The Design and Optimization of Database

Guo Feng
Rizhao Polytechnic, Rizhao, Shandong, 276826, China
66077576@qq.com

Abstract. Database technology is one of the fastest technical fields in the process of computer development. Its popularity has affected all aspects of people's lives, and it is the core of the management information system. This paper discusses and studies all the important aspects of database system optimization, including the importance of the third normal form (3NF) in table design, the design of extended database tables, the design of database table storage problems, and so on. At the same time, various techniques for optimizing database are put forward.

1. Introduction
In recent years, with the popularization of computer and the development of network, database technology develops rapidly.

Database technology has become one of the core technologies in computer field. The use of computers for data management can be seen everywhere. Database is a warehouse for storing data. It is used in all aspects of our life, such as the management system of government agencies, enterprises and institutions, office OA system, the civil aviation data system, the student management system, the university teaching management system, the student management system and so on. It can be said that the performance of the database will directly affect the performance of the computer management system. Therefore, in the process of actual use of database, it is inevitable to face the problem of database optimization. The ultimate goal of database system optimization is to improve the performance of the server where the database system is located. These performance aspects include: server disk access read / write bottlenecks, server CPU utilization, and server system resource footprint.

In the process of using database systems such as MySQL, oracle and db2 for a long time, and according to many years' practical experience, the author studies and expounds the optimization of database system from the angles of basic data table design, extended data table design and data table storage, etc. This paper puts forward some ideas for optimizing the database, and analyzes on the ways to realize the scheme.

2. Design database tables follow 3NF principles
In MIS, the design of the basic data table follows the specifications of the third normal form (3NF). The so-called third normal form refers to the field of the non-primary key attribute in the datasheet, which can only depend on the field where the primary key attribute is located. Database tables designed according to normal form rules have many advantages: first, it can save disk space by eliminating redundant data from datasheets, which is the most basic feature and the fundamental reason for establishing normal form rules. The design of database tables based on normal form rules has the following advantages: eliminating redundant data from data tables and saving disk space is the most basic feature. Second, it can make the data organization more reasonable, the entity integrity
constraints of primary keys and easier to maintain data, and also relatively easy to copy and update data between multiple tables. Third, in the connection query or merge query, it is not easy to omit the data, and it is not easy to repeat it. Fourth, the performance of the transaction operation is better than the data table of the complex field. Fifth, in the later maintenance of table structure, the workload is small.

Among the field constraints, primary key, foreign key and index are very important ones. The author explains the importance of primary keys, foreign keys, and indexes to tables as follows:

### 2.1. Primary key

In a data table, the establishment of the primary key is necessary. The establishment of the primary key field can conveniently constrain the field values. A data table can contain only one primary key. Primary key cannot have default and null values; it must has unique value. For all fields in a datasheet, the primary key is to be built on the column that is most commonly used in the datasheet. The length of the primary key as an index also affects the performance of the query. In general, the length of the primary key should be less than 25 bytes. Because if the primary key length is too long, the index file size will be relatively large, and always make reading and writing disk less efficient. In general, a compound primary key is not recommended if it is not particularly necessary because it complicates the federated query operation and increases the size of the foreign key datasheet.

The primary key is usually based on an integer field, and generally the string field is not used as the primary key.

Typically, meaningful business fields are not used as primary keys, for example, if a data table is designed to store user orders. Then, the order number is not appropriate as the primary key, because sometimes the order number may be invalidated and the order number of the newly added data may be the same order number as the deleted order number.

Therefore, in this case, the order number is not appropriate as the primary key. In general, we use the proxy primary key, not the business primary key. For example, in the above table, we add a single column in the order table as the primary key field.

For a data table containing 10 million pieces of data, table 1 illustrates the difference in the time taken to query the data before and after primary key index of the data table:

| Table 1. Test for difference in query time whether table has or not has primary key (ms⁻¹) |
|---------------------------------------------------------------|
| table has or not has primary key | Table has primary key index | Table has not primary key index |
|---------------------------------|----------------------------|-------------------------------|
| Using time                      | 1.2                       | 47.3                          |

Visible, the establishment of the primary key index of the field, retrieval efficiency greatly improved.

### 2.2. Foreign key

In a database system designed to comply with 3nf, foreign keys can be used to constrain multiple tables if there are associations. The function of a foreign key is to establish a mapping between tables in a relational database. The foreign key points from the child table to the primary key of the parent table, and the column in which the foreign key is located is an attribute column of the parent table, but in the case of frequently querying the parent table and less using the child table, the disk space can be saved and the query speed can be improved by separating the datasheet.

### 2.3. Index

Indexing on the fields of a datasheet is obvious for improving database performance.

The reason is that if the data is queried on the unindexed table, and if the amount of data is very large, the system has to perform the entire Datasheet scan matching when the query operation is carried out, which wastes more time and system resources.
If we build the index on some specific key columns, the original data table structure does not change, the query operation can be quickly located, so as to improve the query speed, improve the retrieval efficiency and speed up the retrieval time.

Index creation need to follow certain rules. One is that indexes should be created for columns that are frequently queried. Because indexes are built on fields that are frequently queried, you can reduce the amount of data you want to access. Second, the field value is not unique to the column, is not suitable for the creation of indexes. This is because, if the field value is not unique, or repeated values too much, if the establishment of foreign keys, due to the existence of more repeated values, when accessing the data will generate a large number of random IOUs, reducing the performance of the query. Third, if a field updates data more frequently, it is not appropriate to create an index.

Updates the data for a field, and if the field has an index property, the index data is updated at the same time.

If the field data is updated frequently and the index file is updated frequently, the IO traffic will be greatly increased, the system resources will be consumed greatly, and the performance of the system will be reduced.

