Database Design And Performance

Ziyu Wang

He Wang

Troy University Computer Science Department

***Abstract*— The purpose of this paper is to create a database and use this database to understand and test the performance of the database under different conditions and the security of the database and the MYSQL optimizer.**

Keywords—MYSQL, JMeter, Query, Optimizer, Database, Index，B-Tree，Hash.

• INTRODUCTION

The invention of the computer has brought tremendous changes to human life and work. If you don't use computers in management, it's impossible. An invoice database allows a business to easily identify and refer to individual transactions with clients. While visiting a web page, if it does not load quickly, customers will leave. If an application's working hours are too long, you will simply close it. If the application makes users struggle to meet their needs, they will go elsewhere. We usually don't think about performance issues until we need them. I find that premature optimization is not worthwhile, but that does not mean it is unimportant. If you can see any issues at the very beginning of the workflow, then applying fixes will be cheaper and easier. This paper focuses on the database design and performance test and optimization of the database. From the code level, the logic structure of the MySQL query optimizer is not very clear, but from the technical level, it can be divided into two stages, one is the logical query optimization stage, and the other is the physical query optimization stage.

• RELATED WORK

A. Database Index Selection

Many people have related work: 1. The B tree or T tree are applied to the database by Zhiqiang Hu, improving the index performance, how to pick these two algorithms to optimize the query, insert and delete performance of database indexes promptly. 2. A tool is proposed by Rinta Kridalukmana that is used to have a test of the logical model for an achieved database as opposed to the one that creates its original requirements. An XML document is used by the proposed tool to represent the request and a JDBC approach is used to carry the logical model of the implemented database. 3. T.J. Moore describes a test process modeling tool used in the process of both the design and manufacturing to identify the most affordable processes to perform testing with no sacrifice in test coverage or productivity, including eliminating experimental process steps and optimizing the test process.

B. database relational model

Before building the database, we must first design the database relational model and make it meet the requirements of the third normal form. The field customerID is the primary key in CUSTOMER and also exists in INVOICE as a foreign key, where it connects each invoice to its customer.

C. Database Optimizer

SQL optimization usually includes two tasks: one is logical optimization, and the other is physical optimization. These two tasks need to modify the morphology of the parse tree and turn the parse tree into a query tree. Among them, logical query optimization will generate a logical query execution plan.

In the process of generating logic programs, according to the principle of relational algebra, the syntax analysis tree is transformed into the style of an algebraic syntax tree, and some predicates in the original SQL numbers are changed into styles such as logical algebra operations. These styles are a temporary intermediate state. , After further logical query optimization, such as executing dynamic transfer, selection push-down, etc. (such as nodes moving down, some nodes moving up), a logical query execution plan is generated.

After generating the logical query plan, the query optimizer will further optimize the physical query of the query tree. Physical optimization will transform logical queries, and the content of the transformation is mainly to adjust the order of connections. The connection sequence determined by the SQL statement is processed by the multi-table connection algorithm, which may cause the connection sequence between the tables to change, so the shape of the tree may be adjusted. In addition to adjusting the connection order of the tables, physical query optimization also uses the cost estimation model to evaluate the scanning method of a single table and the connection algorithm of the two-table connection. The operation with the least cost in each operation is selected as the basis for the next optimization. . The final result of physical query optimization is to generate the final physical query execution plan.

D. Index Data Structure

Suppose we have an array [10 17 19 26 27 29 30 36 47 90], and if we want to find the data item 29, then we will first load the disk block 1 from the disk to the memory. At this time, an IO will occur, and it will be divided into two in the memory. Search to determine that 29 is between 17 and 35. Lock the P2 pointer of disk block 1. The memory time is very short (compared to the IO of the disk) and can be ignored. The disk block 3 is transferred from the disk through the disk address of the P2 pointer of disk block 1. Load to the memory, the second IO occurs, 29 is between 26 and 30, lock the P2 pointer of disk block 3, load disk block 8 to the memory through the pointer, the third IO occurs, and at the same time, a binary search in the memory finds 29. End the query, a total of three IOs. The real situation is that a 3-level b-tree can represent millions of data. If millions of data searches only require three IOs, the performance improvement will be huge. If there is no index, each data item will have one IO, then a

total of millions of IOs are required, which is obviously very expensive.

