_{\text{tt}} = D_{\text{CPU}} \times \text{its} + D_E \times E_S + D_{\text{MSG}} \times \text{msgs} + D_R \times \text{bts}, \tag{6}$$

where D_{tt} represents the total time taken to complete the query, D_{CPU} is the instruction execution time of the query command CPU, and D_E is the time spent reading and writing database data on the local disk during query execution.

When the response time becomes the main consideration performance index of the optimization program, it is necessary to comprehensively consider the problems of local processing and parallel communication. The general response time calculation method is as follows:

$$\begin{aligned} D_{Rt} = {} & D_{\text{CPU}} \times \text{seq\_its} + D_E \times \text{seq\_}E_S + D_{\text{MSG}} \times \text{seq\_msgs} \\ & + D_R \times \text{seq\_bts}, \end{aligned} \tag{7}$$

where seq represents the maximum number required to complete the query serially.

The selection factor is also an important statistic in the database. Its value is a real number between 0 and 1. The expression of the selection factor TK connected by the relations M and N is as follows:

$$TK(M, N) = \frac{T(M) \infty T(N)}{T(M) \times T(N)}, \tag{8}$$

where $T(M)$ represents the cardinality of the range and $T(N)$ represents the number of tuples contained.

For the intermediate results generated by the calculation query, in order to simplify the calculation amount, two simple assumptions are often made on the database, and all the

attributes in the relationship are distributed and uniform with independent attribute values.

Operation result formula is as follows:

$$T(\rho_K(M)) = TK_N(K) \times T(M). \tag{9}$$

The cardinality formula of the relations M and N is as follows:

$$T((M \times N)) = T(M) \times T(N). \tag{10}$$

The first step of query optimization is to generate the search space of the query execution plan. The second step is to select the search strategy to calculate the cost of the equivalent execution plan in the search space one by one, and select the best execution under the current search strategy from the calculated results plan.

2.4.2. Semi-Join Algorithm Query Optimization. Whether it is a central database system or a distributed database system, selection, viewing, and connection are the three most commonly used functions. There is no difference between central database and distributed database options and views, which are executed at the local site. Considering the physical distribution of the sites in the distributed database system and the subdivision characteristics of the relationship, the connection operation on it is very complicated. When the two relational objects associated with the connection operation are located on different sites, in order to complete the connection operation, data transmission between the sites is inevitable. When the two relational objects associated with the connection operation are located on different sites, in order to complete the connection operation, data transmission between the sites is inevitable. In order to reduce communication costs, the intuitive idea is to avoid unnecessary tuple transmission on the network. Especially in the wide area network, the communication cost accounts for a large part of the query cost, and the semi-join algorithm is optimized based on the idea of reducing the amount of data transmission.

The semi-join operation is derived from the projection and the connection operation. The projection operation realizes the reduction of the cardinality of the connection attribute, and the connection operation realizes the reduction of the number of tuples of the left connection relationship. Its definition and operation process semi-join (semi-join) is a reduction operation on the attribute column of the full join result. Assume that the relations M and N have the same attribute join-attr, namely,

$$\begin{aligned} M \infty N &= \pi_M(M \infty N) = M \infty \pi_{\text{join-attr}}(N), \\ N \infty M &= \pi_N(N \infty M) = N \infty \pi_{\text{join-attr}}(M), \end{aligned} \tag{11}$$

where $M \infty N$ or $N \infty M$ is the description of the semi-join operation of the relations M and N.

The semi-join operation can be understood as using another relation to reduce the number of tuples of its own relationship, and the semi-join operation does not satisfy the commutative law; that is, $M \infty N \neq N \infty M$. When performing a semi-join operation, the site information and fragmentation statistics of the database system are used to select $M \infty N$ or $N \infty M$.

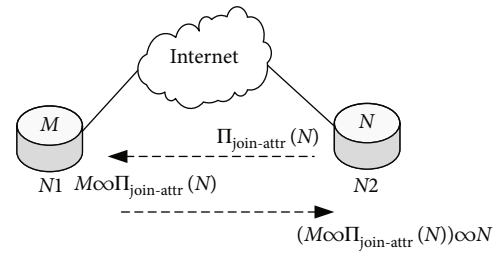


Figure 1: Flow chart of semi-join algorithm optimization.

The execution process of the semi-join operation includes the projection operation and the connection operation of the connection relationship. The execution process is shown in Figure 1. The execution steps of the semi-connection are shown in Figure 2.

