



## Study on the Optimization Method of Select Query Sentence in Standard SQL Language

Yuqing Geng

Department of Computer

Kerqin Art Professional College

Tongliao 028000, China

Tel: 86-475-8239-754 E-mail: nmtlgyq@163.com

Chunsheng Zhang

College of Math and Computer Science

Inner Mongolia University for Nationalities

Tongliao 028043, China

*The research is financed by Neimenggu Talents Fund and Neimenggu Education Scientific Research Project (No. NJZY07140). (Sponsoring information)*

### Abstract

In the actual information management system, the query operation is the main part, and the query method is completed by the Select sentence in SQL language, so the efficiency of Select sentence directly influences the query efficiency, and the factors influencing query efficiency include query condition, view, index, link, operator and memory process. In the article, combining with examples, we analyzed the influences of various factors on the query speed and summarized the strategy which used these five factors to optimize Select query sentence, and the concrete example indicated that these optimization process are simple, convenient and effective, and they could enhance the response speed of query.

**Keywords:** SQL language, Select sentence, Query, Optimization

### 1. Introduction

With the gradual development of computer application, the application of large scale database is more and more extensive. In the large scale database, people have begun to concern how to enhance the query efficiency under the premise ensuring the correct result, especially in large scale or complex database, the efficiency of query largely influences the application of the system.

SQL is the representative database language integrating data definition and data control, and it is a sort of interactive language that any one database management system should provide to users. Select sentence is the main tool of database query, and it is the connection ligament between users and database, and user's query operation to the database is implemented by it. Because in same query process, we can express it by multiple forms of Select sentence, and the execution efficiencies of the Select sentences with different forms are different, so when we compile the software of database application, if we can reasonably optimize the Select sentence, we could improve the query efficiency, optimize the query speed of database, and reduce the interactive time between users and data, especially in the application program design of B/S network database, it could reduce users' waiting time and enhance the response time of the system from the view of optimizing database query.

In this article, we would analyze the influences of six aspects including Select sentence query condition, view, index, operator, link and memory on the query algorithm to offer their optimization forms, and these optimization methods would largely enhance the query efficiency of database.

Convenient for the demonstration, the database of "student management" is supposed in the article, and the database includes following five sheets.

- (1) Student (student number, department number, name, sex, birth data, address)
- (2) Achievements (student number, advanced mathematics, foreign language, gym)

- (3) Department (department number, department name)
- (4) Transient student (student number, department number, name, sex, birth data, address)
- (5) Borrowing books (department number, student number, book name)

## 2. Optimization of Select query conditions

### 2.1 Query of "OR"

Translate the query with "OR" in the sub-sentence of "WHERE" into the query contain multiple "UNION ALL" without "OR". When OR represents the limitation condition to different sheets, this method is preponderant.

For example, in the sheet of "achievement", we have respectively established indexes for two fields including "advanced mathematics" and "foreign language", and we want to inquire students' "name" and the achievements of "advanced mathematics" and "foreign language" whose achievement of "advanced mathematics" exceeds 90 or whose achievement of "foreign language" exceeds 95, so the SQL query sentences are

```
SELECT name, advanced mathematics, foreign language FROM achievement WHERE advanced mathematics> 90 OR foreign language> 95
```

This Select sentence may not require the query optimizer use the index, so it would reduce the query efficiency. And it could be improved as follows.

```
SELECT name, advanced mathematics, foreign language FROM achievement WHERE advanced mathematics> 90 UNION ALL SELECT name, advanced mathematics, foreign language FROM achievement WHERE foreign language> 95
```

### 2.2 "Flattening" technology of sub-query

Transform sub-query into link, half-link or inverse-link to achieve the intention of query optimization.