Through the above analysis, we know that the number of IO depends on the height h of the b number. Assuming that the data of the current data table is N and the number of data items in each disk block is m, then h= log (m+1)N, When the amount of data N is constant, the larger m is, the smaller h is; and m = the size of the disk block/the size of the data item. The size of the disk block is the size of a data page, which is fixed. If the data item occupies The smaller the space, the greater the number of data items, and the lower the height of the tree. This is why each data item, that is, the index field, should be as small as possible. For example, int occupies 4 bytes, which is half less than bigint8 bytes. This is why the b-tree requires that the actual data be placed on the leaf nodes instead of the inner nodes. Once placed on the inner nodes, the data items of the disk block will drop significantly, leading to the height of the tree. When the data item is equal to 1, it will degenerate into a linear table.

- DESIGN OF DATABASE RELATIONAL MODEL

Before building the database, we must first design the database relational model and make it meet the requirements of the third normal form. The field customerID is the primary key in CUSTOMER and also exists in INVOICE as a foreign key, where it connects each invoice to its customer.

A. First Normal Form Design

First, we have to consider whether the structure of Table 1 meets the requirements of the 1NF. First Normal Form (1NF) is a characteristic of relationships in relational databases. In this database, each column of data is indivisible, there are no multiple values in the same column, and there are no duplicate attributes, each row contains information about only one instance.

B. Second Normal Form Design

In this database, each table can be uniquely distinguishable. Each table has a primary key and the other elements correspond to the primary key one by one, that is, the other elements depend on the primary key. But if I want to update the data in one column then the data in other columns have to change with it. Suppose you want to delete a value in a column, then other columns on that row will also be deleted, which will cause a database abnormally. Data duplication and redundancy: the same customer bought the product m times, the information on the customer table is duplicated m-1 times.

C. Third Normal Form Design

In this database, all data elements are independent of each other, and there is no other functional relationship. And there is no dependency of some entities on other non-primary key data entities. In conclusion, Our database meets the three normal form designs.

- DATABASE AND RESEARCH OBJECT

First, create a dataset named ‘invoiceappdb’ which has 6 tables in the database: Customer, PaymentTable, PaymentTable, Users, InvoiceTable, Invoiceitem, ProductTable.

Customer (CustomerID , Name, Address, PanNo, EmailID, IncludeTaxAmount, PlaceOfSupply CustomerName, CompanyName, GetRegisterNo, MobilePhone, Opening Balance, CustomerForm)

PaymentTable (CustomerID, AmountReceived, PaymentDate, PaymentID, PaymentMode, TransactionID, FinancialYear)

Users (UserID, EmailID, Name, Password, PhoneNumber, Roles)

InvoiceTable (InvoiceID, InvoiceDate, Subtotal, TaxAmount, Address, CustomerID, PanNo, PlaceOfSupply, InvoiceNumber, InvoiceDate, RegisterNo, VehicleNo, CreatedTime, LastModifiedTime)

Invoiceitem (InvoiceID, ProductID, Quantity, TaxAmount, ItemTotal, ItemPrice, GstTaxRate, ItemOrder, LineitemID)

ProductTable (ProductID, Unit, Price, GstPercentage, ProductType, ItemDesc, StockKeepingUnit, Status).

A. Security of Database

User table: For example, I am the owner of this supermarket, After you purchase my products, I will give you an invoice (bill), only I can bill you, so I am the owner of this invoice, I am the person who writes the bill. However, only those who meet the conditions in the user table can make a bill, if anyone can make a bill, the supermarket will be chaotic. Therefore, we specify in the user table: you must meet the criteria entered in the database, UserID, EmailID, Name, Password, PhoneNumber, Role you can open this invoice bill.

B. Database Function

The company's customer and order and user management system includes N modules, including Invoice module, Payment module, Product module, and Users module.

1) Invoice

The main goal of an invoice is to generate a sale record for the business and its customers. An invoice provides written verification of the purchase contract between your business and your customers. Invoices define the rules for your payment requirements and allow you to get paid for your services faster.