2.4.3. Cost Estimation of Semi-Join Algorithm. It can be seen from the execution process of the semi-connection that compared to the full connection, the semi-connection has one more data transmission process than the full connection, but in most cases, the total amount of data transmitted by the semi-connection is much less than that of the full connection. At this time, the use of a semi-connected algorithm can reduce the communication cost. The estimated cost of the semi-join operation is as follows:

$$D_{\text{join-attr}} = b_0 + b_1 \times S(\text{join-attr}) \times T(M). \tag{12}$$

The case where the semi-join operation is applicable is that only a small number of tuples in the relation M participate in the connection with the relation N. At this time, the semi-join algorithm can be used to reduce the number of tuples in the relation M. When there are many connection relations, there are many types of semi-connections. At this time, it is necessary to calculate the communication cost of all semi-connection forms, select the semi-connection scheme with the least cost from them, and select the site with the least data transmission cost as the connection operation site.

2.5. Graph Data Query Optimization Algorithm

2.5.1. PageRank Hyperlink Analysis Algorithm. Traditional databases have natural advantages in processing phenotypic data, but there are many difficulties in processing graph data. In terms of query optimization, it is mainly divided into two levels: system-level optimization and application-level optimization. The PageRank algorithm uses a well-known random walk model to model the behavior of a web browser: any web browser may take two possible actions when browsing a web page. The first action is click on a hyperlink in the current webpage to jump to another webpage. The second action is to close the current webpage directly and then randomly open a new webpage in the browser to continue browsing [25, 26].

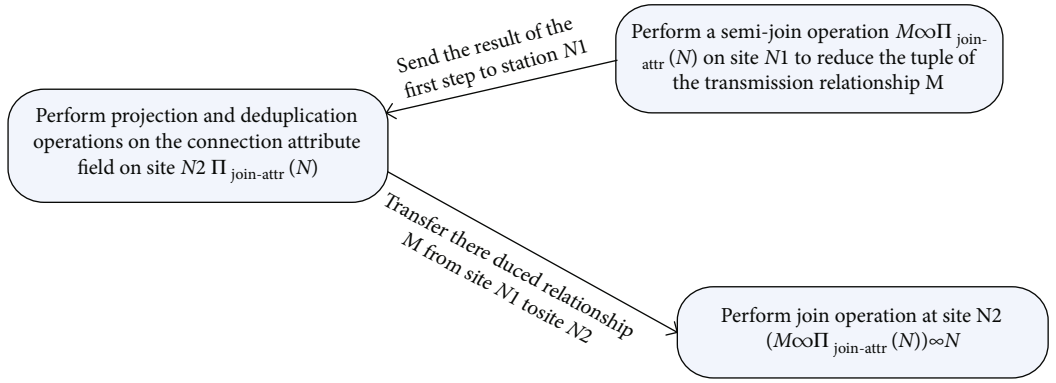


Figure 2: Diagram of the execution steps of the semi-connected algorithm.

