ATLANTIS PRESS

Advances in Computer Science Research, volume 93

2nd International Conference on Mathematics, Modeling and Simulation Technologies and Applications (MMSTA 2019)

A Database Optimization Strategy for Massive Data Based Information System

Qianglai Xie*, Wei Yang and Leiyue Yao
Jiangxi University of Technology, China
*Corresponding author

***Abstract*—With the rapid arrival of the information explosion era, the amount of data stored in a single table of an enterprise information management system is getting more and more common. Therefore, efficiency optimization of SQL based on the massive data is an essential task to break the bottleneck in current enterprise information management systems. This paper proposes single table optimization methods for massive database. These methods including: physical optimization, index optimization and query statement optimization. Each optimization method is integrated into the database design process, and a set of optimization programs for the massive database design process is proposed. In the test environment, the size of the databases file is over 200G and single table records is over 80 million, the proposed optimization methods break the bottleneck of query efficiency and improve the database query efficiency of tens of times.**

Keywords—database optimization; massive data optimization; database structure optimization; data division

I. INTRODUCTION

With the rapid arrival of the information explosion era, massive data storage and management is an important issue in current subject areas. The database storage of large-scale information management systems is usually hundreds of gigabytes in size, the data records of a single table may often reach 10-millions level, and the number of records will continue to grow over time [1]. This not only affects the efficiency of the database, while increasing the difficulty of database maintenance. In order to improve the query and management of large data tables efficiency, the traditional relational database to introduce physical optimization table partition optimization, index optimization, query statement optimization [2].

The physical structure of the database is the basis of the entire database storage, and a persistent storage strategy must be determined during the physical design phase[3]. Liu et al.[4] mentioned that the physical structure design is performed after the logical structure design. The key goal of physical structure optimization is to ensure that the method or operation of database storage and access to data tables is highly efficient based on logical design. Thus, all applications based on relational databases can maximize the data access speed of the entire system. Yang et al.[5] mentioned that the physical structure of a reasonably large databases can be partitions, indexes, materialized views, multidimensional clusters, or any combination of them. Partitioning technology is an important support technology for the physical design of the entire database, and the method of range partitioning is mainly used in the SQL Sever databases design [6][7].

The index of the database table structure is an important factor affecting the database performance. Establishing a reasonable index on the basis of physical optimization will directly affect the efficiency of the entire database access [8,9]. The pages of SQL server data storage include two types: data pages and index pages. By default, data exists in the data pages, but as long as the index data exists, the indexes will be automatically stored in the index page [10,11]. The index types include: clustered and non-clustered indexes. The fundamental difference between the clustered index and non-clustered index is whether the ordering of the table records is consistent with the order of the indexes [12]. Zhang et al. [13] proposed that indexing is an effective tool and method for efficient database retrieval, and provides a reasonable set of ideas and methods for indexing. Yang et al. [14] analyzed the performance difference of using the binary tree to analyze the index structure of the database.

SQL is the only way to operate data in the database, and it is also the only way that data exchange between the applications system and the database [15]. The data operations of the application programs on the database are finally reflected in the form of SQL statements. The main work tasks include receiving SQL, dividing memory, parsing SQL, generating SQL execution plans, and extracting data to occupy the database [16]. As the number of data increases, IO becomes a major bottleneck in database performance. Therefore, the efficiency of SQL statements execution is improved to reduce the number of IO scans, reduce CPU usage, and reduce memory usage [17]. Fan et al. [18]proposed that query optimization is based on the optimization of relational algebraic expressions, and that a certain query can be implemented using a variety of equivalent relational algebraic expressions.

This article uses SQL Server as the database platform to achieve the relevant optimization process. Section 2 mainly introduces the optimization process of physical optimization, index optimization, and query optimization of massive databases. Section 3 mainly outlines our experiments and results. Section 4 summarizes the paper and discusses future work.

II. EASE OF USE

The optimization of the massive database is mainly divided into three steps. First of all, to achieve the best results in the data storage in the physical design phase of the database, the storage of the database is first partitioned and stored in the table. Then, based on the completion of table partitioning, the most reasonable index is established by the index selection algorithm. Finally, the optimization of the query is performed through relational algebraic expressions according to different business table structures. The optimization process is shown in Figure 1.

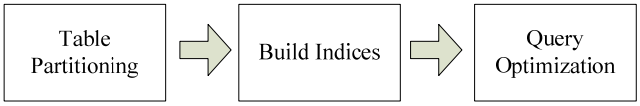

FIGURE I. THE GENERAL BLOCK OF OPTIMIZING PROCESS.

