U.P.B. Sci. Bull., Series C, Vol. 86, Iss. 4, 2024 ISSN 2286-3540

A CASE STUDY ON DBMS STABILITY PERFORMANCE EVALUATION

Daniel-Călin POPEANGĂ[1], Mariana-Ionela MOCANU[2], Alexandru BOICEA[3], Florin RĂDULESCU[4], Sorin-Nicolae CIOLOFAN[5]

The actual Database Management Systems (DBMS) contain significant technologies and elaborate mechanisms that sustain a high level of processing capacities and reduced response times, among multiple possibilities of high amount of data storage. Our present study is focused on another important indicator of a DBMS good performance, the stability during normal operation, as a part of a methodology we developed to evaluate relational DBMS's stability performance during system runtime. We run over 15.900.000 single queries on the tested database, to analyze how the system performs in terms of stability and determine the optimal workload of a tested system for a most stable answer. The results of the studied phase can provide important information for a running system database administrator, in terms of performance stability.

Keywords: database, DBMS, stability, query, performance, evaluation, runtime

1. Introduction

The DBMS domain significantly evolved, in the last decades, proportionally with the technology evolution [1]. The databases vendors continuously improved their capacities to manage the permanent requirement to increase the data volumes of storage capabilities. The database and DBMS have become an integral part of every kind of work, whether in managing business-related data or managing our household accounts [2]. This evolution generated a natural need to evaluate the different DBMS performances on multiple levels, starting from the elementary indexes such as query response time and the query response cost [3], evolving to the new areas, such as cloud computing and NoSQL databases performances.

[1] Assist., Computer Science Dept., National University of Science and Technology POLITEHNICA of Bucharest, Romania, daniel.popeanga@upb.ro
[2] Prof., Computer Science Dept., National University of Science and Technology POLITEHNICA of Bucharest, Romania, mariana.mocanu@upb.ro
[3] Prof., Computer Science Dept., National University of Science and Technology POLITEHNICA of Bucharest, Romania, alexandru.boicea@upb.ro
[4] Prof., Computer Science Dept., National University of Science and Technology POLITEHNICA of Bucharest, Romania, florin.radulescu@upb.ro
[5] Prof., Computer Science Dept., National University of Science and Technology POLITEHNICA of Bucharest, Romania, sorin.ciolofan@upb.ro

In our previous research activities, we've analyzed the DBMS performance in various areas of interest, starting from the influence brought by the transaction execution plan in system performances [4], continuing with various techniques of query optimization [3] and the general framework of database performance evaluation stated by the Transaction Performance Council [5], [9], [10]. The continuous effort on DBMS performance evaluation provided some significant answers in a comparison of different types of database systems, such as Oracle, Microsoft SQL Server, MySQL.

Our studies on the TPC benchmarking systems evolved into multiple directions, evaluating various types of DBMS performances [6], [11], producing performance indexes used to make a comparison between different systems. These indexes provide significant and transparent comparison between different significant DBMS, expressing both the transaction performance and the average cost per transaction comparison.

Another main goal of our DBMS performance evaluation efforts was dedicated to a comparison between different database systems performances and attributes, including comparison between No-SQL and SQL [4].

In our current studies, we proposed a methodology, called C-SGBD, to analyze the DBMS performance in a different area of interest: the stability of a relational database system, during a normal runtime execution of the database server [8]. C-SGBD is an evaluation methodology composed by the evaluation method and experimental algorithms, involving column-based text fetching procedures, word extraction, term weight calculation, multiple queries posed to the DBMS.

This paper presents the experimental activity involved in the database querying phase of the stability evaluation procedure. Our proposed goals is to reach a conclusion, based on the experimental results, related to the number of queries that must be run on the database within the methodology in order to reach a stable performance.

2. C-SGBD Methodology

C-SGBD contains a stability performance evaluation method for a real runtime DBMS. It is based on a database populated with real records, generated in a normal runtime evolution of a real website/on-line application [8]. It is mostly dedicated to analyzing the performance stability of on-line systems that provide public hosting and database services. The performance analysis takes into consideration the influence of the specific moment when the performance was recorded to the measured values. The method consists of five main phases: `

1. Identifying the target tables from the system database tables that contain the main text records of the system (**ST** - Selected Tables)

2. Words extraction from text columns of the ST and weight calculus for every distinct word. The **word weight** counts the number of the word appearances within the text columns in the ST.
3. Multiple sets of database queries, each of one consisting of **N** queries (Queries Set - **QS**) of an equivalent of medium word search weight, are posed to the database, during the system normal operation. The total response time for every run set is registered.
4. A quantitative analysis is conducted on the influence of the test runtime moment of the day to the recorded response time of every QS. Several time slots are defined as references within a complete day. **A first DBMS performance index** is obtained, based on the time slots variations applied to the recorded response time of QS.
5. The **global stability index** (**SI** – Stability Index) is obtained based on the time slots medium values of the QS results.