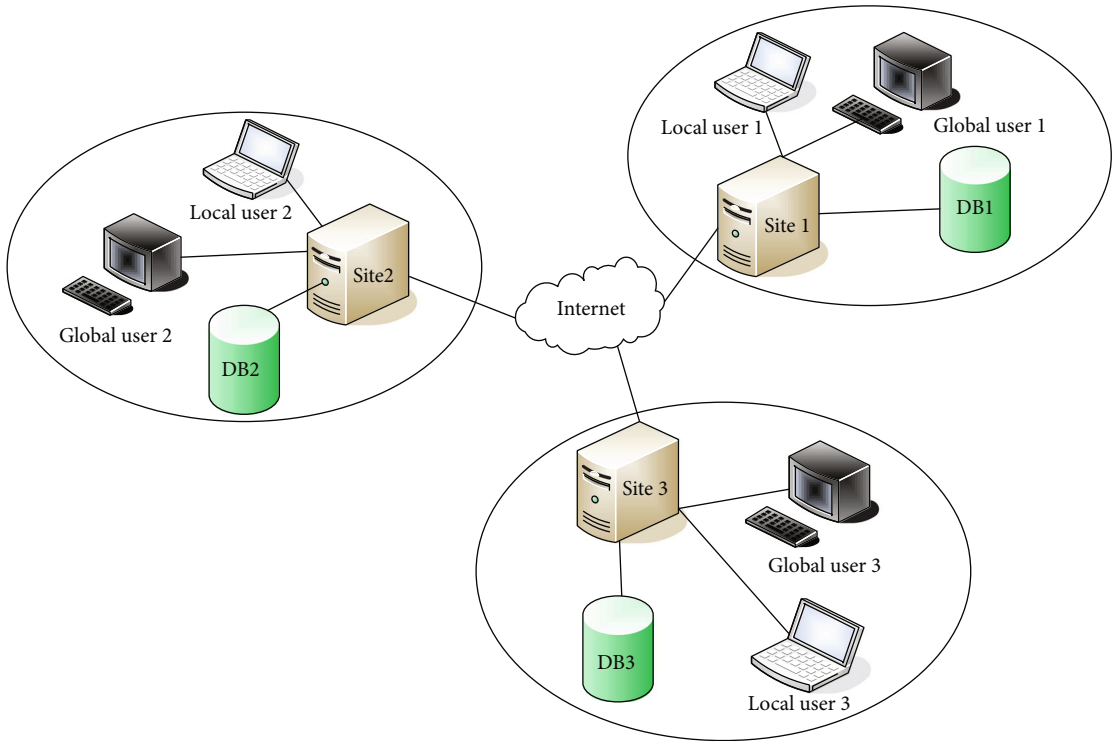


Figure 3: Physical structure diagram of distributed database system.

According to this random walk model, the PageRank algorithm calculates the score expression as follows:

$$GE\left(g_j\right) = \frac{1-a}{M} + a \times \sum_{g_i \in N\left(g_j\right)} \frac{GE(g_i)}{K(g_i)}, \tag{13}$$

where g_i and g_j represent each webpage, M represents the number of all webpages in the Internet, a represents the retention coefficient, $GE(g_j)$ represents the score of the webpage g_j, $N(g_j)$ represents the collection of all webpages constituting (g_j), and $K(g_j)$ represents the number of hyperlinks derived from the webpage g_j.

During the execution of the PageRank algorithm, these web pages converge slowly and require more iterations, but for most web pages, only a few generations are needed to converge to a stable score. Therefore, it is obviously not an efficient way to repeat the calculation of the entire graph for a few unconverged nodes.

Therefore, we can optimize it and only consider webpage nodes that have not yet converged in the calculation process, namely,

$$F^{t+1} = SF^t, \tag{14}$$

where the score value of each web page node is recorded in the vector F. If a distinction is made between currently converged nodes and unconverged nodes, the expression is as follows:

$$F_x^{t+1} = S_x^t \times F_x^t + S_y^t \times F_y^t, \quad (15)$$

where F_x^t represents the set of nodes that have converged in the t-th iteration and F_x^{t+1} represents those nodes that have not converged in the t-th iteration. The matrix block S_x^t represents the connection mode from the set of unconverged nodes to the set of unconverged nodes in the graph, and some new nodes will converge in this round of iterations. Therefore, the dimension of the vector F_x will continue to shrink, and the amount of calculation for the entire algorithm will also become smaller.

3. Distributed Database Query Optimization Operation Experiment

3.1. Distributed Database. A distributed database is physically composed of multiple databases connected to each other and distributed in different physical locations [27], which can manage the data stored in each physical location independently of other physical locations, and is logically managed by the database management system. Since the data has been scattered in different physical locations, the database can be easily expanded, and distributed databases can be easily accessed from different networks [24, 28]. Compared with centralized databases, this database is more secure. The physical independence of distributed databases often triggers queries in distributed systems that span multiple sites. This includes subquery site allocation, timeline settings, and multisite query results. Issues such as merging routing make the query function of a distributed database much more complicated than that of a central database [29, 30]. The physical structure diagram of the distributed database system is shown in Figure 3.

Figure 3 shows the physical structure of a typical DDBS. Computers or servers located at various physical sites and database systems deployed on them are the basic units of distributed database systems. The sites, databases, and local users on the site are controlled by computers or servers on the site. Different site network communication links communicate, and the network can be a local area network or a wide area network.

3.2. Database Query Steps. Figure 4 is the work flow chart of the research on data query optimization in this paper. To generate the final physical query plan, the combined cost of each query operation is calculated, and the physical query plan with the smallest combined cost is selected as the final query plan, which requires every query operation to be an estimated cost is given in advance, so the establishment of the query operation cost estimation model is crucial to the correctness of the final query plan selection. The acquisition process of statistics is carried out in the preprocessing stage of the operation of the entity-based inferior data management system. It is not in the query compilation stage and does not take up the actual query time. The physical query plan is generated in the query compilation process, so we have to consider it as the time efficiency of the optimal physical query plan selection algorithm.