For example, inquire the "department name" that the student's achievement of "advanced mathematics" exceeds 90.

```
SELECT B. department name FROM department B WHERE B. department number IN (SELECT C. department number FROM achievement C WHERE C. advanced mathematics> 90)
```

This query will scan every line in the department sheet to check students' records which fulfill the query condition, so we can take the achievement sheet as the interior sheet of link when optimizing. In this condition, the query is implemented as the usual link, and first filtrate distinct "department number" to the sheet of "achievement" in order to eliminate redundant department numbers. The improved SQL sentences are

```
SELECT B. department name FROM (SELECT DISTINCT department number FROM achievement WHERE advanced mathematics> 90) C, department B WHERE C. department number= B. department number
```

### 2.3 Implementing multiple selection operation in same sheet

The sequence of selection condition largely influences the computation quantity of query sentence, and if we want to enhance the response speed of query, we should write the strict selection condition in front, and put the weak condition back, so in the execution process, the line which couldn't fulfill the conditions will be deleted quickly.

### 2.4 Selection operation of multiple sheets

When implementing selection operation to multiple sheets, the sequence of sheet influences the response speed of query also. So we need a queuing principle, i.e. operating to one selection condition will return the sheet with multiple lines in front, and return the sheet with few lines back to reduce the operation of insert.

For example, suppose the sheet of "student" include 10000 students in Tongliao, and the sheet of "transient student" include 100 students in Tongliao.

- (1) SELECT name FROM student, transient student WHERE address= "Tongliao"
- (2) SELECT name FROM transient student, student WHERE address= "Tongliao"

In the concrete execution process, (1) firstly selects the sheet of "student" and obtains one temporary sheet with 10000 lines, then selects the sheet of "transient student" and obtains the record with 100 lines, and finally implements 100 times insert operation, and insert the 100 lines record into the temporary sheet. Contrarily, if exchange the sequences of two sheets, enter into (2), so first obtain one temporary sheet with 100 lines record, and finally implements 10000 times insert operation, and the computation quantity will far exceed (1).

## 3. Optimizing Select sentence by view

### 3.1 Establishing view

Some problems still exist in the data sheets stored in the database management system, and the first one is the scale of

total sheet is large, but users usually need one little part of it, and every operation will waste much time. And the second one is that the total sheet generally contains some secret data which users couldn't acquire, so we can establish the view to dispose and store data in common use in advance by the form of view, which could largely enhance the speed of query, especially when the scale of view is smaller than the original sheet, the query after transformation is much quicker than original query.

For example, inquire the students whose native place is "Tongliao", we can establish the following view.

```
CREATE VIEW AS SELECT name FROM student WHERE address="Tongliao"
```

### 3.2 Combination of views

Some queries need define the view to reduce the data scale returned in the sub-query, and the combination of views is to eliminate the query with view through combining the view definition with query.

For example, inquire the student's average of "advanced mathematics" whose "department number" is "05".

```
CREATE VIEW AS SELECT AVG (advanced mathematics) FROM achievement WHERE department number="05"
```

## 4. Optimizing Select sentence by index

Index is one important database object in the database, and the essential of index is a series of finger which point at the sheets which pass the index, and once define the index for certain attribute, when searching the attribute, and the database management system will directly search the index of the attribute, so to establishing the index is the important method to optimize the query, but to establish the index will increase the speeding of time and space for the system, so we should keep to following principles when using index.

(1) Establish index to the line used in WHERE sub-sentence, for example, the "student number" in the sheet of "student."

(2) Establish index to the line frequently implementing "grouping" or "ranking", especially for some lines which need to implement ranking with multiple lines, we can establish composite index on these lines, for example, the achievements of various subjects in the sheet of "achievement".

(3) Don't establish index to the line with few different values, for example, the line of "gym" in the sheet of "achievement" which only have two sorts of value such as "eligible" and "ineligible".

(4) For the inserting and deleting of large quantities of data records, first delete the index, then implement data operation, and finally consider rebuilding necessary index to reduce the consumption of system time.

(5) For the line with large numbers of repetitive values, range query, grouping and ranking, we can establish the clustered index, and it will reduce 50% time than non-clustered index.

(6) For the multiple lines storing at the same time and every line contains repetitive values, we can consider establishing the combined index which should cover the key query form index possibly and the precursor line is certainly the line which is used most frequently.

