An Optimization for Distributed Database Multi-join Query Based on Improved Genetic Algorithm

Shiwu Ye and Yunjian Peng

ABSTRACT

The paper considers the optimization of the multi-join query process of distributed database, which is mainly based on the computer time or resource cost and search strategy of querying database. To model the cost, the multi-join query is divided from a full-join query into some semi-join queries steps to reduce the communication cost and then a new cost model is presented. As for the search strategy, by comparing the advantages and disadvantages of dynamic programming algorithm (DPA), simulated annealing algorithm (SAA) and iterative correction algorithm (ICA), genetic algorithm (GA) is improved to optimize the query process because of its better search performance. To implement such an optimization, the paper deals with GA’s process of coding, selection, crossover and mutation carefully in novel way so as to improve the accuracy and speed of searching for optimal values of querying a distributed data base. Finally, the simulation results show that the optimization of cost model and search strategy achieves the expected effect and it can improve the accuracy and speed of the distributed database query.

INTRODUCTION

Compared with other common query, a multi-join query in distributed database is relatively complex because it has to take into account not only the case that different tables are distributed at different sites, but also the situation where the same
table is distributed at different sites. Since the data transmission among different sites is highly frequent, the delay of transmission will have huge impact on the accuracy and immediacy of the distributed database multi-join query [1].

Multi-join query has always been basic operation of a distributed database query and many methods have been applied to distributed database query optimization. Among all the methods in distributed database query optimization, dynamic programming is the most common one [2]. It uses exhaustive search methods to ensure that the best result can be found. When the number of relationship in a multi-join query is small, the time for executing query plan is acceptable. However, when the number of relationships is greater than a certain number, it will be too long to meet the needs of cloud computing technology. In the article [3], an iterative solution of database query using greedy algorithm is proposed. Although this algorithm shortens the query time and improves the query efficiency based on the dynamic programming algorithm, it has no significant effect. Heuristic algorithms such as simulated annealing, iterative correction, and two-stage optimization are also applied to the query optimization program. They try to find an optimal solution by making a heuristic rule that produces a feasible solution. Although these methods can reduce the complexity of query optimization, they are still local optimal methods and may not find the global solution. The genetic algorithm initializes a certain number of populations by simulating the way of biological evolution, and then produces the optimal individual through the operation such as selection, crossover and mutation of individuals in the population. This algorithm has been applied to the distributed database query optimization program [4], but the optimization performance of traditional genetic algorithm is not very good. Sometimes it may leads to the situation of local convergence so that the global optimal solution can not be obtained. In this paper, the genetic algorithm is improved and then applied to solve the problem of distributed database query optimization. Finally, we made several experiments to verify the optimization performance of the algorithm.

The Query optimization consists of three parts: search space, cost model, and search strategy. Search space is a collection of query plans that are equivalent to the input query, it can be obtained by the equivalence conversion rule of relational algebra. Cost model is used to calculate the cost of each query plan, which can be considered as the entire time required for the query. Search strategy selects the optimal query plan in the search space according to the cost model. This article aims mainly at the optimization of cost model and search strategy.

**COST MODEL OPTIMIZATION**

In the distributed database cost model, the formula for calculating the total time required for a query is:
Total cpu I/O trans

T_{total} = T_{cpu} + T_{I/O} + T_{trans} \quad (1)

Where T_{total} is the total time of the query execution, T_{cpu} is the total time of the cpu execution, T_{I/O} is the total time of the I/O execution, T_{trans} is the total time of the data transmission.

During the execution of the query, the time to read and write on disk I/O takes much longer than caching data, so T_{cpu} can almost be ignored in the estimation of the total execution time of the query. Meanwhile, as the data transmission in distributed database system among different sites may encounter situation like low bandwidth, poor communication and so on, it usually has T_{trans} : T_{I/O} = 20 : 1 [5], the total time for query execution mainly depends on the time it takes for T_{trans}.

In the process of multi-join query in distributed database, the full-join operation spends a lot of time. A lot of data will be transmitted through network and the time spent of different query plans have significant differences. Therefore, this paper uses the semi-join algorithm instead of full-join algorithm to reduce the cost of data transmission among sites. The steps of semi-join algorithm can be described in Figure 1.

\[ T_{R,S} = 2T_{init} + V_{tran}(\text{Length}(B)\text{Card}(S) + \text{Length}(R')\text{Card}(R')) \quad (2) \]

Where T_{init} is the time required to start a data transmission among sites, V_{tran} is the time it takes to transmit one byte in the network, \text{Length}(B) is the bytes attribute of B, \text{Card}(S) is the number of tuples relation of S.