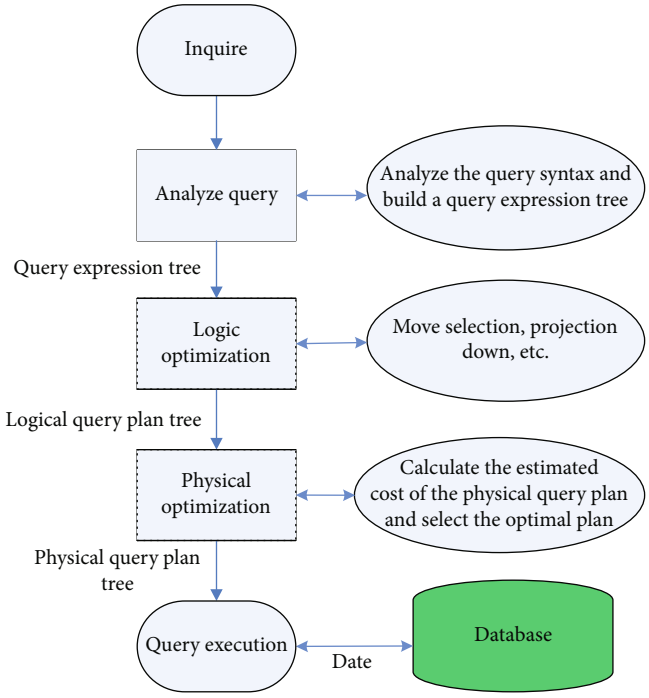


Figure 4: Data query optimization research work flow chart.

Table 1: Simulation parameter design.

Parameter	Value/description
Signal-to-noise ratio threshold (dB)	SNR = [−1.25,−0.36, 0.79]
Vibration length (s)	F_d=18
Arrival rate	$\lambda = [0.1,0.3, \cdots, 0.9]$
Bit error rate constraint	BER = 10^{-3}
Learning factor	$\theta = 0.19$
Discount factor	$\delta \in (0, 1)$

4. Database Query Optimization Algorithm Based on Artificial Intelligence

4.1. Deep Learning Algorithm Based on Artificial Intelligence. Use detailed learning methods to get the best behavior information, save it in the search panel, and then use this information to monitor SAE web training. The input part of training is the saved learned system state, and the output label part is the behavior in that state. There are many states involved in the system. It is more difficult to find an optimal decision for each state. So, use the method in this article to compare performance. The simulation parameter settings are shown in Table 1.

Figure 5 shows the average comparison of system power consumption using different data arrival rates and different algorithms. The figure shows that the RS method is relatively stable. Because the RS method is not affected by system status information, the data arrival rate basically does not affect the choice of RS transmission mode, but will greatly affect the functions of the other three methods. When the amount of data in the system is large, the pressure in the buffer becomes

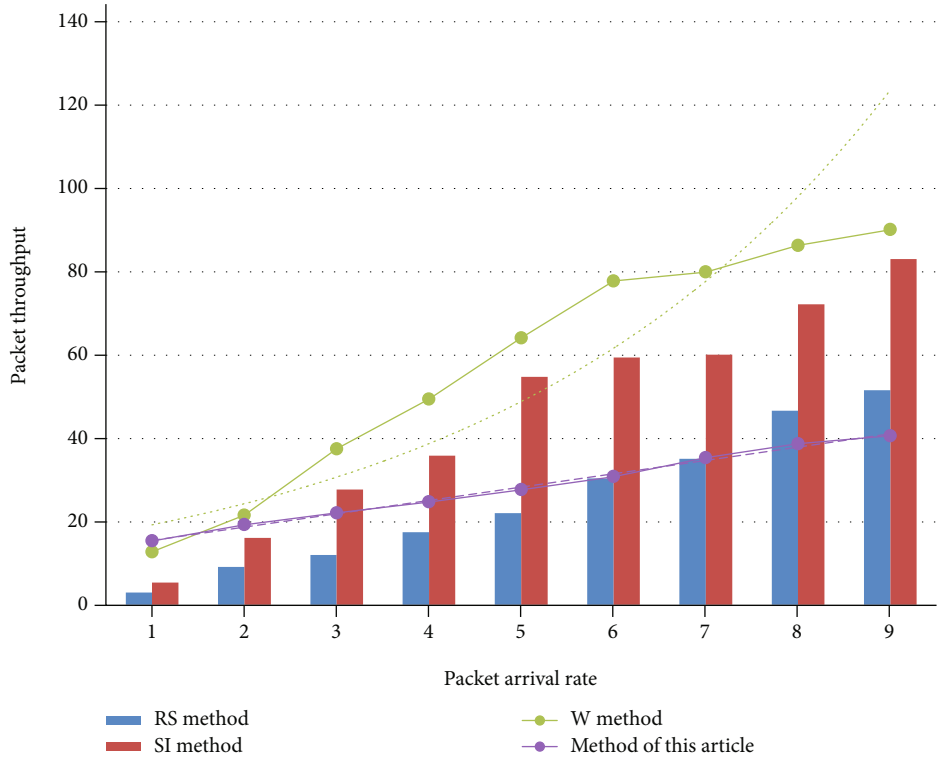


Figure 5: Energy consumption comparison chart.

Table 2: TPCH operation result graph.

