Testing of Oracle database utilization

JAN PANUŠ, JOSEF PIRKL
Institute of System Engineering and Informatics
Faculty of economics and administration
University of Pardubice
Studentska 84, Pardubice 53210, Czech republic
CZECH REPUBLIC
jan.panus@upce.cz, jpirkl@centrum.cz http://www.upce.cz

Abstract: - This paper deals with the influence of the Oracle Optimizer hints (sql hints) usage on the query execution. It is the part of parallel computing for data storage as the result of parallel computing. We test the comparison of the classic SQL queries execution towards SQL queries which use Oracle hints. It also examines how the efficiency is influenced by the presence of indices when using hints, how the hints behave when connecting several tables in a query and how the hints are influenced by the chosen Oracle optimizer mode. The CPU time is the basic criterion for the assessment of the efficiency.

Key-Words: - database, optimizer hints, SQL statement.

1 Introduction

Oracle's hints are special comments embedded in SQL queries that can affect the way of processing particular query significantly. Optimizer doesn't need to identify the optimal plan for the query on creating a query plan for various reasons. In this case, it is possible to modify the implementation plan of the hint in such a way that the result will be more adequately of the developer expectations[9]. One of the advantages of hint policy is that changes are affected on level of used queries only – behavior of other queries is default. This problem is usable for many types of problems such as parallel computing [11] or management decision [10].

This paper is divided into two parts. The first part shows the theoretical base for using of Oracle's hint, the optimizer modality, and methods of joining tables. We will show how to use the hint and we will divide hints into groups, we will describe the most documented hints.

We will show testing of hints in the second part of the paper. We will compare queries implemented by default way with queries implemented by Oracle's hints. The query cost and CPU of query time are criteria that we will evaluate in this paper.

It is very important to say that the way how hint affects the database query may be dependent on many factors. Hint should be used only if the developer knows exactly what data he can expect. We can expect improved performance of SQL query if we use the hint by properly using. But we can expect worse results if we perform hint in inappropriate way.

Aim of the paper is to describe the basic structure and using of Oracle hints and their role for influencing of activity optimizer. We tested whether optimizer suggested implementing plans optimally in given situations or whether it is better to modify the query using hints to have more positive results of SQL queries.

2 Oracle database

Oracle is one of the most advanced database systems that support object-relational data storage (ORDBMS). Relational model dates from 1970, when his appearance E. F. Codd published in the journal Communications of the ACM [1].

Oracle uses Structured Query Language (SQL) for its own access and manipulation of data which was developed by IBM based on Codd's suggestion of relational model. SQL was used for the first time in 1979 commercially in 1979[1].

It is proved several steps of query processing at the moment user performing of SQL query that returns specific set of data. It is transformation of SQL query to internal implementation plan. The result is required data[1]. Database optimizer that is responsible for determining of implementation plan of data access is involved in the query processing.

Oracle optimizer activity is described in the next part of this paper part.

3 SQL Query Processing

The challenge of the Oracle optimizer is to optimize SQL query to obtain target output data set as soon as possible [3].

Following methods are used depending on the Oracle optimizer mode [3]: statistics, histograms and hints.

Earlier versions of Oracle (prior to version 10g) provided a rule-based optimizer (RBO). There is option to use that old mode in version 10g but the current modes of optimizer are cost-based (CBO). It means that the plan require the lowest possible cost.

The actual processing of Oracle SQL query is shown on Fig. 1. Parser makes syntactic and semantic check within query processing. Optimizer (starting from Oracle 10g version) selects the methods of CBO for that can be the current statistic very useful. On the basis of existing information is compiled detailed query plan which represents a tree structure describing the different phases which define access methods to the tables and order of joining of tables. The outcome data set is returned after execution of the query.

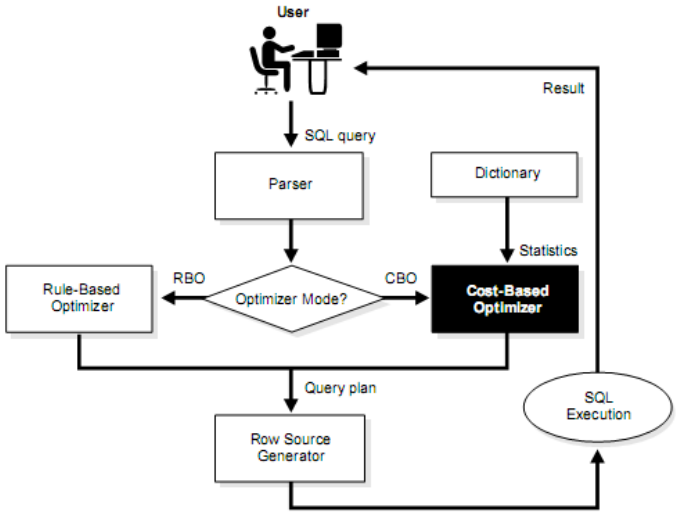

Figure 1: Oracle SQL query processing - source [2]

4 Oracle optimizer activity

Execution plan mapping way to data is the output from the optimizer activity. Optimizer internally proposes several ways and then chooses the optimal one [4].