2) Payment

The payment table records each payment made by a customer, with information such as the amount and the products being paid for (when applicable) and the date customers purchased and their financial information.

3) Product

Specific information about items is stored by the product table. We propose to make it available to our customers. It defines a product database as a gathering of product-related data, deposited and managed digitally. Product table also include status of the product. The status indicates whether the product has been recently changed, or if it has been deployed on your store, or if has been retired from your catalog.

4) User

A list of users who have the right to get into the MariaDB server and their overall priviledge are stored by the USER table, The table itself is queryable and whilst it can be updated

directly. To use GRANT and CREATE USER to add users and permissions is preferrable.

- SECURITY AND INTEGRITY DEMANDS

Privacy begins with the view mechanism, where different parties can only access their priviledged view. Then through the user authorization mechanism, the user level is identified through user login, and user permissions are assigned according to this level to achieve a higher level of data security.

Integrity requirements are used to ensure that the value of the primary attribute of each processing object is unique and generally cannot be empty; the value of the reference attribute of each processing object must come from the referenced attribute. User-defined integrity (in accordance with actual requirements) can be used to ensure that the data meets the requirements of higher specifications. The detailed integrity requirements can be seen in the logical design stage of the system.

Database integrity (Database Integrity) indicates the logical concordance, rectitude, validity and compatibility of the contents of the DB. Database integrity is ensured by a variety of integrity restrictions. The integrity constraints of database might be implemented by DBMS or application. DBMS-based integrity constraints are kept in the database as part of the schema. DBMS-implemented database integrity is programmed according to the DB design steps, and application-implemented database integrity is incorporated into the application design.

RELATIONAL DATA MODEL

A. E-R diagram

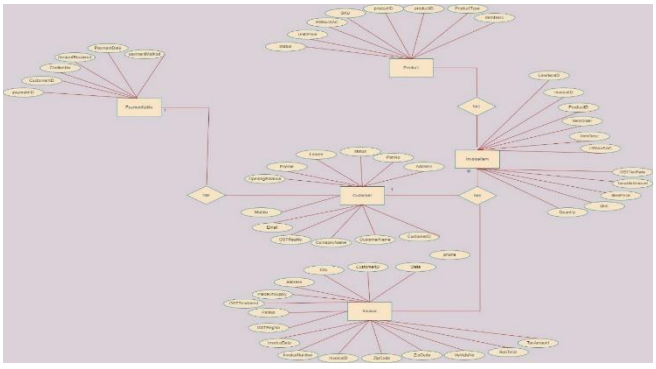

Figure1 E-R Diagram

B. Relationship Model Establishment

The relational model is transformed from the ER diagram. In fact, it is to clearly show the entity, the attribute of the entity, and the connection between the entity. This transformation typically adheres to the following rules. An entity type is being converted to a relational pattern. The attribute of the entity is the attribute of the relationship, and the code of the entity is the code of the relationship.

This database system includes multiple relational modes such as Invoice, payment, product, user, etc.

Customer (CustomerID, Name, Address, PanNo, EmailID, IncludeTaxAmount, PlaceOfSupply CustomerName, CompanyName, GetRegisterNo, MobilePhone, Opening Balance, CustomerForm)

PaymentTable (CustomerID, AmountReceived, PaymentDate, PaymentID, PaymentMode, TransactionID, FinancialYear)

Users (UserID, EmailID, Name, Password, PhoneNumber, Roles)

InvoiceTable (InvoiceID, InvoiceDate, Subtotal, TaxAmount, Address, CustomerID, PanNo, PlaceOfSupply, InvoiceNumber, InvoiceDate, RegisterNo, VehicleNo, CreatedTime, LastModifiedTime)

Invoiceitem (InvoiceID, ProductID, Quantity, TaxAmount, ItemTotal, ItemPrice, GstTaxRate, ItemOrder,LineitemID)

ProductTable (ProductID, Unit, Price, GstPercentage, ProductType, ItemDesc, StockKeepingUnit, Status)

The 'InvoiceID' is the foreign key for the Invoiceitem table and invoiceID will be used by the 'invoice item' table to connect or reference to the invoice table. Because Invoice ID is the primary key for the invoice table and the foreign key for the invoice item table.