Statement number	2G data before improvement	Improved 2G data	5G data before improvement	Improved 5G data
1	6.84	3.25	20.52	9.46
2	18.46	8.48	69.34	33.06
3	20.35	9.37	76.78	37.28
4	9.25	4.92	31.45	14.12
5	4.18	2.01	41.62	17.91

greater, and the relay node sends more data to reduce the pressure in the buffer and finally chooses a better transmission method to increase power consumption. The energy consumption of the three methods first showed a trend of rapid growth and then showed a moderate increase. The more data in the buffer, the more power is consumed, and the faster the power curve grows. Due to the limited cache space, when the amount of data reaches the maximum cache strength, the cache pressure will not increase, and eventually, the power consumption will stabilize.

4.2. Query Optimization Analysis Based on Semi-Join Algorithm. According to the proposed improved semi-connected algorithm, it is applied to the GreatDDB database system. In order to analyze the performance of the improved semi-connected algorithm in the database system, the performance test of the GreatDDB distributed database system with semi-connected algorithm and without semi-connected algorithm is compared. Analyze the improvement effect of the algorithm through the time taken by the system to execute the TPC-H sentence before and after the improvement of the algorithm. Execute 5 different TPCH statements in the GreatDDB environment to analyze the optimization results of the algorithm before and after the improvement.

Table 3: Data query improvement rate after improved algorithm.

Statement number	2G data promotion rate	5G data promotion rate
1	52.49%	53.90%
2	54.06%	52.32%
3	53.96%	51.45%
4	46.81%	55.10%
5	51.91%	56.97%

It can be seen from Table 2 and Table 3 that the execution speed of the improved connection algorithm is significantly higher than that of the previous connection algorithm, and it takes less time to execute the same TPCH test statement; that is, it can return results faster. Improve when the amount

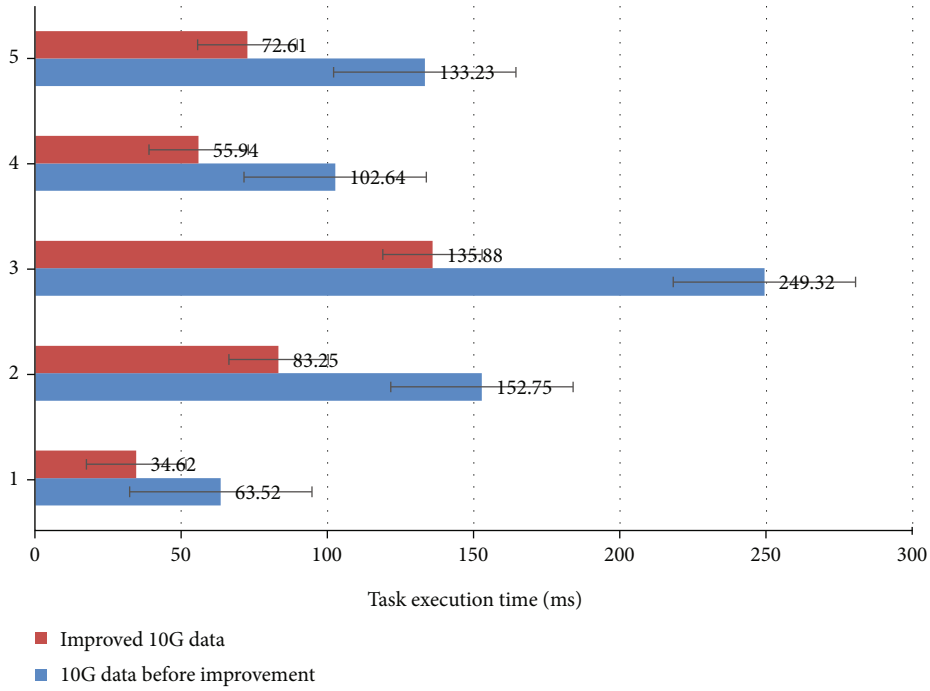


Figure 6: Test result comparison chart.

of data is large. The optimization effect of the latter algorithm is more obvious, because when the amount of data is large, more connection operations are involved, which shows that the connection algorithm does have practical significance in the case of multitable connection. Table 2 shows the time improvement ratio of the improved algorithm when executing test sentences. It can be seen that the improved algorithm runs better and the network transmission cost is reduced more obviously.

Figure 6 is a comparison chart of 10G data size data results, using the improved algorithm and the unimproved algorithm to execute sentences in the same amount of time. This is mainly because the statement only involves the selection and projection operations of a table, and these operations can be executed at the local site where the table line item is involved; the statement does not involve the connection operation between multiple tables, so there is no intersite table due to the data transmission cost incurred by the connection. The improved algorithm is optimized based on the multitable connection. Since only a single table is involved, the connection algorithm is not called during query execution. Therefore, when the parameter use_update_GreatDDB is equal to 0 or 1, and the amount of data is the same, there is not much difference in the time spent using improved and unimproved connection algorithms for query operations.

