Original Scientific Paper/Originalni naučni rad UDC/UDK: 004.6:005
Paper Accepted/Rad prihvaćen: 4. 7. 2019.
doi:10.5937/SJEM1902059L

Optimizacija baza podataka u poslovnim sistemima

Zlatko Langović [1]

[1]Faculty of Hotel Management and Tourism in Vrnjačka Banja, University of Kragujevac, zlangovic@kg.ac.rs

Apstrakt: Sažetak: Problem istraživanja u ovom radu jeste poboljšanje performansi poslovnog informacionog sistema kroz optimizaciju rada baze podataka. Definisani su procesi optimizacije sistema za upravljanje bazama podataka koji omogućavaju određene prednosti. Predstavljene su tehnike čijom upotrebom se postiže bolja iskorišćenost računarskih resursa, a koja se ogleda kroz veću brzinu izvršenja upita. Dobijeni rezultati čija primena može smanjiti vreme izvršenja upita, odnosno smanjiti opterećenja memorijskih sistema i procesora, dati su u obliku preporuka za optimizaciju. Krajnji rezultat je produktivniji rad baze podataka, pa samim tim i informacionog sistema.

Ključne reči: optimizacija, sistem za upravljanje bazom podataka, SQL

Database optimisation in business systems

Abstract: The research problem in this paper is to improve the performance of the business information system through the optimisation of the database operation. Defined optimisation processes for database management systems provide certain advantages. The techniques used to achieve better utilisation of computer resources are presented, which is reflected in the higher speed of the query execution. The obtained results whose application can shorten the execution time of the queries, or reduce the loads of memory systems and processors, are given in the form of recommendations for optimisation. Final result is the more productive work of the database, and therefore the information system.

Key words: optimisation, database management system, SQL

1. Introduction

By developing the economic and social aspect, the amount of data that is needed to process and store is exponentially increased. Processes characterised by increasing the amount of information influence the development and administration of information systems, that is, database system development. A variable business environment causes faster data processing by the database system in order to meet business requirements. One way to improve database characteristics is the query optimization process.

The subject of research in this paper is based on improving the commands' performance that have a longer execution time and therefore unnecessarily occupy database resources. This can be solved using certain optimisation techniques. The optimisation effects can affect the work of the database in different ways. Non-optimised queries can occupy available resources.

The aim of the paper is to come up with practical recommendations related to database optimisation, whose application leads to the efficiency improvement of SQL commands executing. Given recommendations have a different effect on instruction performance improvement in different environments. Adequate use of recommendations in defining certain commands helps the optimiser to create a preferable execution plan.

2. Optimiser functions

The optimiser itself is an essential part of database optimisation, because the optimiser output is execution plan that presents the optimal way of command executing. The plan provides a combination of steps that execute SQL command. The goal of an optimiser is to choose a plan for which execution

requires the least number of resources. Also, the optimiser aims to select the one whose execution time is the shortest due to choosing a plan.

Several steps of the optimiser are given below:

1. The optimiser generates multiple potential plans for SQL commands;
2. Optimiser calculates the cost of each plan individually, based on statistics. The cost is the estimated value proportional to the specific resource consumption required to carry out the execution plan. The optimiser calculates the cost of access paths and join operations, which depends on the estimated consumption of computer resources, i.e., i/o, CPU and memory;
3. The optimiser compares the cost of the generated plans and chooses the one with the least cost.

Optimiser operations include (Oracle, n.d):

1. The SQL command transformation,
2. Total costs estimation,
3. Generating a plan of execution.

The main function of the optimiser is to take into account different plans for executing a particular command and to choose the one with the lowest cost. Different plans are possible due to different combinations of access paths, table join method, the merging table ordering. The order of joining is the order in which specific data is accessed, sush as tables, and the connection is made. The number of possible plans is proportional to the number of objects listed in the FROM clause. Given number increases exponentially with the number of objects being merged increasing. The planner internally eliminates other plans when finding the plan with the lowest costs. Internal check is based on the costs of the current executive plan. If the cost of the current plan is high, the planner is trying to find a plan with lower costs. If the current plan has low costs, the planner terminates further plan searching, as there would not be significant improvements. An internal search will be good if the planner initially finds the order of execution that will produce a cost plan close to the optimum, but this rarely happens. (Boicea, et al., 2016)

The execution plan is formed by generating the execution subplans for some embedded subqueries. The inner part is optimized first and subplan is generated for it. The outer part that represents the complete command, the last one is optimised.

3. Some optimisation techniques

The paper presents some techniques for database optimising, that is, techniques which will be used in next pages.