A. Table Partitioning

Table partitioning refers to partitioning a large table physically into multiple small table storages in a database, but in logically, it is still a table partitioning method. Partitioning massive data table and storing them on different physical disk database files improve the database I/O operation efficiency. Then, based on the cache update mechanism [19], use appropriate partition fields and create a database partition file number follow the requirements of the business, SQL Server table partition completed by the following four steps, the specific structure can refer Figure 2:

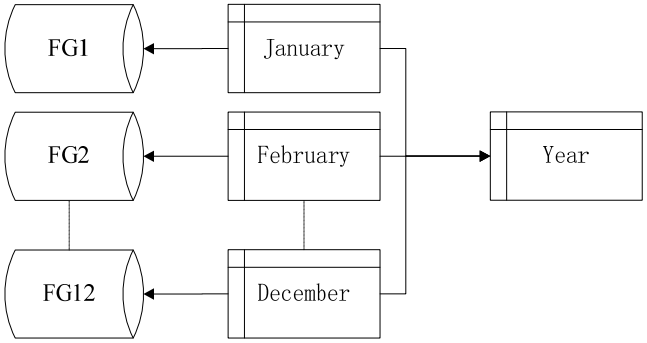

FIGURE II. A EXAMPLE OF PARTITION FILE USED FOR TABLE DIVISION.

B. Index

SQL Server indexing has two purposes: to maintain the uniqueness of the indexed columns and to provide a quick query of data tables. Large database systems contain both clustered and non-clustered indexes. Non-clustered indexed data tables store data in a heap structure. The data is added to the table's tail. Clustered index table data is stored in the order of the index key, and a data table only allows the existence of a clustered index [20]. Table index is very like book index. It provides us a link between logical data and physical data, which can let the data be queried more quickly. Table index contains “Key” created by columns from table or view, these Keys are stored in a BTree structure, which accelerate the speed of data query [21]. Table 1. lists some strategies of index creation.

TABLE I. 8 MOST COMMONLY USED RULES FOR INDEX CREATION

Rule	Field action description	Aggregated index	Non-aggregated index
1	Field as grouping sort	Yes	Yes
2	The data in the query range	Yes	No
3	Less different data values	No	No
4	A few different data values	Yes	No
5	Most of the different data values	No	Yes
6	Data modification frequently	No	Yes
7	Foreign key	Yes	Yes
8	Primary key	Yes	Yes

The above simple list indicates when to use a clustered index or a non-clustered index. However, in the real application situation, these rules are easily overlooked or cannot be based on the actual situation. Based on the problems encountered in the practical application, the following three errors in the establishment of the index can be summarized as following:

C. Query Statement Optimization

After creating an efficient index, the next step is to optimize the program's query statement. To further improve the performance of the entire database and access speed, analyzing the equivalent transformation algebra expression expressions and query costs is essential. For a given SQL query correspondence relations with relational algebraic expressions, research and analysis of SQL query optimization should be based on equivalent algebraic equivalent transformation rules [22].

D. Database Optimization Process

As shown in Figure 3.the database optimization process can be generally included in three steps:

Step1. According to the situation of database optimization, buffer strategy is used to create partition files, such as, FG1, FG2….FG10, FG11, FG12. Then, to reduce the IO load of the different table partition of the same table, the logical tables of different table partitions were stored into each corresponding partition files.

Step2. After table partitions have been created, the following step is choosing certain columns for index creation based on data operation and records number.

Step3. 1~10 represent the equivalent transformation rules of 10 kinds of relational algebras respectively, and generate the optimal SQL query statement through the equivalent transformation rules of relational algebras.