C. *Fact Tables Relationship*

1) The relationship between our invoice table and invoice item table is one to many, so the invoice table and invoice item table cannot be merged into one table.

2) The relationship between customer and payment is one to many, a customer can have more than one payment, the same person can enter the supermarket twice to buy goods and thus get two payment bills. We have only one user in our dataset, single user, so we don't need to connect it with other tables. customer cannot generate invoice . In the invoice, we have the cutomer ID, which is used to connect the customer.

3) Why is that there is an invoice table and there is still an invoiceItem table? For example, a customer in the customer table goes to the supermarket to buy something, he buys 5 things, these 5 things are put into the Product Table, so from the figure you can see that the relationship between the customer table and the product table is one to many relationship. But when the owner (user table) generate the invoice, he will give you a single invoice (there are 5 products on this bill), User gives you a single invoice, and there are 5 items in this single invoice, so the relationship between our invoice table and invoice item table is one to many, so invoice table and invoice item table cannot be merged into one table.

4) The relationship between customer and payment is one to many, a customer can have more than one payment, the same person can enter the supermarket twice or three times to buy goods and thus get 2 or 3 payment invoices(bills). The database has only one user in our dataset, single user, so don't need to connect it with other tables. customer cannot generate invoice . in the invoice, there is the cutomer ID, which is used to connect the customer.

- DATABASE PERFORMANCE TEST

A. JMeter Interpretation of test results

First, create 10 threads, which means 10 users will query the database at the same time. And then set 30 seconds as the ramp-up period, which means it will take 30 seconds for the database to finish the request. The JMeter will take 30

seconds to get all 10 threads up and running. Each thread will start 3(30/10) seconds after the previous thread was begun. Loop Count is 1 means: 1 iteration for each user in the group using Loop Count. In this picture, JMeter will take 30 seconds to get all the 10 threads up and running. Create a pool whose variable name is ‘ Test_pool ‘. The max number of connections is 10. Set the Database URL as jdbc:mysql (format) localhost:3307, and find it on MYSQL workbench. Invoiceappdb is the database name. Set the JDBC Driver class as com.mysql.jdbc.Driver, which is a unique driver for Mysql Username is root by default, I find it in MYSQL workbench (connect to database part). And then fill in the Pool name, which I just seted into the blanks. And then select the query type as a select statement. To write a query, SELECT * from customertable ORDER BY customer name ASC to search all the customer names in the customertable and order them by ascending order.

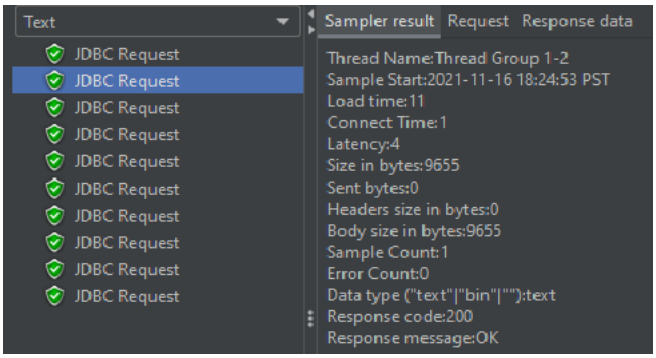

Figure 1 JDBC Sampler Result

In this result, thread 2 has a loading time of 11, which means it takes 11 milliseconds for thread 2 to run the query. Thread 2 will use 11 milliseconds to return the result.

2. Connect time: In How long (how much time) does the database connected to the Jmeter

Latency means the 2nd thread has to wait 4 seconds for the 1st thread to finish.