For a table containing 10 million pieces of data, table 2 illustrates the difference in the time taken to query the data before and after the general index of the table:

Table 2. Test for difference in query time whether table has or not has general index (ms⁻¹)

| table has or not has general index | Table has general index | Table has not general index |
|----------------------------------|------------------------|-----------------------------|
| Using time                       | 0.51                   | 47.3                        |

For some datasheets, querying multi-field data takes too long if some of the fields in the query do not have an index. If you set up an aggregate index on multiple fields, the query time is significantly reduced, and 10 million pieces of data are tested for the difference in time, as shown in Table 3:

Table 3. Test for difference in query time whether table has or not has aggregate index (ms⁻¹)

| table has or not has aggregate index | Table has aggregate index | Table has not aggregate index |
|-------------------------------------|---------------------------|------------------------------|
| Using time                          | 0.01                      | 45.2                         |

Visible, the establishment of the index of the field, retrieval efficiency greatly improved.

2.4. Query optimization method
In the data query, we should try to avoid sub-query operations, sorting operations, join query operations and other high-traffic query operations.

Query multiple tables, complex data, you can create temporary tables as a data source transition, which will improve query efficiency, of course, will increase disk overhead.

When querying with the where clause, conditions should be added to avoid complex expressions, or long strings.

2.5. Database lock
Because database systems are mostly parallel processing systems, if the data is processed concurrently. We need to use locks to ensure data integrity.
It is possible to encounter deadlock problems in transaction processing, but in the design of query statements, it is necessary to consider reducing the number of long transactions, in transaction processing, reduce user interaction. Do not let the user control the length of the transaction.

Avoid batch data execution at the same time, especially if good things are done with the same data table.

3. Expansion Design of Data table
Although following the third normal form (3NF) design of the datasheet, it has great advantages. However, in the practical application process, sometimes we have to carry on the de-normalized data table design. That is to say, out of the third paradigm. For example, if you frequently query data, data calculations, and other operations on a data table, sometimes you need to reduce the response time to improve access performance.

In order to reduce the corresponding time, there are usually the following methods: dividing the data table, storing the redundant data of the query, merging the related data table, and so on.

Although following the third normal form (3NF) design of the datasheet, it has great advantages. However, in the practical application process, sometimes we have to carry on the de-normalized data table design. That is to say, out of the third paradigm. For example, if you frequently query data, data calculations, and other operations on a data table, sometimes you need to reduce the response time to improve access performance.

In order to reduce the corresponding time, there are usually the following methods: dividing the data table, storing the redundant data of the query, merging the related data table, and so on.

3.1. Partition table
Partitioning is especially important when data is complex and has many foreign keys, and the efficiency of processing tables of nearly 10 million data is not high.

The following is an example of a partitioned datasheet. Divided data table is divided into two ways: horizontal segmentation and vertical segmentation. If the data table is partitioned, the workload of maintaining data integrity is greatly increased. Dividing tables horizontally means dividing all data rows into more than two sections. There is a typical case in which the payroll of all employees is stored on a monthly basis and the payroll data sheet stored can be horizontally divided into multiple parts. If you need to use the partitioned datasheet query, you need to join the datasheet.

For datasheets containing 10 million pieces of data, the data in Table 4 shows the difference in the query time before and after partitioning (horizontally) of the datasheet:

| Number of times   | Horizontally segmented data table | Undivided data table |
|-------------------|----------------------------------|----------------------|
| For the first time| 3.5                              | 136.2                |
| A number of        | 3.5                              | 27.1                 |
| subsequent times  |                                  |                      |

On the premise of following the third normal form, when some fields of a data table are not visited by high traffic, it can be separated from the main table and become another data table.

In this way, it not only follows the third paradigm, but also improves the query efficiency.

Of course, after the data table is partitioned, if the data is updated, the relevant data table will be updated with triggers, which will increase the disk IO resources.

3.2. Derivative data storage
A large number of repeated query operations on a data table, if the repeated process of the same results, that is, the results of a query can be used many times. Or the operation of the query is complex and the
system resources are consumed greatly. In this case, the query results need to be stored, which is called redundant storage.

If a row of data requires double counting, an additional column is added to the datasheet to store the query calculation results. Of course, the resulting data source column needs to be updated with a trigger if the data changes. Obviously, storing redundant data can speed up access, but it violates the principles of the third paradigm and increases the workload of maintaining data integrity. When necessary, you must use triggers to update stored redundant data.

3.3. Data file placement scheme
The so-called data file placement scheme is to evenly distribute the database table files in the disk system so as to balance system IO access and avoid IO bottleneck. The following are the main rules to be followed: Placing the system transaction log on a separate disk reduces IO resource consumption and increases the likelihood of system failure recovery. IO contention can be avoided by dispersing frequently manipulated tables across different disks.

4. Equations
In a word, many realistic factors should be taken into account in the design of database tables. It is necessary to consider not only following the normal form rules, but also considering the optimization of system performance. We should not only improve query efficiency, but also consider reducing disk IO loss.

References
[1] Wang, S. Sa, S. (2014) Introduction to database system. Higher Education Press, Beijing.
[2] Connolly, T. Begg, C. (2016) Database system: design, implementation and Management (Foundation). China Machine Press, Beijing.
[3] Wang, Y. Liu, J. Li, L. (2001) Technology and application of large-scale database. Chongqing university Press, Chongqing.
[4] Wang, X. (1992) Software Design Methodology. Tsinghua University Press, Beijing.
[5] Lei, J. Ye, W. Lou, Y. (2015) Principle and Application of Database. Tsinghua University Press, Beijing.
[6] Wan, C. Liao, G. Wu, J. Liu, X. (2017) Principle and Design of Database system (3rd Edition). Tsinghua University Press, Beijing.