C-SGBD produces two different indexes, both of them containing the contribution of the different time slot influence posed to the QS results:

- The DBMS weighed medium response time for a query. The lower the response time is, the higher is the DBMS performance
- SI, as an indicator for DBMS performance stability, Also, the lower the stability index is, the higher is the DBMS performance, indicating an increase in terms of system performance of stability.

Two different DBMS can be compared basing this methodology by both indexes

3. Experiment on the influence of the number of QS on the time slots values

3.1 The experimental base

In this paper, we will present an experiment run to analyze the system response to multiple different number of queries, to obtain the optimal number of the QS that must be run in the third phase, reaching, as much as possible, stable and consistent indexes in the C-SGBD methodology. In the proposed methodology, the minimum number of QS generated and executed by the DBMS is set to 200 queries and this number will be increased, gradually, and the influence of the number of QS in the performance stability will be evaluated in order to obtain the minimum number Noptimal of queries that can provide a consistent performance to the third phase of C-SGBD (Noptimal >=200).

The queries must be run at different moments, as much as possible in a evenly distributed way of daytime. An individual QS consists in N queries of different types:

➢ SELECTS (e.g.: "SELECT BOOK_ID, TITLE, DESCRIPTION FROM BOOKS WHERE DESCRIPTION LIKE '%$cuv_crt%'") on the **ST** tables, that emulate normal user search queries on the portal text tables. The SELECT operations simulates normal searches that regular users are posing to the system, using the application search tools. The search tools are generally used to retrieve information from the portal on the users' topics of interest.

➢ AGGREGATE queries (e. g.: "SELECT COUNT (*) FROM OFFICIAL_JOURNAL WHERE JOURNAL_CONTENT LIKE '%$cuv_crt%'") such as SUM and COUNT operations, also regularly used in a normal on-line application operation. These requests simulate regular application information provided by the application to the user as feedback to different requests (e.g no of items retrieved by a search, no of elements found in a search, sum of selected item prices).

➢ UNION operations (e.g.: "SELECT TITLE, BOOK_ID FROM BOOKS WHERE DESCRIPTION LIKE %$cuv_crt%' UNION SELECT TITLE, NEWS_ID FROM NEWS WHERE NEWS_CONTENT LIKE '%$cuv_crt%'") between different types of data sources, that integrate multiples records from various sources within one result.

3.2 The experimental procedure

The experimental procedure consists in running multiple QS on the database system and measuring the response times recorded for finalizing the complete sets of queries. The number of queries N is increased from 200 to higher values and the response times are analyzed within the same number of queries recorded.

The experiment consists of measuring the total execution time for multiple operations made on database, including random choices of the exact data tables on every QS run. The C-SGBD procedure requests that every query should be generated based on a specific search complexity within the column texts of the text tables, named word weight.

In the analyzed system, on the Romanian running legal website DreptOnline.ro, the identified data tables are:

NEWS (2,381,679 individual selection terms), QUESTIONS (51,803 individual selection terms), BOOKS (113,178 individual selection terms), JOURNAL (3,837,220 individual selection terms), ANSWERS (8,128,808 individual selection terms) and the generated terms weight table, called WEIGHTS

Every QS is generated according to the following procedure:

➢ *The number of queries N is set. The number of SELECT operations is set, the number of GROUP BY operations is set, the number of UNION operations is set (e.g. 45% +45%+10%)*
➢ *The timer is started T1*

- *For every step I from 1 to N*
    - *The average* ***word weight*** *is selected based on the data generated by the previous phases of C-SGBD*
    - *The text table is randomly selected*
    - *The query is generated based on:*
        - *The selected text table*
        - *The type of operation*
        - *The selected word within the average calculated word weight*
    - *The query is run*
- *The timer is stopped, T2 and the QS total execution time is calculated: ET=T2-T1*
- *The result is recorded*

The QS generating procedure, and the ET calculation are described in Fig. 1.

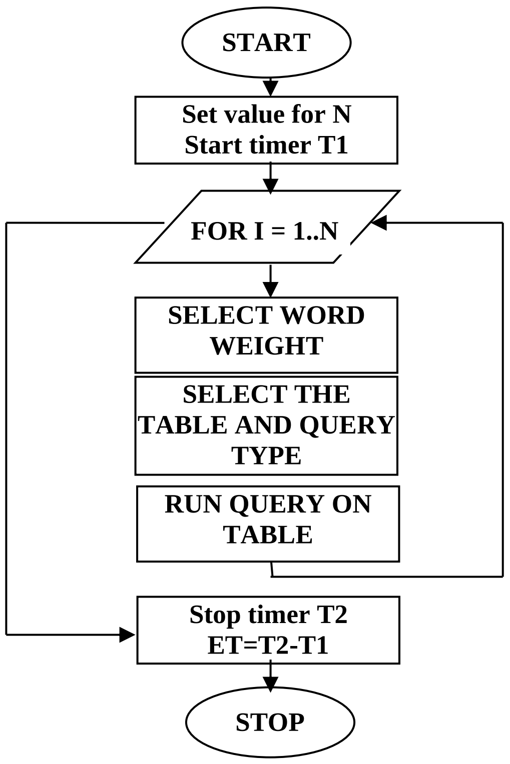


Fig. 1: Generating A Query Set, and the ET recording

For every significant value of N (starting from 200) the experiment consists of over NoIt (number of iterations) =1000 set of queries, in order to obtain a significant behavior of the system.