3. Response code(200) is a digital representation of response message (OK)

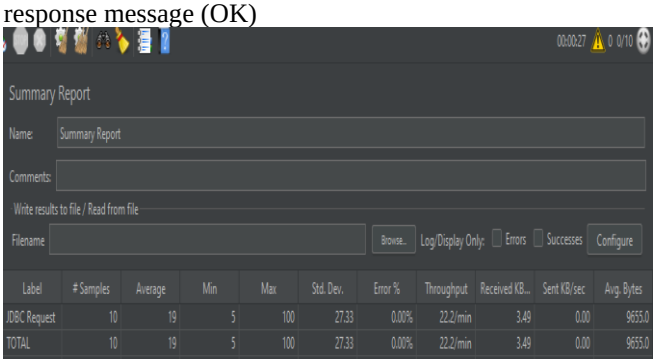

Figure 2 Summary Report

In the summary report, we have 10 samples,
The average execution time of the query of these 10 users is 19 milliseconds.
The minimum execution time of the query of these 10 users takes 5 milliseconds.
The maximum execution time of the query of these 10 users takes 100 milliseconds.
The standard deviation of the execution time of the query is 27.33 mimlliseconds.
Error is 0
*Throughput: number of requests / (total time) = 10 requests/ 27 seconds *60 = 22.22 requests/ min*

B. Advantages of functional testing.

It does not depend on the interface, if the service starts normally and the parameters are passed clearly, test cases can be added and tests can be executed. 2. Test scripts do not require programming, familiar with HTTP requests and business processes, you can write test cases based on input objects on the page. 3. Test scripts are easy to maintain, you can copy the test scripts and save a part separately. 4. you can skip the page limit and add illegal data to the backend program, so you can test the robustness of the backend program. 5. using badboy to record test scripts, you can quickly form test scripts. 6. Jmeter assertions can verify whether there are values in the code that need to be obtained. 7. using parameterization and the function provided by Jmeter, you can quickly complete the test data addition and modification, etc.

C. Disadvantages of functional testing.

The use of Jmeter can not verify the JS program, and can not verify the page, so you need to verify manually. 2. Jmeter's assertion function is not very powerful 3. even if the JMeter script is executed smoothly, it is still impossible to determine whether the program is executed correctly, and sometimes it is necessary to enter the program to check the response data of Jmeter. 4. JMeter scripts need to be saved as local files for maintenance, and each script file can only save one test case, which is not conducive to the maintenance of scripts.

- MYSQL OPTIMIZER

A. Cost Model

We know that MySQL has been evolved for decades, but the optimizer still uses hard-coded weight values to measure the resource conditions such as CPU, and these weight values are based on the experience many years ago or even ten years ago I want to see how fast the hardware has developed over the years. Hundreds of core servers are not a few, especially large-scale use in some large companies, SSD specific daily, NVME SSD(directly connected to CPU) network imagination. All this has affected the realization and realization of the database system. Defining those hard-coded permissions is outdated, we need to provide users with a method, and even further, can be intelligently set automatically according to the hardware environment.

If we run the server cost query it will give us result like below:

```
mysql> select * from mysql.server_cost;
+----------------------------+------------+---------------------+---------+---------------+
| cost_name                  | cost_value | last_update         | comment | default_value |
+----------------------------+------------+---------------------+---------+---------------+
| disk_temptable_create_cost |       NULL | 2021-08-22 02:52:57 | NULL    |            20 |
| disk_temptable_row_cost    |       NULL | 2021-08-22 02:52:57 | NULL    |           0.5 |
| key_compare_cost           |       NULL | 2021-08-22 02:52:57 | NULL    |          0.05 |
| memory_temptable_create_cost |       NULL | 2021-08-22 02:52:57 | NULL    |             1 |
| memory_temptable_row_cost  |       NULL | 2021-08-22 02:52:57 | NULL    |           0.1 |
| row_evaluate_cost          |       NULL | 2021-08-22 02:52:57 | NULL    |           0.1 |
+----------------------------+------------+---------------------+---------+---------------+
6 rows in set (0.00 sec)
```

Figure 4 Server Cost

And now if we run the command update mysql.server_cost

```
mysql> update mysql.server_cost set cost_value = 40 where cost_name = 'disk_temptable_create_cost';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Figure 5 Update

Now we can see after the update our cost name has been reduced significantly.