Memory management. By setting certain memory parameters, it can be noticed the difference between commands executed in a few seconds, unlike those that take a few minutes. There are three parameters that need to be set due to properly memory managing:

PGA_AGGREGATE_TARGET,
MEMORY_MAX_SIZE,
MEMORY_TARGET.

An important part of optimisation is adequate memory management. If memory allocates inadequately, performance problems could occur, and in this case, optimisation techniques may encounter restrictions when attempting to improve the efficiency of executing commands. It is therefore desirable to know the memory structure in Oracle and MS SQL server database management systems.

Indexes. In most cases, the cause of poor performance is the use of tables above which no index for the searching columns is made. When there is a large number of rows in the table, there is also an inefficient set of scanning operations for the entire table in order to find those rows that satisfy given condition. Indices can also have a poor performance impact. When data is added, deleted or updated, all table indexes that affect it must be updated due to maintaining given changes, thus extending processing and increasing the system load. (Microsoft, n.d.).

Indexes have three important uses:

1. Quickly find the appropriate rows without using a full table scan operation;

2. They serve to avoid reading the table if the information is located in the index.
3. They serve to prevent sorting. Sorting is executed for a lot of reasons if there are clauses: ORDER BY, GROUP BY, DISTINCT. If the sorting operation requires the same order as in the index, the data can be read from the table via the index.

Partitioning. Partitioning is a method of optimisation, which in most cases is the last one applied. The term means the database division or parts of a database into elements that are independent. Usually it is used to improve performance, simpler data management, as well as to increase data availability. Partitioning is performed by the partition key.

Hint. The tips allow us to influence the optimiser's decision during the execution plan generating. Hint is a mechanism that directs the optimiser in order to choose execution plan with certain access paths or operations. The role of the optimiser is to generate an optimal execution plan. If the optimiser performs well, hints should not be used. However, due to frequent data changes, the statistics are outdated, therefore, the optimizer could make mistakes in choosing the best execution plan. In Oracle databases there is a possibility to lock statistics if they are well, making the use of the hint useless. Hints can be complicated, which is why they are often used as the lowest tool due to changing the execution plan of the command (after changing statistics, changing instance parameters, etc.). The use of a hint can serve to override parameters of parallelism, makes indexes visible, can cache an appropriate query, etc. (Nyffenegger, n.d.)

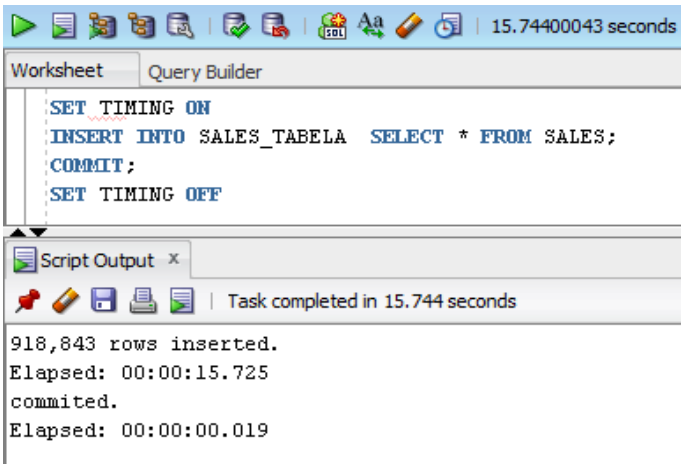
3. Optimisation techniques application

In the following example the application of described optimisation technique will be shown. Study is executed in Oracle SQL developer tool. It will be shown that the application of the hint, index, and partitioning techniques reduce the time/ cost of command execution.

First, the SALES_TABELA table will be created, which will be identical to the SALES table of the SH scheme by columns. The time taken to load 918 843 lines with and without using the APPEND hint is included in the table. Using the hint in this table, the data will be loaded 25 times faster than without hitn usage. APPEND hint otherwise reduces the time by bypassing the memory and buffer cache when data is inserted, and, using NOLOGGING, it does not record anything in the regular log files that normally stores all changes related to the database. (Fiorillo, 2012)

```
CREATE TABLE MY_SALES AS SELECT * FROM SALES WHERE ROWNUM < 1;
SET TIMING ON
INSERT INTO SALES_TABELA SELECT * FROM SALES;
COMMIT;
SET TIMING OFF
```