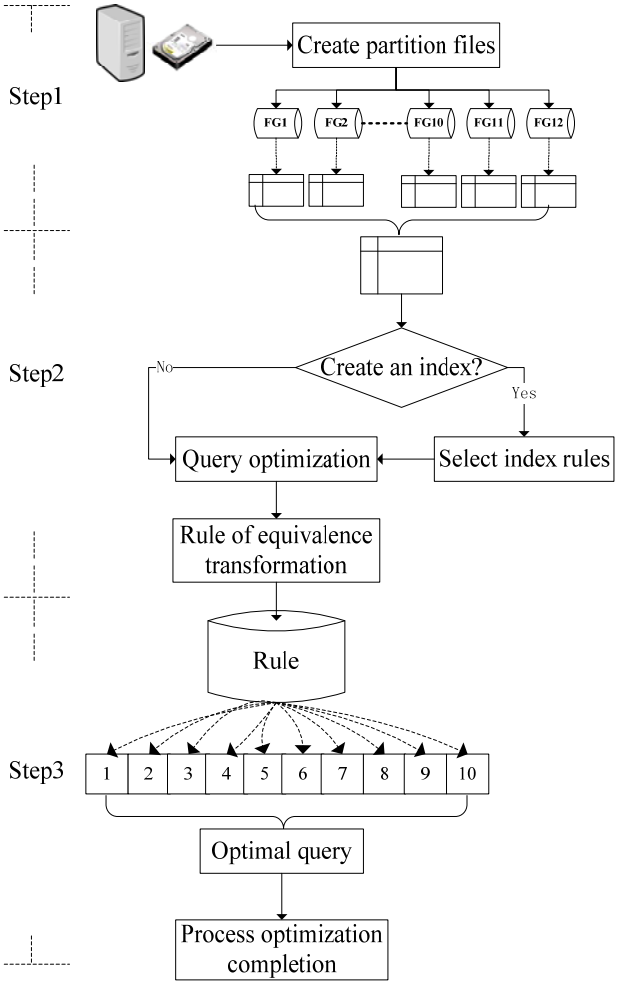

FIGURE III. THE FLOW CHART OF OPTIMIZING PROCESS USED IN THE PROPOSED METHOD.

III. EXPERIMENTAL EVALUATION

We ran our experiments using Intel E5-2650 processor, 8 GB RAM and 1T SCSI hard disk. The operating system and database system our method experimented are Windows Server 2012R2 Datacenter and Microsoft SQL Server 2012 Enterprise Core Edition respectively. Application system, Nanchang Railway Bureau freight operation inspection system, is the operational system for the entire tenth cargo inspection station of Nanchang Railway Bureau. The annual database file size of the system is about 200G, and the maximum number of individual confirmed information tables reaches more than 80 million, among which the report information table structure is shown as Table 2.

TABLE II. ENSURE INFORMATION TABLE STRUCTURE

serial number	Column name	type	meaning
1	ID	bigint	Record ID
2	车次	nvarchar(10)	Freight train trips
3	车号	nvarchar(10)	Number of each carriage
4	载重	decimal(10,2)	The actual weight of each vehicle
5	所在	nvarchar(50)	The cargo inspection station
6	解体站	nvarchar(10)	Arrival of freight train
7	品名	nvarchar(20)	Each car carrying items
8	采集时间	datetime	Time to collect current data
9	操作人 id	bigint	Relationship User Information Table
10	发车时间	datetime	Freight train departure time
11	到达时间	datetime	Freight train arrival time
…	……	……	……

There are sereval columns in ensure information table. Many of them are foreign keys or frequently queried by users. To ease the complexity of SQL, a view named “view_确报” was created for data searching.

During the entire optimization process, the test query executed by the SQL Server database query analyzer is shown in Figure 4:

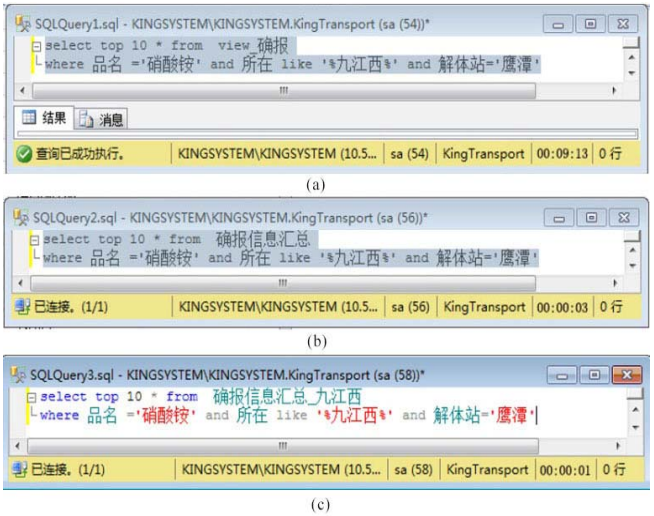

FIGURE IV. SQL QUERY EFFICIENT TEST AND COMPARISOND.

In Figure 4 (a), it cost 9 minutes and 13 seconds to get query records from “view_确报” which has no optimization at all. As show in Figure 4, it can be inferred that the out joins

among tables cost a lot. To avoid out joins costs and speed up the efficiency of data query, the operating column should be set as a partition field, and the data of each month is used as a partition. A total of 12 partition files FG1~ FG12. These 12 partition files are stored in 4 disk enclosures, and each file is about 15G in size. After performing the above partition method, the data storage after the specific partition is shown as Table 3.