Figure 1. Steps for semi-join algorithm.
The total cost of the query is represented by $T_{\text{total}}$. We can assume that $n$ sites are involved in the semi-join operation, the $i$th parent node can be represent as $t_i$ and the cost of generating this node is $\text{cost}(t_i)$. The total cost of the query plan is the sum of the cost for generating all the parent nodes, it can be expressed as follows:

$$T_{\text{total}} = \sum_{i=1}^{n-1} \text{cost}(t_i), i = 1, 2, \ldots, n-1$$ (3)

Then from (2)(3), we can summarize that the total query time is:

$$T_{\text{total}} = 2(n-1)T_{\text{init}} + \sum_{i=1}^{n-1} V_i \text{Length}_i(\text{Card}(S_i) + k_i \text{Card}(R_i)), i = 1, 2, \ldots, n-1$$ (4)

SEARCH STRATEGY OPTIMIZATION

Genetic algorithm abandons the traditional search method, which regards the possible solutions in the problem domain as an individual in the population and encodes each individual into a certain string, and then repeats the operation of the population such as selection, crossover and mutation. The population will get better according to the survival principle of fittest.

Genetic Coding

The task of genetic coding is to express the individual in the population as a string that can participate in the operation. The distributed database multi-join query plan can be represented by a query tree. The child node represents the site where the relationship in the multi-join query is located. The parent node represents the semi-join operation of the left child node and the right child node. And then we can use the suffix expression method to convert the query tree to the certain string to represent individual in the population.

Population Initialization

We can randomly generate a number of individuals represented by a string whose length is $n$. Each string contains $n/2$ parent nodes and $n-n/2$ child nodes. Then we can use a shuffle algorithm to disorder the string representing to produce a new individual. However, the new individual can not guarantee that the string can be expressed as a query tree. We should convert the old string to a new one according to the suffix expression method to represent the corresponding query tree.
Fitness Function

Genetic algorithm uses fitness function to measure the fitness of individuals in the population and then choose relatively better individuals to be inherited to the next generation. The probability of being inherited to the next generation of individuals with higher fitness is larger than those with lower fitness.

Individuals with higher cost of a multi-join query in the distributed database should have lower fitness. The fitness function can be expressed as follows:

\[ f = \frac{1}{(2(n-1))^{T_{init}}} + \sum_{i=1}^{n-1} V_{Length}(\text{Card}(S_i) + k_i\text{Card}(R_i)), i = 1, 2, \ldots, n-1 \]  

Selection

After crossover or mutation, individuals with higher fitness have more opportunities to be inherited to the next generation. Traditional roulette algorithm uses fitness function to determine whether the individual can be inherited to the next generation. However, this method may just find local optimal solution instead of global optimal solution. It can be optimized by using the elite strategy. That means we can replace the individual which has lowest fitness with the one which has highest fitness and ensure that the excellent gene will be inherited to the next generation of individuals.

Crossover and Mutation

Mutation can help produce new individual with different genes. Mutation rate reflects the differences between children and parents. The traditional crossover and mutation operation allows all individuals to crossover and mutate at the same rate, so the crossover rate and mutation rate are single and it can not make full use of the characteristic of each individual gene. Individuals with higher fitness should have a smaller crossover rate and mutation rate so that good genes can be inherited to the next generation. On the other hand, individuals with lower fitness should have a bigger crossover rate and mutation rate so that they will have the opportunity to evolve into better individuals. The FCM (Fuzzy C-means) algorithm divide them into three categories according to the fitness of each of them: excellent, medium and poor. Individuals in the excellent group have the lowest crossover rate and mutation rate, individuals in the poor group have the highest crossover rate and mutation rate.
EXPERIMENTAL RESULTS AND ANALYSIS

In this paper, we have made experiments to verify the performance of multi-join query optimization based on improved genetic algorithm. On the parameter of traditional genetic algorithm and improved genetic algorithm, the selected population size is 100. The traditional genetic algorithm has a single crossover rate of 0.7 and a mutation rate of 0.15. The crossover rate of the improved genetic algorithm is 0.6, 0.7 and 0.8, respectively, and the mutation rate is 0.1, 0.15 and 0.2. The terminate condition is that if the optimal solution can not be found within 200 consecutive generations, the optimal solution is obtained and the experiment can be stopped. The experimental results are shown in Figure 2.

Figure 2. (a) shows the comparison of query accuracy, (b) shows the comparison of query time.

Figure 2 shows that the improved genetic algorithm is more accurate than the traditional genetic algorithm and the query time is shorter.

CONCLUSIONS

In this paper, a distributed database multi-join query algorithm based on genetic algorithm is proposed. The cost model is optimized by using the semi-join instead of the full-join operation and a new cost model is given. The traditional genetic algorithm is improved on search strategy. Besides, FCM clustering algorithm is used to classify the individual population. Setting appropriate crossover rates and mutation rate for each category can take full advantage of the characteristics of individuals. The simulation results show that in the distributed database, the improved genetic algorithm is more accurate and the query speed is faster. But this paper ignores the cost of CPU and I/O which may influence the
multi-join query of distributed database to a certain extent, we will address this issue as a future work.

ACKNOWLEDGMENT

Supported by National Natural Science Foundation of China (61573154, 60904032), Science & technology planning project of Guangdong Province (2015A010106003).

REFERENCES