## 5. Optimizing Select sentence by operator

### 5.1 Avoiding using normal expression of fuzzy query possibly

MATCHES and LIKE keywords support the wildcard character matching and it is called as normal expression technically. It is more convenient and direct to use the normal expression to describe the query condition, but this sort of matching especially wastes time, for example, in the query, sometimes, it needs compare the character strings. The LIKE operator could complete the fuzzy matching to the character string, but large numbers of characters should be compared one by one, and the query efficiency will be largely reduced.

For example, student number is numbered by the information such as enrollment year, now we should inquire the information of the student who enrolled in 2005.

```
SELECT * FROM student WHERE student number LIKE "2005%"
```

Even if we establish the index on the field of SNO, we should also adopt the mode of sequent scan under this situation. If the sentence is modified as

```
SELECT * FROM student WHERE student number > "2005000" AND student number < "2006000"
```

This sentence will inquire by using the index when implementing the query, and obviously it will largely enhance the executive speed of sentence.

5.2 Any operation to line would induce the scanning of sheet, it includes the database function, computation expression and so on, and the operation should be moved to the right of the equal sign possibly

For example, SELECT \* FROM student WHERE LEFT (student number, 4) = "2006"

```
SELECT * FROM student WHERE AMOUNT/30 < 1000
```

Any operation result of WHERE sub-sentence to the line is obtained by the computation line-by-line when SELECT runs, so it must implement sheet search, so the index on the line is not used, and if these results could be obtained when inquiring compiling, so it could be optimized by SQL optimizer, use index and avoid the sheet search, so the SELECT could be modified as

```
SELECT * FROM student WHERE LIKE "2006%"
```

```
SELECT * FROM student WHERE AMOUNT < 1000*30
```

*5.3 Work sheet is usually used in IN and OR sub-sentence, which will invalidate the index, and if large numbers of repetitive values don't occur, we can take part the sub-sentence, and the sub-sentence should include index*

For example, the sheet of "student" has twenty thousand lines, and the "student number" has non-clustered index.

```
SELECT COUNT (*) FROM student WHERE student number IN ("20060001", "2006008")
```

The SELECT sentence could be modified as

```
SELECT COUNT (*) FROM student WHERE student number= "20060001"
```

```
SELECT COUNT (*) FROM student WHERE student number= "20061008"
```

Make one addition operation for the results of above two sentences, we will get the same result, and because every sentence uses index, so the execution time will be largely reduced.

*5.4 Comparing with constant possibly in the comparison*

For example, SELECT \*FROM achievement WHERE advanced mathematics > foreign language

The selection condition contain two comparisons of attribute, and the logic computation quantity of the condition comparison is large, so we should avoid use it.

*5.5 Avoiding using negative query and negative forms such as !=, <>, NOT IN and NOT EXIST*

For the query tree of general database, because the traversal method of B-tree is unfit for the comparison of inequality, so these relationship operators could not be used in the optimization of index selection, i.e. the system needs direct search sheet.

## 6. Optimizing Select sentence by link

### 6.1 Link and selection operation by multiple sheets

For the link and selection operation process of multiple sheets, the computation quantity which first implements the selection operation, then implements the link operation is small, and the response time of query is short and the demand of memory is small.

For example, SELECT \* FROM achievement A, student B WHERE A. student number= B. student number AND A. advanced mathematics > 90

```
SELECT * FROM student B WHERE B. student number = (SELECT A. student number FROM A WHERE advanced mathematics > 90)
```

Two sentences return same result, but the response times of query are different, and the former would first link and obtain a temporary sheet with many lines, then select in the temporary sheet, and the later would first select, which will delete large numbers of redundant data, and then link two sheets, so it will largely enhance the performance of database.

### 6.2 Selection of interior and exterior sheets when linking

Before the multiple sheet operation is actually executed, the query optimizer will list several groups of possible link projects and find out the optimal project that the system spending is least according to the link condition. The link condition should fully consider the sheet with index and the sheet with many lines.

The selection of interior and exterior sheets could be confirmed by that the lines matching in the exterior sheet multiply the query times of every line in the interior sheet, and the selection that the product is least is the optimal project.