If the query contains a WHERE section with individual conditions than each of this condition is evaluated by value counted with usefulness of indexes, availability of statistics returning the better information about stored data in the columns of the tables. Optimizer proposes such an implementation plan that has the lowest assessed value on the first place [4].

4.1 Optimizer modes

The parameter OPTIMIZER_MODE specifies mode setting of the optimizer. Available options (for 10g version) are as follows:

4.1.1. ALL_ROWS

The optimizer optimizes the query in this mode. The approach is based on the cost of query [4]. This approach is usually used in OLAP systems where data are handled procedurally in batches and it is no need to have the first line returned as quickly as possible. Optimization of query output obtains the last row of the query as quickly as possible in this case [3].

4.1.2. FIRST_ROWS_n

This mode uses statistics and costs to optimize such that the first *n* output rows of a query are returned [4]. This optimization method is usable in interactive environments, OLTP systems where the user go through the selected rows – it is important the fastest possible displaying of the first line in this case [3].

4.1.3. RULE

The mode is set by the rules – it is not set by estimating of the query costs. This mode is maintained only for backward compatibility. The mode can be changed for each running instance of Oracle but it is possible to change the mode of the optimizer for each query using by the hint [4].

5 Important hints

We will introduce some important hints with its description now.

5.1 ALL_ROWS

This hint performs switching of the optimizer (parameter OPTIMIZER_MODE) to the ALL_ROWS mode for obtaining the most powerful query [4].

5.2 APPEND

This hint sets APPEND mode for INSERT command. New data are added to the table if there are no free allocated blocks in the table [6].

5.3 DRIVING_SITE

Query is process in specific location in case of distributed processing. This site is in the place where the table is located and the table is set in the parameter of this hint. Rows of other tables are replaced to this location for next utilization. [6].

5.4 FIRST_ROWS

Th hint performs switching of optimizer to FIRST_ROW mode for displaying first *n* rows as fast as possible (*n* is parameter of this hint). This hint is ignored in case of performing query for all rows, e.g. using GROUP BY clause or DISTINCT operator. [6].

5.5 FULL

The FULL hint explicitly chooses a full table scan for the specified table. Retrieving data is implemented by checking whole table. Considerable disadvantage of this approach is that table can contain significant amount deleted results that were deleted by DELETE TABLE command. Solution for this case is using TRUNCATE TABLE command. [6].

5.6 INDEX

The INDEX hint explicitly chooses an index scan for the specified table. Optimizer chooses the most cost profitable index in case of using more indexes [6].

5.7 NOAPPEND

This hint enables conventional INSERT by disabling parallel mode for the duration of the INSERT statement. [6].

6 Testing

This part of the paper is focused on testing some methods for accessing of tables and the effect that cause the hints on application approach.

We will monitor CPU of the query and query cost. Cost of query is analyzed by the value obtained by explain plan. Query cost is the value that represents estimation resources. This value includes I/O operations, CPU cycles and memory operations. The higher is this value the higher expected resource we need [4]. It is possible to compare relatively two queries using the criteria of the query cost. The result is the same output dataset.

CPU of the query is pursued criterion for testing of query execution duration. This time interval is expressed in seconds of CPU time within the testing. Time expressed as follows is not affected by waiting for system resources occupation. A large difference between CPU time and the actual elapsed time suggests frequent waiting for the allocation of system resources [6]. We will use our own SQL script for recording the CPU time of implementing of the query. The script will make 1000 calls of the query.

We will test changes of costs and changes of CPU of the query for each version of the query when two tables will be merge. There are some prerequisites for the measurements. Both tables have an index on merging field. no statistic on tables and using the hint of LEADING (a,b) or LEADING (b,a) can be affected order of merging tables. We will use such order that will return lower costs of the query.

For merging of tables we will test models 1:1 and 1: N. In practice this means that for 1:1 model one sentence is connected in the second table, for 1:N model there is connection to the none, one or more sentences.