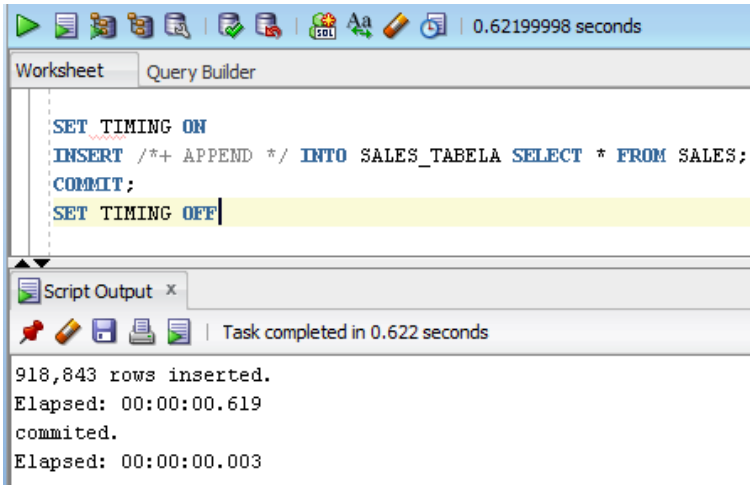
Figure 1: Rows insertion in new table without APPEND hint



Source: Author

```
SET TIMING ON
INSERT /*+ APPEND */ INTO SALES_TABELA SELECT * FROM SALES;
COMMIT;
SET TIMING OFF
```

Figure 2: Rows insertion in new table with APPEND hint



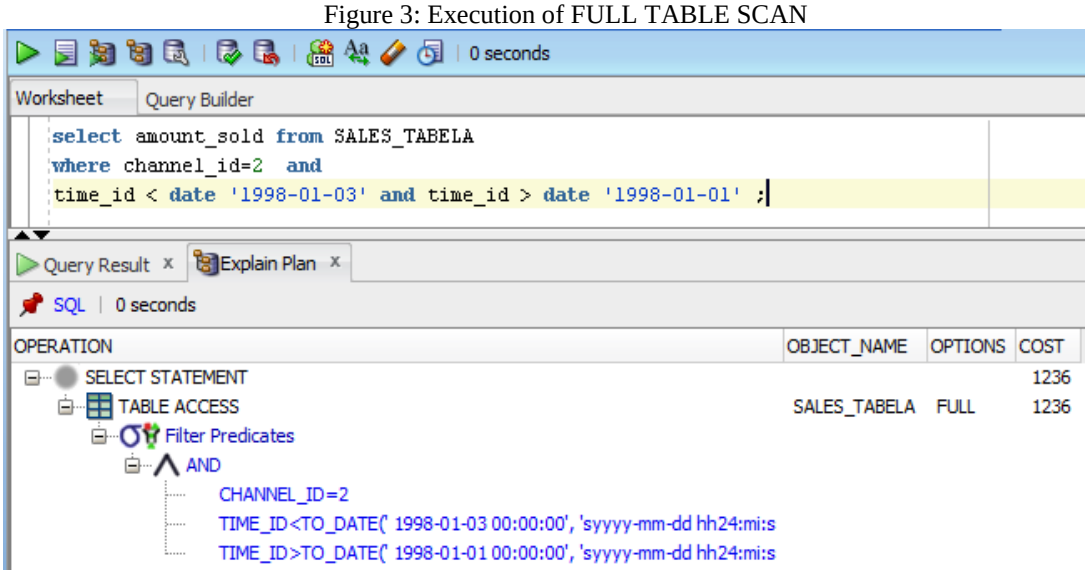
Source: Author

Since the data in the table is inserted, the following command is created:

SELECT AMOUNT_SOLD FROM SALES_TABELA
WHERE CHANNEL_ID = 2 AND
TIME_ID <DATE '1998-01-03' AND TIME_ID> DATE '1998-01-01';

The cost of execution of the command was monitored, which was 1236. The high cost appeared due to FULL TABLE SCAN operation, which was used due to the lack of index.