```
mysql> select * from mysql.server_cost where cost_name = 'disk_temptable_create_cost';
+----------------------------+------------+---------------------+---------+---------------+
| cost_name                  | cost_value | last_update         | comment | default_value |
+----------------------------+------------+---------------------+---------+---------------+
| disk_temptable_create_cost |         40 | 2021-11-29 19:32:48 | NULL    |            20 |
+----------------------------+------------+---------------------+---------+---------------+
1 row in set (0.00 sec)
```

Figure 6 Cost Time

B. Global cache

The global cache maintains current cost model information. The user thread will determine whether the local pointer has been initialized at the time of lex_start. If not, the pointer will be copied from the cache to the local.

C. Hidden index in optimizer index

If we delete the index in an explicit way, if the index is found to be deleted after the deletion, we can only add the deleted index back by creating an index. If the amount of data in the database is very large, or the table is relatively large, this cost of operation is very high. But if this index is set as a hidden index first, the query optimizer will no longer use this index. However, this index still needs to be maintained by the MySQL background. When it is confirmed that setting this index as a hidden index system will not be affected, Then delete the index completely. This will not affect the performance of the entire database even if the error is deleted. For example, if we want to hide the index of the employee table in our database, we should do it like the following：

```
mysql> create database if not exists employees;
Query OK, 1 row affected, 1 warning (0.00 sec)
```

Figure7 hidden index

- HASH INDEX VS BTREE INDEX

In the previous part of the algorithm in the database, we have already mentioned the algorithm of the b-tree, and the hash algorithm is to transform the input of any length into a fixed-length through the Hash algorithm (generally there is a direct remainder method or a product rounding method) Output, the output is the hash value. This conversion is a compression mapping, that is, the space of the hash value is usually much smaller than the space of the input, and different inputs may be hashed into the same output, and it is impossible to uniquely determine the input value from the hash value.

Here we use a different index to see which index can find the specific ID from the same table. First, we use the same table to find users which ID is>900 and <950(total we have 100000 IDs).

```
mysql> select name from bigdata where id>900 and id<950;
```

Figure 8 Select query

It returns us the 41 results with 0.01 seconds. This is the default index, so it's a Btree index. (The index method in the MYSQL is default as B-Tree)

```
| Nl |
| 7V |
| he |
| wn |
| 5U |
| ls |
| YQ |
| t8 |
| Zg |
| Oa |
| q6 |
| MV |
| RA |
+------+
41 rows in set (0.01 sec)
```

Figure9 B-tree Result

Now we are creating a hash index to see what's different in the same query.

```
mysql> create index id using hash on bigdata(id);
Query OK, 0 rows affected, 1 warning (0.27 sec)
Records: 0  Duplicates: 0  Warnings: 1

mysql> select name from bigdata where id>900 and id<950;
```

Figure10 Hash Index

It returns us the same 41 results but using less time.