We will implement collection of all sentences and LIKE selection returning a few sentences within the scope of each model. We will implement the query within scope of each model and specific selection and we will measure cost query and CPU of the query for all three types of merging tables. We will compare these values with default value by default plan.

6.1 1:1 model - all sentences

We will test default implicit plan in this section. Oracle optimizer will choose hash table (Table 1) for Query 1

```
select a.item, b.item_name
from
ba.sale_i a, b ba.item_i
WHERE
b.item = a.item
```

Query 1: 1:1 model, all sentences and default plan

Table 1: Implementation Plan for Query 1

Id	Operation	Rows	Bytes	Cost (%CPU)	Time	
0	Select statement	39219	2527K	204 (2)	00:00:03	
1	Hash join	39219	2527K	204 (2)	00:00:03	
2	Index fast full scan	39218	651K	34 (0)	00:00:01	
3	Table access full	25784	1233K	38 (3)	00:00:01	

Costs of default implementing plan are 204. CPU of query that we measured shows Fig. 1.

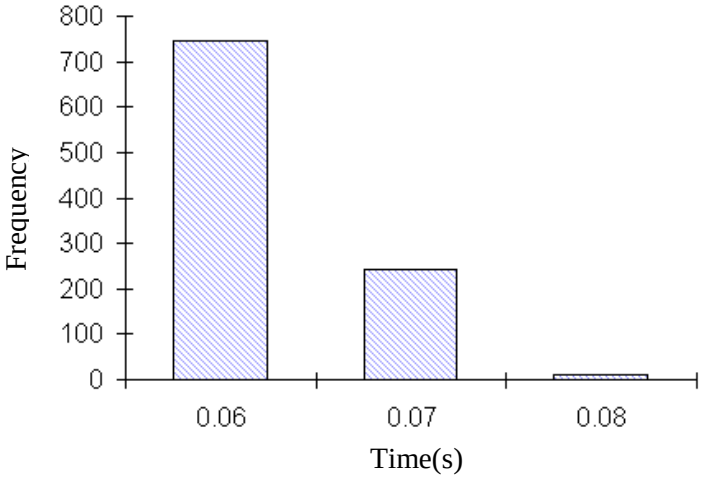

Figure 1: Histogram of measured CPU for query 1 (1000 iterations)

We calculated followed values from obtained measurements:
- average time – 0.0626 seconds,
- standard deviation – 0.0046 seconds.

6.2 1:1 model – LIKE selection

The 1:1 model was performed with LIKE operator selection with specific number of rows (6179 rows). Specific row with name of the item (sold commodity) was traced to each of them. We will implement default plan. Oracle will choose hash table (Table 2) for Query 2.

```
Select a.item, b.item_name
from
ba.sale_i a, b ba.item_i
WHERE
a.item LIKE 'S%' and
b.item = a.item
```

Query 2: 1:1 model, LIKE selection and default plan

Table 2: Implementation Plan for Query2

Id	Operation	Rows	Bytes	Cost (%CPU)	Time
0	Select statement	6435	414K	71 (3)	00:00:01
1	Hash join	6435	414K	71 (3)	00:00:01
2	Index range scan	6436	106K	32 (0)	00:00:01
3	Table access full	5339	255K	38 (3)	00:00:01

The value of costs of default implementing plan are 71. Acquired CPU of query for 1000 iterations shows Fig. 2.

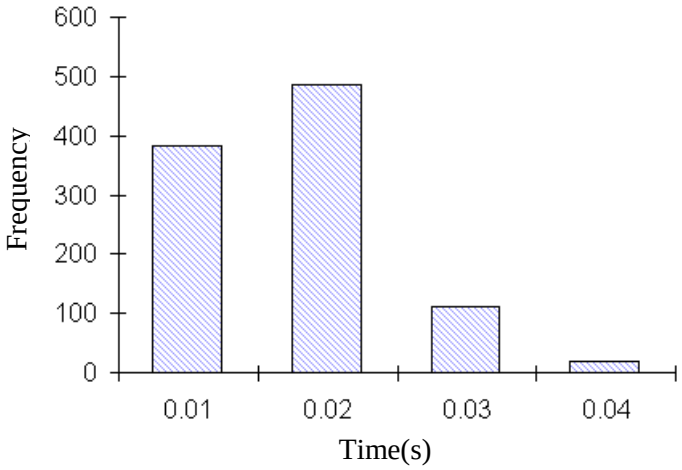

Figure 2: Histogram of measured CPU for query 2 (1000 iterations)