Figure 3: Execution of FULL TABLE SCAN

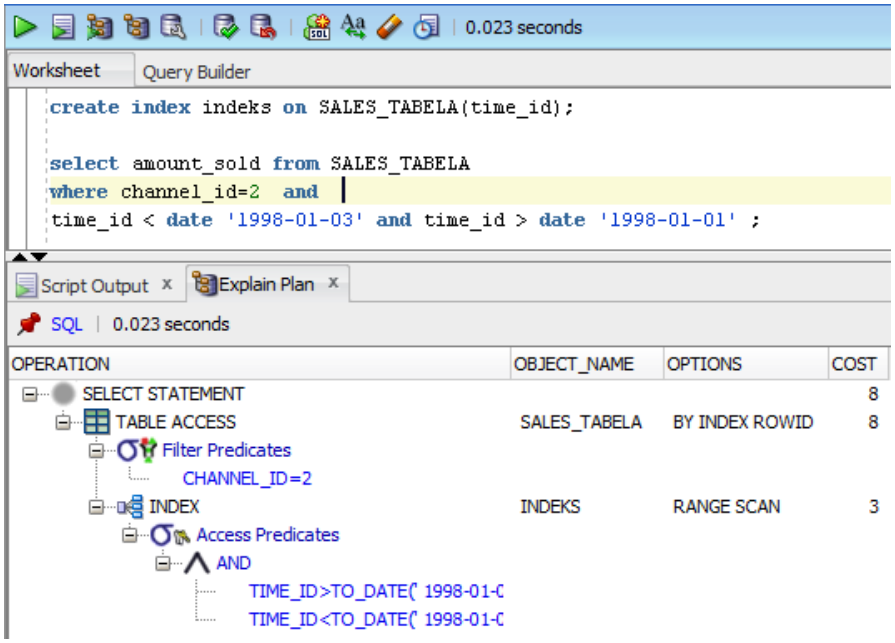


Source: Author

After creating an index over the column time_id, the cost was reduced, even, to 8. Instead of accessing the table, the index from which the data was read, was accessed.

CREATE INDEX INDEKS ON SALES_TABELA (TIME_ID);