```
| YQ |
| t8 |
| Zg |
| Oa |
| q6 |
| MV |
| RA |
+------+
41 rows in set (0.00 sec)
```

Figure 11 Hash Result

(using less than 0.01 seconds so it will display 0.00 sec). So what factors cause different indexes, in the same table, make the same query have different performance?

We are negligent in the algorithm in the previous database. The time complexity of the B-Tree algorithm is O(logN), the binary tree search tree, and the hash algorithm we just introduced is an algorithm with time complexity of O(1). And a value in the lookup table, similar to the algorithm we introduced before, B-Tree needs to traverse the tree height, and the hash algorithm only needs to find the matching hash value, so the hash index, in this case, is much faster than the B-Tree indexes.

When is the tree tallest? That is when the tree has the least forks, that is, the root node only has two forks. Each node has $\lceil \frac{n}{2} \rceil$ branches (this is the definition of B tree, and each node (except the root) must contain at least this many child nodes). The root node divides the tree into two subtrees, and on average, the value of each subtree is N/2, and the tree height is h, then

$$(\frac{n}{2})^h \geq \frac{N}{2}$$

This means that each node has at least n/2 choices corresponding to the next node, and there are a total of h such choices (one choice per layer). This choice needs to cover all possible leaf nodes, so the tree height is

$$h \geq log_{\frac{n}{2}}(\frac{N}{2})$$

If we use the Big O notation of time complexity, B-Tree can become

$$O(log_n N)\ or\ O(logN)$$

Therefore, according to the results of this experiment, we can conclude that the hash algorithm is much faster than the B-Tree index when performing peer-to-peer comparisons. Therefore, in this case, we should use the hash index (if the value is different High performance, and based on equivalent search (=, <, >, in), The hash index is a more efficient choice), but if the difference of the value is relatively poor, and the range search is the main, B-tree is more Good choice, it supports range search. Since the Hash index compares the Hash value after the Hash operation, it can only be used for equivalent filtering, and cannot be used for range-based filtering. In addition, when the Hash index encounters a large number of equal Hash values, the performance of the hash index at this time is not necessarily higher than that of the B-Tree index. For index keys with low selectivity, if a Hash index is created, there will be a large amount of record pointer information associated with the same Hash value. It will be very troublesome to locate a record in this way, and it will waste multiple accesses to the table data, resulting in low overall performance.

- FUTURE WORK

In future work, I want to explore what query needs to be optimized. For this problem, we need to consider the impact on the entire system. What optimization of Query can bring greater benefits to the system as a whole requires optimization. Generally speaking, high-concurrency and low-consumption (relative) queries have a far greater impact on the entire system than low-concurrency and high-consumption queries. Suppose there is a Query that is executed 10,000 times per hour and requires 20 IOs each time. Another Query is executed 10 times per hour and requires 20,000 IOs each time. Let's analyze it first through IO consumption. It can be seen that the total number of IO consumed by the two queries per hour is the same, both are 200,000 IO/hour. Suppose we optimize the first Query and reduce it from 20 IOs to 18 IOs, that is, only reduce 2 IOs, then we save 2 * 10000 = 20000 (IO/hour). And if we want to achieve the same effect by optimizing the second query, we must reduce each query by 20000/10 = 2000 IO. I think everyone will believe that saving 2 IOs for the first Query is much easier than saving 2000 IOs for the second Query.

In the future, I also want to study the impact of CPU performance on database access speed. We must first determine whether the bottleneck of this query is IO or CPU. Is it because it consumes too much time in data access, or is it because too many resources are spent in data operations (such as grouping and sorting, etc.), and how to improve the efficiency of the CPU, and even use the GPU for some calculations. We all know that what makes CPUs and GPUs so different is that they are built with different goals. They target two distinct scenarios of application. CPUs rely on robust and versatile nature to handle various data types, it is also necessary to make logical judgments and bring in plenty of branch jumps as well as interrupt handling. They all make the architecture inside the CPU supremely sophisticated. GPUs are confronted with a highly integrated, self-contained, large-scale datum and a clean computing environment void of glitches. The database is exactly this kind of large-scale simple data scenario, which is very suitable for using GPU for calculation. Therefore, if GPU is used for acceleration when processing super-large-scale databases in the future, the speed of the database will increase exponentially, so this is Why I want to study the application of GPU in the database in the future.