6.3 Model 1: N - all sentences

Corresponding items of sales are traced for all rows in model 1:N. Structure is 1:N and number of records (cardinality) ranges from 0 - 650 records. The average number of records within used tables is 1.59 records. Oracle will choose hash table (Table 3) for Query 3.

```
Select a.item, b.period, b.date_, b.price
ba.item_i from a, b ba.sale_i
WHERE
b.item = a.item
```

Query 3: Model 1:N, all the sentences, the default plan (nested merge)

Table 3: Implementation Plan for query 3

Id	Operation	Rows	Bytes	Cost (%CPU)	Time
0	Select statement	39219	2336K	81 (19)	00:00:01
1	Nested loops	39219	2336K	81 (193)	00:00:01
2	Table access full	39218	1685K	68 (3)	00:00:01
3	Index unique scan	1	17K	0 (0)	00:00:01

The value of costs of default implementing plan are 81. Acquired CPU of query for 1000 iterations shows Fig. 3.

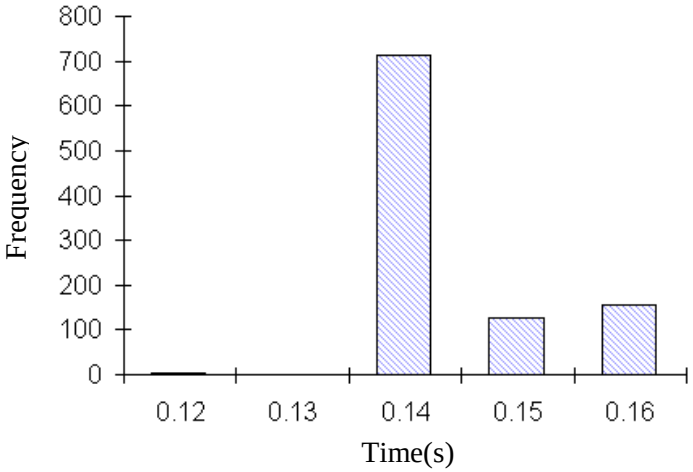

Figure 3: Histogram of measured CPU for query 3 (1000 iterations)

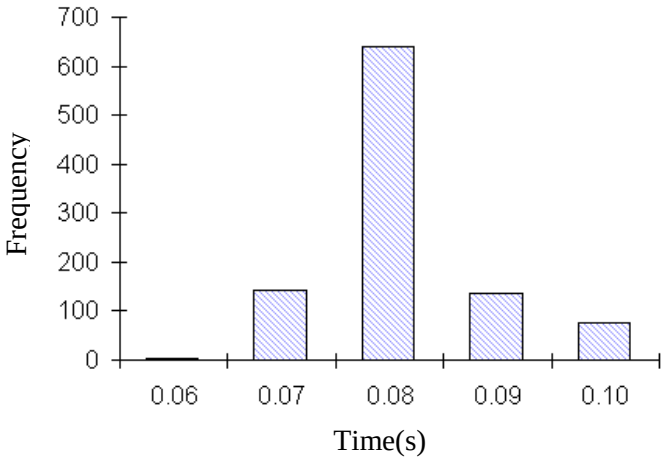

Figure 4: Histogram of the measured CPU request 1927 (1000 measurements)

We calculated followed values from obtained measurements:
- average time – 0.0176 seconds,
- standard deviation – 0.0072 seconds.

6.4 Model 1: N - all sentences, forced hash merge by hint

Forced hash merge is used for table merge (query 4). Obtained detailed query plan table describes Table 4.

*select / * + USE_HASH (a, b) * / a.item, b.period, b.date_, b.price*
ba.item_i from a, b ba.sale_i
WHERE
b.item = a.item
Query 4: Model 1: N, all sentences, forced hash merge

Table 36: Implementation Plan for the 27 inquiry

Id	Operation	Rows	Bytes	Cost (%CPU)	Time
0	Select statement	39219	2336K	91 (5)	00:00 :02
1	Nested loops	39219	2336K	91 (5)	00:00 :02
2	Index fast full scan	25784	428K	21 (0)	00:00 :01
3	Table access full	39218	1685K	68 (3)	00:00 :01

The value of costs of forced hash merge is 91. Acquired CPU of query for 1000 iterations shows Fig. 4.

We calculated followed values from obtained measurements:
- average time – 0.0814 seconds,
- standard deviation – 0.0076 seconds.