For example, the sheet of department has 7000 lines, and the department number has one non-clustered index, and the sheet of borrowing books has 180000 lines, and the department number has one clustered index. Under different sheet link conditions, the executions of two SELECT sentences are

```
SELECT*FROM borrowing book A, department B WHERE A. department number= B. department number
```

```
SELECT*FROM borrowing book A, department B WHERE B. department number= A. department number
```

Under the first link condition, the optimal project is to take "borrowing book" as the exterior sheet and take "department" as the interior sheet. Utilize the index on the department, and the I/O times can be represented as 180000 lines in the

exterior sheet of “department” \* 3 pages which should be inquired in the interior sheet department corresponding to the first line of the exterior sheet = 540000 times I/O.

Under the second link condition, the optimal project is to take “department” as the exterior sheet and take “borrowing book” as the interior sheet. Utilize the index on “borrowing book”, and the I/O times can be represented as 7000 lines in the exterior sheet of “borrowing book” \* 4 pages which should be inquired in the interior sheet of “borrowing book” corresponding to every line of the exterior sheet = 28000 times I/O.

So it is obvious that only for sufficient link condition, the real optimal project can be executed.

### 6.3 Same condition limitation of multiple sheets link

If the limitation condition for one data sheet is available to the data sheet which should be linked, in another words, two sheets possess same attribute line and same limitation, so the data quantity of two sheets could be reduced before linking.

For example, look for the department name which “department number” exceeds or equal to “04” and the average achievement of “advanced mathematics”, we can use following optimized SELECT sentences to obtain higher efficiency.

```
SELECT A. department name, B.AVG (advanced mathematics) FROM department A, achievement B WHERE A. department number >= “04” AND B. department number >= “04” AND A. department number = B. department number
```

In the example, we will first eliminate the data which don’t accord with the condition through selecting condition before linking, which will largely reduce the data of two sheets when linking. The method is more important for the sheet which has large scale when linking.

## 7. Process of storage

The storage process is a group of Transact-SQL sentence which is defined in advance and compiled, and it could receive parameter, return status value and parameter value, and it could be transferred again, and when one storage process is transferred, the system transfers the storage process into the memory and compiles it completely, and when the storage process is transferred again, we can deal with it at once, and there is no any extra spending, so the query speed is enhanced.

## 8. Conclusions

The optimization of SELECT query sentence expression could enhance the search performance of data, and the optimization design of query in the development and maintenance of database could enhance the system performance, especially for the database system which is often used and possesses large scale data.

Through the analysis of five aspects, we possibly reduce the I/O times of sheet scan, reduce the demand of memory, enhance the speed of interview, reduce the response time and reduce the computation quantity under the premise that the SELECT query result is correct.

In a word, only to exactly observe and analyze various information offered in the system, fully combine the actual application characters, we could reasonably establish good optimization strategies, and realize rapid and highly efficient data query and application analysis, and finally write high efficient SELECT query sentences.

## References

- He, Manhui, Wen, Tingxin & Qu, Xiaohua. (2003). Optimizing Method for SQL Query Speed. *Journal of Liaoning Technical University (Natural Science Edition)*. No.4.
- Huang, Zhizhen & Yangwu. (2001). The Research on the Optimization Method for Database SQL Query. *Ordnance Industry Automation*. No.1.
- Jiang, Yuanli. (2005). Optimization Methods of Database SQL Query. *Ordnance Industry Automation*. No.6.
- Jin, Tianrong. (2006). Design and realization of Query Optimization in SQL Server. *Control & Management*. No.18.
- Lei, Hongwei, Wang, Kuisheng & Quzhan. (2004). Optimized Strategy for Relational Data Inquiry Based on SQL. *Journal of Beijing Electronic Science and Technology Institute*. No.6.
- Sha, Shixuan & Wang, Shan. (2000). *Introduction of Database System*. Beijing: Higher Education Press.
- Tan, Dingying & Fang, Zhencong. (2005). Optimization Strategy of SQL Inquiring Database. *Computer and Modernization*. No.6.
- Wang, Zhenhui & Wu, Guangmao. (2005). Optimization of SQL Query Sentence. *Journal of Computer Applications*. No.12.
- Zhang, Limin. (2002). *Program Design of SQL Server 2000 Transact-SQL*. Beijing: Higher Education Press.