- SUMMARY

The database system conveniently realizes the centralized management of data. For an enterprise with various departments, the obstacle of management of data or the intercommunication of information can be intractable. If each department has its own independent data, even if these data are mainly for the department, it is impossible to grasp the overall information as a whole. The establishment of a database system is positive in terms of data utilization. Our purpose in establishing a database is this, and how the performance of the database improve the performance of the database is also very important, because the enterprise wants a system that can be used anytime, anywhere. The database to be accessed is not a database that has to be delayed for a long time each time. After all, the improvement of efficiency can bring direct profits to the enterprise. Moreover, Database systems are incredibly important to a businesses since a business is all about data. At its core, effective data management can bring more profit margins to enterprises. Through this paper, we have established a preliminary database system, and conducted some basic database performance tests and attempts to improve database performance, as well as comparing the efficiency of the database when using different indexes and what different indexes are suitable for We hope that we will have more opportunities to learn about the database and database performance improvement in the future.

REFERENCES

[1] T. J. Moore, "A test process optimization and cost modeling tool," Proceedings., International Test Conference, 1994, pp. 103-110, doi: 10.1109/TEST.1994.527941.

[2] M. Liu and R. Shan, "Design and implementation of the Relationlog deductive database system," Proceedings Ninth International Workshop on Database and Expert Systems Applications (Cat. No.98EX130), 1998, pp. 856-863, doi: 10.1109/DEXA.1998.707505.

[3] L. Wei and W. Zhang, "A Method for Rough Relational Database Transformed into Relational Database," 2009 IITA International Conference on Services Science, Management and Engineering, 2009, pp. 50-52, doi: 10.1109/SSME.2009.79..

[4] W. Yin Mok, "A Feasible Schema Design Strategy for Amazon DynamoDB: A Nested Normal Form Approach," 2020 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), 2020, pp. 903-907, doi: 10.1109/IEEM45057.2020.9309967.

[5] Xiuling He, Yang Yang, Zengzhao Chen, Ying Yu and Cailin Dong, "Form analysis and understanding based on knowledge," 2008 7th World Congress on Intelligent Control and Automation, 2008, pp. 9286-9291, doi: 10.1109/WCICA.2008.4594401.

[6] K. Norvag, "A performance evaluation of log-only temporal object database systems," Proceedings. 12th International Conference on Scientific and Statistica Database Management, 2000, pp. 256-258, doi: 10.1109/SSDM.2000.869795.

[7] S. Y. W. Su, S. Ranka and Xiang He, "Performance analysis of parallel query processing algorithms for object-oriented databases," in IEEE Transactions on Knowledge and Data Engineering, vol. 12, no. 6, pp. 979-996, Nov.-Dec. 2000, doi: 10.1109/69.895805.

[8] D. Taniar, "High Performance Database Processing," 2012 IEEE 26th International Conference on Advanced Information Networking and Applications, 2012, pp. 5-6, doi: 10.1109/AINA.2012.140.

[9] Vamsi Krishna Myalapalli and Bhupati Lohith Ravi Teja, "High performance PL/SQL programming," 2015 International Conference on Pervasive Computing (ICPC), 2015, pp. 1-5, doi: 10.1109/PERVASIVE.2015.7087001.

[10] S. Selvarani and R. Janarthanan, "Performance analysis on ranked queries in uncertain databases," Trendz in Information Sciences & Computing(TISC2010), 2010, pp. 63-67, doi: 10.1109/TISC.2010.5714610.

[11] S. Y. W. Su, S. Ranka and Xiang He, "Performance analysis of parallel query processing algorithms for object-oriented databases," in IEEE Transactions on Knowledge and Data Engineering, vol. 12, no. 6, pp. 979-996, Nov.-Dec. 2000, doi: 10.1109/69.895805.

[12] Q. Li, M. Shao, V. Markl, K. Beyer, L. Colby and G. Lohman, "Adaptively Reordering Joins during Query Execution," 2007 IEEE 23rd International Conference on Data Engineering, 2007, pp. 26-35, doi: 10.1109/ICDE.2007.367848.

[13] "B Tree vs Hash Table", StackFlow, 10/30/2021, Https://Stackoverflow.Com/Questions/7306316/b-Tree-vs-Hash-Table.

[14] "Comparision of B-Tree and Hash Index" https://dev.mysql.com/doc/refman/8.0/en/index-btree-hash.html

[15] "The Optimizer" https://dev.mysql.com/doc/internals/en/optimizer.html

[16] "Controlling the Query Optimizer" https://dev.mysql.com/doc/refman/8.0/en/controlling-optimizer.html

[17] Paul Dubois, *The MySQL Query Optimizer* (ISBN-10: 0-672-32673-6.).

[18] Baron Schwartz, Peter Zaitsev, Vadim Tkachenko, *High Performance MySQL, 3rd Edition* (ISBN: 9781449314286.). O'Reilly Media, Inc.

[19] Carlos Coronel , Steven Morris, *Database Systems: Design, Implementation, & Management 13th Edition* (ISBN-13 : 978-1337627900.). Publisher : Cengage Learning; 13th Edition (January 1, 2018).