7 Statistical results

We made comparison with some statistical methods to discover the strength of results measured in our research. We used test of variance, robust test of variance, student's t-test for the same variance and student's t-test for different variance and two samples Kolmogorov-Smirnov test. We also used some basic statistical methods like average, standard deviation etc. We made these tests for two types of database models – 1:1 and 1:N. For 1:N model we measured two samples data; one for query with all rows and one for query with LIKE operator. Results are shown below.

Model 1:1 – comparison of two statistical samples "A2_ALL - C2_ALL"

Significance level:	0,05	
Compared rows:	A	B
Amount of data:	1000	1000
Average:	0,06266	0,12192
Standard deviation:	0,004663276332	0,007705385972
Variance:	2,174614615E^{-005}	5,937297297E^{-005}

Test of variance

Rate variance:	2,730275635	
Degrees of freedom:	999	999
Critical value:	1,10696448	
Resume:	Variances are different	
Probability:	6,963020856E^{-055}	

Robust test of variance

Rate variance:	2,730275635	
Reduced degrees of freedom:	405	405
Critical value:	1,17317294	
Resume:	Variances are different	
Probability:	$1,929369173E^{-023}$	

Student's t-test for the same variance

t-statistics :	208,0654807
Degrees of freedom:	1998
Critical value:	1,961152015
Resume:	Averages are different
Probability:	0

Student's t-test for the different variance

t-statistics:	208,0654807
Reduced degrees of freedom:	1644
Critical value:	1,961408017
Resume:	Averages are different
Probability:	0

Two-sample Kolmogorov-Smirnov test

Difference DF:	1
Critical value:	0,06073614619
Resume:	Probability distributions are different

Model 1:N – comparison of two statistical samples "A1_ALL – C1_ALL"

Significance level:	0,05	
Compared rows:	A	B
Amount of data:	1000	1000
Average:	0,14434	0,08141
Standard deviation:	0,007564497447	0,007640043866
Variance:	$5,722162162E^{-005}$	$5,837027027E^{-005}$

Test of variance

Rate variance:	1,020073682	
Degrees of freedom:	999	999
Critical value:	1,10696448	
Resume:	Variances are same	
Probability:	0,347421028	

Robust test of variance

Rate variance:	1,020073682	
Reduced degrees of freedom:	327	327
Critical value:	1,194575787	
Resume:	Variances are same	
Probability:	0,3953649044	

Student's t-test for the same variance

t-statistics:	185,0947266
Degrees of freedom:	1998
Critical value:	1,961152015
Resume:	Averages are different
Probability:	0

Student's t-test for the different variance

t-statistics:	185,0947266
Reduced degrees of freedom:	1998
Critical value:	1,961152015
Resume:	Averages are different
Probability:	0

Two-sample Kolmogorov-Smirnov test

Difference DF:	1
Critical value:	0,06073614619
Resume:	Probability distributions are different

Model 1:N – comparison of two statistical samples "A1_LIKE – B1_LIKE"

Significance level:	0,05	
Compared rows:	A	B
Amount of data:	1000	1000
Average:	0,03879	0,03471
Standard deviation:	0,008804170229	0,007427723142
Variance:	$7,751341341E^{-005}$	$5,517107107E^{-005}$

Test of variance

Rate variance:	1,404964811	
Degrees of freedom:	999	999
Critical value:	1,10696448	
Resume:	Variances are different	
Probability:	$3,854542167E^{-008}$	

Robust test of variance

Rate variance:	1,404964811	
Reduced degrees of freedom:	712	712
Critical value:	1,127942812	
Resume:	Variances are different	
Probability:	$2,799310359E^{-006}$	

Student's t-test for the same variance

t-statistics:	11,20082707
Degrees of freedom:	1998
Critical value:	1,961152015
Resume:	Averages are different
Probability:	$2,758142417E^{-028}$

Student's t-test for the different variance

t-statistics:	11,20082707
Reduced degrees of freedom:	1943
Critical value:	1,961185665
Resume:	Averages are different
Probability:	$2{,}905839518E^{-028}$

Two-sample Kolmogorov-Smirnov test

Difference DF:	0,225
Critical value:	0,06073614619
Resume:	Probability distributions are different

Model 1:N – comparison of two statistical samples "A1_ALL – C1_ALL"

Significance level:	0,05	
Compared rows:	A	B
Amount of data:	1000	1000
Average:	0,06266	0,0971
Směr. odchylka :	0,004663276332	0,007787852763
Standard deviation:	$2{,}174614615E^{-005}$	$6{,}065065065E^{-005}$