TABLE III. PARTITION TABLE DATA RECORDS

Partition number	The number of records in the partition
1	9,200,678
2	7,100,570
3	9,900,621
4	8,710,110
5	8,200,678
6	8,120,677
7	8,224,618
8	7,910,124
9	8,570,179
10	8,210,128
11	8,302,316
12	9,503,217

Query 1, 5, 10, 15, 20, and 30 pieces of data in non-partitioned and partitioned environments respectively. The query record quantity is D, the total data record is G. as follows:

$$X = D/(G \cdot S^{-1}) \quad (1)$$

According to the ratio of the amount of data for each query and the corresponding query time, Figure 5 can be obtained.

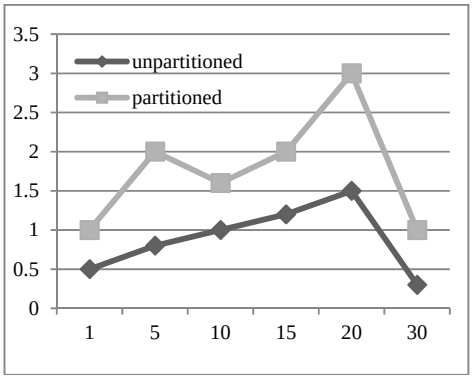

FIGURE V. THE TOTAL AMOUNT OF DATA QUERIED AFTER TABLE PARTITIONING IS COMPARED WITH THE CORRESPONDING QUERY TIME.

Figure 5 shows that the query efficiency per unit time after partitioning the table has been significantly improved. However, the larger the query data volume within a certain range, the higher the query efficiency after partitioning. When the amount of data reaches its peak, the sudden drop was followed in query efficiency. This phenomenon is mainly caused by the limitation of hardware conditions. Thus, the layout of the partition file should be displayed in a separate IO operating disk.

In the second step, create indexes on table "确报" after the partition. Follow the eight rules of establishing an efficient index, the indexes of the 确报 table are list in Table 4.

TABLE IV. 确报 INFORMATION INDEXING

Column name	Index name	type	order
作业时间	Index_确保_ 作业时间	cluster	Descend
车次	Index_确保_ 车次	Non-cluster	Ascend
车号	Index_确保_车号	Non-cluster	Ascend
发站、到站	Index_确保_发站_到站	Non-cluster	Ascend
所在	Index_确保_所在	Non-cluster	Ascend
解体站	Index_确保_解体站	Non-cluster	Ascend
品名	Index_确保_ 品名	Non-cluster	Ascend

The same query conditions are constructed based on the index field, that is, the number of query records is consistent with the partitioning experiment. The total amount of data for each query and the corresponding query time can be obtained as shown in Figure 6.

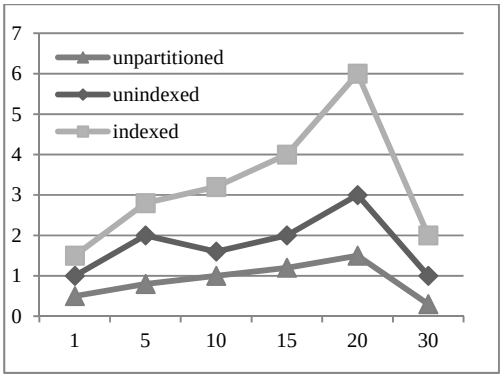

FIGURE VI. COMPARISON OF QUERY DATA AND CORRESPONDING QUERY TIME AFTER INDEXING.

According to the analysis of Figure 6, the query efficiency per unit time after indexing based on the table partitioning has been significantly improved. However, there is also the problem of query data volume peak.

In the last step, through the equivalent transformation algebra of the rules of the table partitioning and indexing of the test query corresponding optimization, the optimized results are recorded in Figure 7.

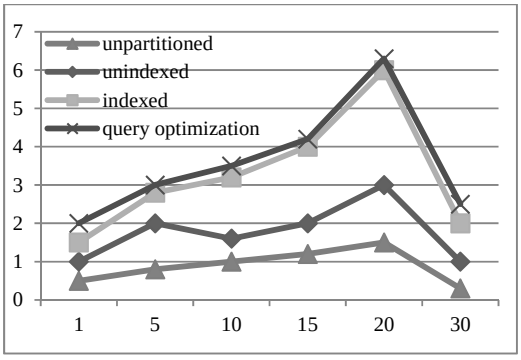

FIGURE VII. COMPARISON OF QUERY DATA AND QUERY TIME AFTER QUERY OPTIMIZATION.

As shown in Figure 7, the query efficiency per unit time is further improved by optimizing the query based on table partitioning and indexing.