4.3. Graph Data Query Optimization Analysis. Perform a query experiment of graph pattern matching on the US patents data set. Construct 6 different graph pattern types (triangle, diamond, cross, hexagon, double diamond, and octagon, respectively) and then use the MATCH clause to describe and query. We choose Neo4j as the comparison object. There is no table in Neo4j, but the structure of the graph (nodes, edges, attributes, etc.) is directly stored. The Neo4j database system is completely implemented in Java language, supports standard ACID features, and also provides a query language for users to use. Among them, the description method of graph mode is very similar to the MATCH clause introduced in this article. We repeated the same experiment on Neo4j and GraphLab to compare the gap in query performance between GraphView and graph databases.

Table 4: Connection summary diagram.

Connection relationship	Number of tuples	Number of connection attribute value sets	Tuple length
RE	15	10	8
TU	40	50	5

Table 4 is the statistical data table of the relationship in the graph data query. In order to implement the PageRank algorithm described above, the performance comparison and analysis of the two will be shown later.

For each graph mode, we have tested multiple sets of data. From the final result shown in Figure 7, we can see that the advantages of the graph data query optimization algorithm in this article are very obvious. The method in this paper automatically converts the graph mode equivalently, and the generated execution plan is less expensive, but in Neo4j, there is no such optimization technology. It is executed completely according to the user's input. The specific graph traversal sequence is in the database and the user. The input order in the query statement is completely the same, but because the user's input only gives a description of the graph mode, its order may not be the best performance in terms of execution efficiency, so the query performance of Neo4j is good or bad. Guarantee the best. GraphLab is a

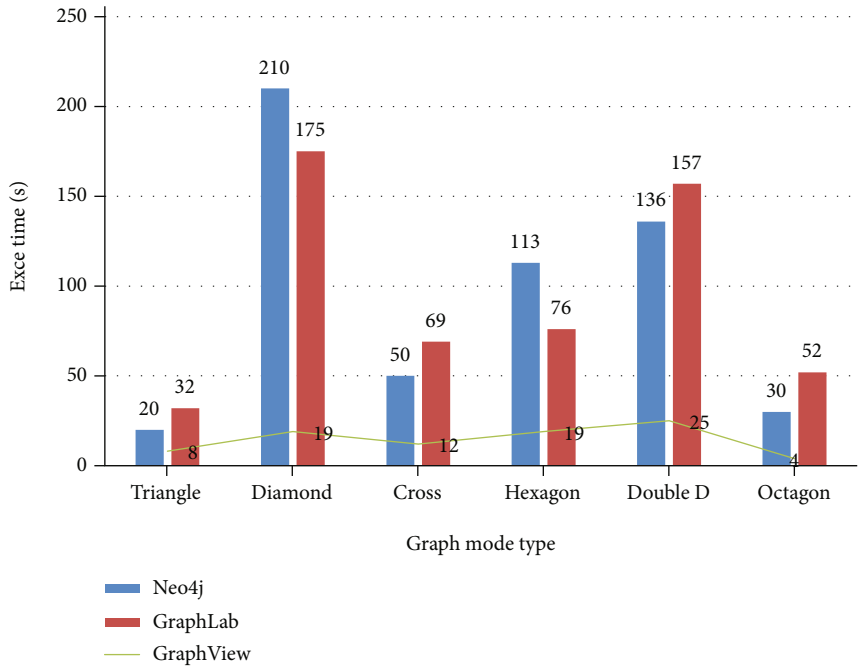
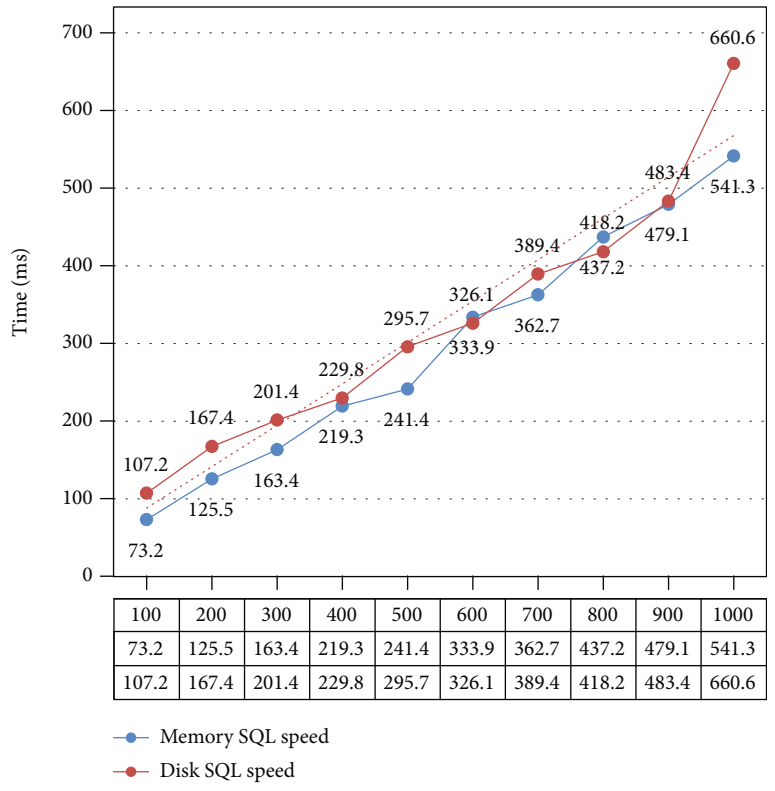