Test of variance

Rate variance:	2,789029847	
Degrees of freedom:	999	999
Critical value:	1,10696448	
Resume:	Variances are different	
Probability:	$4{,}715708486E^{-057}$	

Robust test of variance

Rate variance:	2,789029847	
Reduced degrees of freedom:	494	494
Critical value:	1,155555491	
Resume:	Variances are different	
Probability:	$3{,}050957779E^{-029}$	

Student's t-test for the same variance

t-statistics:	119,9797571
Degrees of freedom:	1998
Kritická hodnota:	1,961152015
Resume:	Averages are different
Probability:	0

Student's t-test for the different variance

t-statistics:	119,9797571
Reduced degrees of freedom:	1634
Critical value:	1,961416861
Resume:	Averages are different
Probability:	0

Two-sample Kolmogorov-Smirnov test

Difference DF:	1
Critical value:	0,06073614619
Resume:	Probability distributions are different

8 Conclusion

It is possible to significantly impress form of implementation plan proposed by Oracle optimizer. Oracle provides several types of hints which are able to work with really dynamically. We can change chosen mode of optimizer, chosen access method to the table or we can change the order and the way of merging two or more tables.

We can say that optimizer chose default proposed implementing plan really properly at most cases. This plan seemed to be optimal in comparison with other options. These options were forced by using of hints. Optimality was measured by using two criterions: query cost and CPU of query.

We made statistical comparison of two samples for 1:N model. The result is that samples are different both from test of variance and Student's t-test point of view. We also used robust tests of variance and two samples Kolmogorov-Smirnov test.

References:

[1] Alapati, S. *Expert Oracle Database 10g Admistration,* APress, Berkeley, 2005. 1305 s. ISBN 1-59059-451-7.

[2] Čech, P., Bureš, V.: Advanced Technologies in e-Tourism, The 9th WSEAS International Conference on Applied Computer Science, Genova, WSEAS Press, pp.85-92, ISBN 978-960-474-127-4 (2009)

[3] Cyran, M. Oracle 8i - *Designing and Tuning for Performance*, Release 2 (8.1.6), Redwood City, Oracle Corporation, 1999. 646 s. A76992-01.

[4] Foot, Ch. 10g *Optimizer Statistics Gathering* [online]. DBAZine.com, 2006. [2010-08-20]. Online: http://www.dbazine.com/blogs/blog-cf/chrisfoot/blogentry.2006-02-11.0939245208

[5] Chan, I. *Oracle Database Performance Tuning Guide*, 10g Release 2 (10.2). 1st ed., Redwood City, Oracle Corporation, 2005. 474 s.

[6] Kyte, T. *Expert One-on-One Oracle*. Berkeley: Apress, 2003. 1544 s.ISBN 1590592433.

[7] Lorentz, D. *Oracle Database SQL Reference*, 10g Release 2 (10.2), Oracle, 2005. 1428 s. B14200-02.

[8] Panuš, J., Šimonová, S.: Measurability of evaluative criteria used by optimization methods. In: Proceedings of Conference on Computional, Intelligence, Man-Machine Systems and Cybernetics. 1st edition. Dallas, Texas: Wseas Press. pp. 451-455. ISBN 960-8457-55-6. (2007)

[9] Jirava, P., Krupka, J.Information System Classification. In: Proceedings of the 8th WSEAS International Conference on Systems Theory and Scientific Computation (ISTAC'08). Rhodes, Greece: Wseas Press. 2008. pp 94-98. ISBN 978-960-6766-96-1.

[10] Šimonová, S., Skopečková, H: Applicability of Six Sigma methodology for services output evaluation. In *Applied Computer Science*. International Conference on APPLIED COMPUTER SCIENCE (ACS), 15-17. 9. 2010, Institute for Environment, Engineering, Economics and Applied Mathematics, Sliema, Malta. Malta: WSEAS Press, 2010. ISBN 978-960-474-225-7, ISSN 1792-4863, p. 297 – 302.

[11] Rebai, M., Kacem, I., Adjallah, K. H.: Scheduling Jobs and Preventive Maintenance Activities on Parallel Machines. In *Selected topics in Applied Computer Science*. International Conference on APPLIED COMPUTER SCIENCE (ACS '10), 4-6. 10. 2010, Iwate Prefectural University, Japan, Japan: WSEAS Press, 2010. ISBN 978-960-474-231-8, ISSN 1792-4863, pp. 406 – 411.

Acknowledgement:
This paper was supported by grant P403/10/P334 of Czech science foundation**.**