Figure 4: Execution of FULL TABLE SCAN by index accession

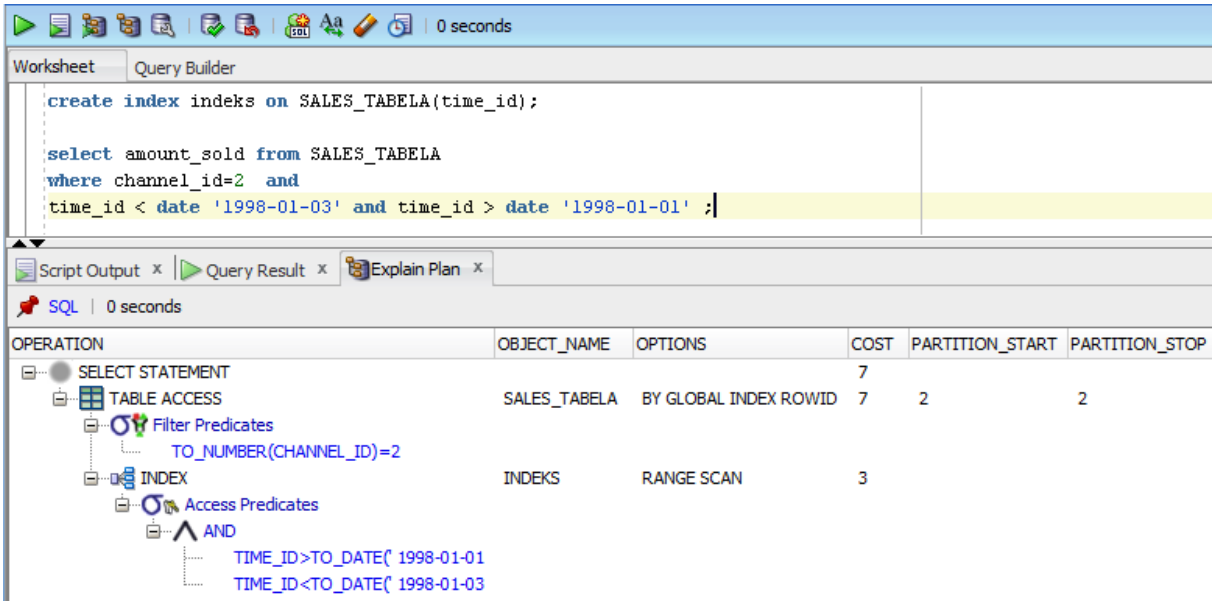


Source: Author

After creating an index, partitions were created, using the RANGE partitioning strategy. Partitioning was performed by the time_id column. 13 partitions have been created, where for each one a certain date range was defined. In the case of performing the same command, but this time with partitions, the cost would be 7. The additional reduction was generated because only partition number 2 was accessed, while the other partitions were eliminated when the command has been performed.

```
PARTITION BY RANGE (time_id)
 (
PARTITION particija_1 VALUES LESS THAN (TO_DATE('01-JAN-1998','dd-MON-yyyy'))
PARTITION particija_2 VALUES LESS THAN (TO_DATE('01-FEB-1998','dd-MON-yyyy'))
    PARTITION particija_3 VALUES LESS THAN (TO_DATE('01-MAR-1998','dd-MON-yyyy'))
PARTITION particija_4 VALUES LESS THAN (TO_DATE('01-APR-1998','dd-MON-yyyy'))
PARTITION particija_5 VALUES LESS THAN (TO_DATE('01-MAY-1998','dd-MON-yyyy'))
PARTITION particija_6 VALUES LESS THAN (TO_DATE('01-JUN-1998','dd-MON-yyyy'))
    PARTITION particija_7 VALUES LESS THAN (TO_DATE('01-JUL-1998','dd-MON-yyyy'))
PARTITION particija_8 VALUES LESS THAN (TO_DATE('01-AUG-1998','dd-MON-yyyy'))
    PARTITION particija_9 VALUES LESS THAN (TO_DATE('01-SEP-1998','dd-MON-yyyy'))
PARTITION particija_10 VALUES LESS THAN (TO_DATE('01-OCT-1998','dd-MON-yyyy'))
PARTITION particija_11 VALUES LESS THAN (TO_DATE('01-NOV-1998','dd-MON-yyyy'))
PARTITION particija_12 VALUES LESS THAN (TO_DATE('01-DEC-1998','dd-MON-yyyy'))
PARTITION particija_13 VALUES LESS THAN (TO_DATE('01-JAN-2019','dd-MON-yyyy'))
 );
```

Figure 5: Implementation of RANGE partitioning strategy



Source: Author

4. Recommendations for database optimisation

Instructions written in different ways can generate the same task. Given commands use a different amount of computer resources, and it is necessary to write instructions in the form that provides higher work productivity. Technique optimisation requires adequate management of the computer system memory resources. In other words, certain processes should be allowed the use of memory structures in both, optimal quantity and time.

In general, the most common cause of poor performance is the inefficient use of the index. When performing frequent searches by certain columns, indexes should be created over these columns. If a very small or very large index number is defined in the database, finding the data can take longer, and there exists highly probability of the non-utilisation of certain indexes. Therefore, it is necessary to delete indexes that optimiser does not use. Further, it is necessary to be cautious with the mixing of data types, as well as avoiding the operators, such are: <>, ! =, !>, !<, NOT IN, NOT EXISTS, LIKE '% text'.

If there is a need for aggregated data, materialised views I n Oracle databases could be created. In the case of a static data table, storage with minimal use of the memory space is recommended (ie. data warehouse systems). For tables with many updates, it is necessary to reserve a larger space in memory systems. Historical data should be placed on larger partitions and placed on slower disks. Newer data is placed on faster disks and smaller partitions, therefore due to frequent access to the given data, the processing process is accelerated.

If we have not been able to apply some of the techniques explained in an appropriate way or we believe that the execution plan we have received is not optimal, we could use a hint that will affect the generation of a different execution plan because it would force the optimiser to use another access path. (Pažun, 2017).

6. Conclusion

Business users of today's information systems require higher database perfomance. Database properties are 75% to 80% in the function of SQL command quality. High-quality executable instructions enable optimal database performance. This paper covers several optimisation techniques in terms of faster execution of SQL commands and, therefore, more efficient memory management.

Some optimisation techniques in Oracle systems have been described in this paper. Through the optimisation techniques practical recommendations have been contributed, what was the aim. Given application of the optimiser better execution plans could be generated, i.e. command performance could be improved. Improvements are reflected in the speed of execution of given instructions, as well as more efficient CPU engagement, and finally which results higher-quality management of memory systems. The optimisation process positively influences the quality of the database functioning, as well as the entire information system. These improvements increase the productivity and efficiency of the business system as whole.

Literature

1. Boicea, A., Rădulescu, F., Truică, C.O., Urse, L. (2016). Improving Query Performance in Distributed Database. Control Engineering and Applied Informatics. vol. 18, p. 57-64.
2. Fiorillo, C. (2012). *Oracle Database 11gR2 Performance Tuning Cookbook.* Packt publishing.
3. Nyffenegger. (n.d). Oracle SQL hints. Retreived 6. 5. 2019. from http://www.adp-gmbh.ch/ora/SQL/hints/index.html
4. Oracle. (n.d.). Introduction to the optimizer. Retreived 5. 5. 2019. from http://docs.oracle.com/cd/B10500_01/server.920/a96533/optimops.htm#51003
5. Microsoft. (n.d.). Kreiranje i upotreba indeksa radi poboljšanja performansi. Retreived 1.5.2019. from http://office.microsoft.com/sr-latn-cs/access-help/kreiranje-i-upotreba-indeksa-radi-poboljsanja-performansi-HA010341594.aspx
6. Pažun, B. (2017). Preporuke za optimizaciju baza podataka. Serbian Journal of Engineering Management, ISSN 2466-4693, vol. 2, no. 1, str. 46-53.