Figure 7: System performance comparison chart.



100	200	300	400	500	600	700	800	900	1000
73.2	125.5	163.4	219.3	241.4	333.9	362.7	437.2	479.1	541.3
107.2	167.4	201.4	229.8	295.7	326.1	389.4	418.2	483.4	660.6

Figure 8: Concurrent query graph for key value.

graph computing platform that supports parallel processing. It uses a node-oriented iterative computing model. It is not a graph database itself and is not responsible for storing and managing data. Graph data used in calculations is directly read from disk. The method in this paper has more prominent performance advantages when processing graph queries.

Since the algorithm designed in this article will automatically make the structure of the memory dimension table the

same as that of the dimension table in the database, there is no need to worry about the problem of data out of synchronization, and the maximum number of connections to the database is set to 1000. Now simulate multiple network users to query the value in the "ID" attribute at the same time. Because of the limitation of the experimental equipment and the experimental environment, in order to reduce the experimental error as much as possible, the number of concurrent tests for each item will be tested 50 times to find the average value. The result is shown in Figure 8. From the figure, it can be seen that the memory SQL of the query time consumption basically increases linearly with the number of concurrent user accesses; the time consumption of database SQL queries will reach more than twice the time consumption of memory SQL queries after the number of concurrent user accesses reaches 120 or more. It is proved that the algorithm designed in this paper has achieved the expected effect on the concurrent query optimization of the key value in the relational database dimension table.

5. Conclusions

This article optimizes database queries based on artificial intelligence and edge computing and uses query optimization algorithms to optimize database performance and improve query efficiency. This paper proposes a database query optimization solution for the problems in the database query data processing process, which reduces the network transmission cost to reduce the cost in the query process and optimize the data query response time. This article introduces the mechanism for implementing this optimization technique in detail and focuses on analyzing the performance advantages of this technique on storage sites. Finally, we tested and evaluated different data sets through a series of detailed experiments. The implementation results show that the query optimization algorithm shown in this article can better reflect the effectiveness and efficiency by comparing with other query optimization methods and has obvious advantages. However, this article has not invested too much research on logical query optimization, and further research is still needed.

Data Availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Acknowledgments

The achievement is supported by the 13th Five-Year Plan (2018) of Social Science Fund of Jiangxi Province.

References

[1] G. Pigozzi, A. Tsoukias, and P. Viappiani, "Preferences in artificial intelligence," *Annals of Mathematics and Artificial Intelligence*, vol. 77, no. 3-4, article 9475, pp. 361–401, 2016.

[2] L. Fabisiak, "Web service usability analysis based on user preferences," *Journal of Organizational and End User Computing*, vol. 30, no. 4, pp. 1–13, 2018.

[3] K. Li, X. Zhou, D. Z. Wang, C. Grant, A. Dobra, and C. Dudley, "In-database batch and query-time inference over probabilistic graphical models using UDA–GIST," *VLDB Journal*, vol. 26, no. 2, pp. 177–201, 2017.

[4] A. Moreau, O. Pivert, and G. Smits, "Linguistically characterizing clusters of database query answers," *Fuzzy Sets and Systems*, vol. 366, pp. 18–33, 2019.

[5] D. Hassabis, D. Kumaran, C. Summerfield, and M. Botvinick, "Neuroscience-inspired artificial intelligence," *Neuron*, vol. 95, no. 2, pp. 245–258, 2017.

[6] H. Lu, Y. Li, M. Chen, H. Kim, and S. Serikawa, "Brain intelligence: go beyond artificial intelligence," *Mobile Networks and Applications*, vol. 23, no. 2, pp. 368–375, 2018.

[7] Q. Wang and P. Lu, "Research on application of artificial intelligence in computer network technology," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 33, no. 5, article 1959015, 2019.