IV. CONCLUSIONS

In this paper, we proposed an optimization process for the entire design phase of the database. First, according to physical hardware resources, the massive data tables are used to establish the corresponding table partitions. Then, a reasonable indexing scheme is established on the basis of the table partitions. Finally, an optimal SQL query is generated by the equivalent transformation algebra of the relational algebra. Experiments show that after optimization, there is no delay in data interaction operations, satisfying the current application requirements and user experience in the management information system of a 200G-sized database file and a single table expressed to tens millions records.

The inadequacy of the optimization method proposed in this paper is that the query efficiency will decline if the amount of data in the query reaches its limitation. In the future work, the deeper research will be continued on the combination of physical hardware and network environment for optimization of massive data.

ACKNOWLEDGMENT

This paper was supported by the sience and technology research project of Jiangxi Education Department GJ180978, natural science general project of Jiangxi University of Technology JY1718&16ZRYB08 and the science and technology research project of Jiang Xi science and technology agency under Grant 20171BBE50060.

REFERENCES

[1] Qianqian Zheng, "database optimization based on automatic screening of large data storage areas", Software Guide, vol. 10, pp.165-167, 2016.

[2] R. Akerkar, "Big Data Computing", CRC Press Inc, Boca Raton, America, 2013.

[3] Xiaoya Xu, Junfang Li "database design for performance optimization analysis of SQL Server database", China Computer & Communication, vol.2, pp.177-179, 2017. http://dx.doi.org/10.3969/j.issn.1003-9767.2017.02.064

[4] Chen Yang, "research on database physics design and optimization technology", Electronics World, vol. 19, pp. 178-179, 2013. http://dx.doi.org/10.3969/j.issn.1003-0522.2013.19.141

[5] EMC Education Services, "Data Science and Big Data Analytics",Posts &Telecom Press, Beijing , China, 2016.

[6] Wei Wang, "enterprise database design of information construction", Digital Technology and Application, vol. 2, pp. 172-173, 2015.

[7] Lei Shi, "research on index algorithm based on memory database", Information Technology,vol.11,pp.139-142, 2016. http://dx.doi.org/10.13274/j.cnki.hdzj.2016.11.035

[8] Hong Chen, "distributed SQL database index design and practice based on Hadoop", Ship Electronic Engineering, vol. 4, pp.73-74, 2018. http://dx.doi.org/10.3969/j.issn.1672-9730.2018.04.020

[9] Shan Wang, Shixuan Sa, "introduction on database system", higher education press, Beijing , China, 2014.

[10] T. White, "Hadoop:the Definitive Guide Storage and Analysis At Internet Scale", Tsinghua University press, Beijing , China, 2017.

[11] Jinping Meng, Yidong Li, "on SQL adjustment and optimization on Oracle database", Digital Technology and Application, vol. 3, pp.217-219, 2018.

[12] Hanming Tang, "deeper but easier MySQL: database development, optimization and management maintenance", Posts &Telecom Press, Beijing , China, 2014.

[13] X. Y. Zhang and J. P. Qiu, "The Research of Database Index Compilation Progress and Trends at Home and Abroad", Library Journal, vol. 3,pp.60-67, 2016.

[14] Zexue Yang, "research on a hybrid index structure in spatial databases",Computer Engineering and Applications, vol.20, pp.20-23+165, 2017. http://dx.doi.org/10.3778/j.issn.1002-8331.1707-0390

[15] Zhibo Chen, "principles and applications on database", Posts &Telecom Press, Beijing , China, 2017.

[16] Shuqing Yan, "high performance SQL optimization essence and case analysis", Publishing House of Electronics Industry , Beijing , China, 2017.

[17] Gongchang Zhang, "the essence of MySQL technology -- architecture, advanced features, performance optimization and cluster practice", Tsinghua University press, Beijing , China, 2015.

[18] Jianbo Fan, Liangxu Liu, "database technology and design", XIDIAN University, Xi 'an, China, 2016.

[19] J. Wang, "Partitioned table used in VLDB optimization",Journal of Dalian Dalian Polytechnic University, vol. 6, pp. 465-468, 2013.

[20] F. W. Mu, "Database Indexing Technology Overview",Computer Knowledge and Technology , vol. 25, pp. 9-11+13, 2017.

[21] T. Lahdenmaki, M. Leach, "Realtional Database Index Design and the Optimizers", Press of Electronics Industry, Beijing , China, 2015.

[22] H. X. Li, "The art of database query optimizer: Principle Analysis and SQL performance optimization", China Machine Press, Beijing , China, 2014.