[8] S. Jha and E. J. Topol, "Adapting to artificial intelligence: radiologists and pathologists as information specialists," *JAMA*, vol. 316, no. 22, pp. 2353-2354, 2016.

[9] L. D. Raedt, K. Kersting, S. Natarajan, and D. Poole, "Statistical relational artificial intelligence: logic, probability, and computation," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 10, no. 2, pp. 1–189, 2016.

[10] F. Zhong, J. Xing, X. Li et al., "Artificial intelligence in drug design," *Science China Life Sciences*, vol. 61, no. 10, pp. 1191–1204, 2018.

[11] N. J. Nilsson, "Artificial intelligence: a modern approach," *Applied Mechanics & Materials*, vol. 263, no. 5, pp. 2829–2833, 2003.

[12] J. Davies, "Program good ethics into artificial intelligence," *Nature*, vol. 538, no. 7625, pp. 291–291, 2016.

[13] S. Makridakis, "The forthcoming artificial intelligence (AI) revolution: its impact on society and firms," *Futures*, vol. 90, pp. 46–60, 2017.

[14] Z. Lv, D. Chen, R. Lou, and Q. Wang, "Intelligent edge computing based on machine learning for smart city," *Future Generation Computer Systems*, vol. 115, pp. 90–99, 2021.

[15] I. E. Gordon, L. S. Rothman, C. Hill et al., "The HITRAN2016 molecular spectroscopic database," *Journal of Quantitative Spectroscopy & Radiative Transfer*, vol. 130, no. 11, pp. 4–50, 2017.

[16] S. H. Yoon, S. M. Ha, S. Kwon et al., "Introducing EzBioCloud: a taxonomically united database of 16S rRNA gene sequences and whole-genome assemblies," *International Journal of Systematic and Evolutionary Microbiology*, vol. 67, no. 5, pp. 1613–1617, 2017.

[17] J. Klimešová, J. Danihelka, J. Chrtek, F. de Bello, and T. Herben, "CLO-PLA: a database of clonal and bud-bank traits of the central European flora," *Ecology*, vol. 98, no. 4, pp. 1179–1179, 2017.

[18] K. Corder, S. J. Sharp, A. J. Atkin et al., "Age-related patterns of vigorous-intensity physical activity in youth: the International

Children's Accelerometry Database," *Preventive Medicine Reports*, vol. 4, no. C, pp. 17–22, 2016.

[19] M. Gleeson, A. Hannigan, R. Jamali et al., "Using electronic medical records to determine prevalence and treatment of mental disorders in primary care: a database study," *Journal of Chromatography A*, vol. 33, no. 1, pp. 3–12, 2016.

[20] G. Wernet, C. Bauer, B. Steubing, J. Reinhard, E. Moreno-Ruiz, and B. Weidema, "The ecoinvent database version 3 (part I): overview and methodology," *The International Journal of Life Cycle Assessment*, vol. 21, no. 9, pp. 1218–1230, 2016.

[21] J. Xiao, K. A. Ehinger, J. Hays, A. Torralba, and A. Oliva, "SUN database: exploring a large collection of scene categories," *International Journal of Computer Vision*, vol. 119, no. 1, pp. 3–22, 2016.

[22] J. Gartner, D. B. Larson, and G. D. Allen, "Religious commitment and mental health: a review of the empirical literature," *Journal of Psychology & Theology*, vol. 19, no. 1, pp. 6–25, 2018.

[23] A. R. Bradlow and D. B. Pisoni, "Recognition of spoken words by native and non-native listeners: talker-, listener-, and item-related factors," *Journal of the Acoustical Society of America*, vol. 106, no. 4, pp. 2074–2085, 1999.

[24] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[25] J. H. Thrall, X. Li, Q. Li et al., "Artificial intelligence and machine learning in radiology: opportunities, challenges, pitfalls, and criteria for success," *Journal of the American College of Radiology*, vol. 15, no. 3, pp. 504–508, 2018.

[26] Z. Xiong, Y. Zhang, D. Niyato, P. Wang, and Z. Han, "When mobile blockchain meets edge computing," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 33–39, 2018.

[27] S. Nastic, T. Rausch, O. Scekic et al., "A serverless real-time data analytics platform for edge computing," *IEEE Internet Computing*, vol. 21, no. 4, pp. 64–71, 2017.

[28] Z. Li, W. M. Wang, G. Liu, L. Liu, J. He, and G. Q. Huang, "Toward open manufacturing," *Industrial Management & Data Systems*, vol. 118, no. 1, pp. 303–320, 2018.

[29] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[30] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: partial computation offloading using dynamic voltage scaling," *IEEE Transactions on Communications*, vol. 64, no. 10, pp. 1–4282, 2016.