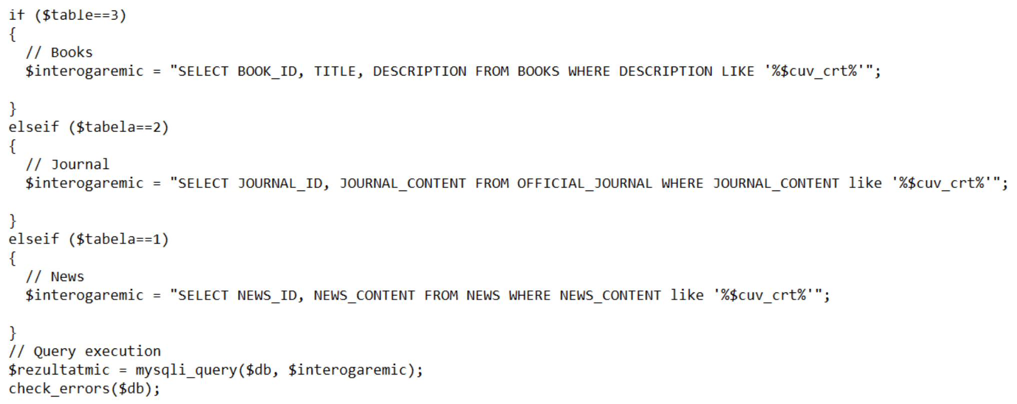
The total number of single queries run can be obtained from the formula:

$$\boldsymbol{TQ = NoIt * \sum_{N=200}^{1800} N\,, where\ NoIt \geq 1000} \qquad (1)$$

Using formula (1), TQ = 15.900.000 individual queries run on the server.

Therefore, we run over **15.900.000** single queries on the tested database to obtain a conclusive behavior of the system.

Every individual query is run on the DBMS using a PHP application code, exemplified in Fig. 2.



```
if ($table==3)
{
  // Books
  $interogaremic = "SELECT BOOK_ID, TITLE, DESCRIPTION FROM BOOKS WHERE DESCRIPTION LIKE '%$cuv_crt%'";

}
elseif ($tabela==2)
{
  // Journal
  $interogaremic = "SELECT JOURNAL_ID, JOURNAL_CONTENT FROM OFFICIAL_JOURNAL WHERE JOURNAL_CONTENT like '%$cuv_crt%'";

}
elseif ($tabela==1)
{
  // News
  $interogaremic = "SELECT NEWS_ID, NEWS_CONTENT FROM NEWS WHERE NEWS_CONTENT like '%$cuv_crt%'";

}
// Query execution
$rezultatmic = mysqli_query($db, $interogaremic);
check_errors($db);
```

Fig. 2: PHP code example for individual query execution

3.3 The Server details

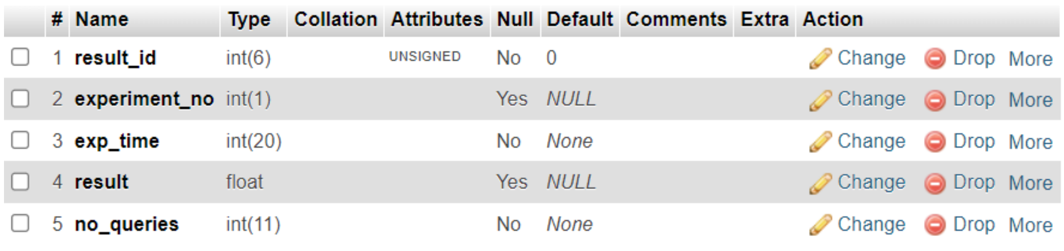
C-SGBD methodology is designed to measure the stability performances for a real operating DBMS during its normal runtime evolution and among the normal users tasks overload. The server used for the main operations was running with the parameters described in Table 1.

Table 1

cPanel Version	110.0 (build 31)
Apache Version	2.4.58
MySQL Version	10.3.27-MariaDB
Architecture	x86_64
Operating System	Linux
Kernel Version	4.18.0-348.23.1.lve.el7h.x86_64
Perl Version	5.16.3

4. Interpreting the results

The result for every QS is recorded into the RESULTS table, having the structure presented in Fig. 3.



	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	**result_id**	int(6)		UNSIGNED	No	0			Change Drop More
☐	2	**experiment_no**	int(1)			Yes	*NULL*			Change Drop More
☐	3	**exp_time**	int(20)			No	*None*			Change Drop More
☐	4	**result**	float			Yes	*NULL*			Change Drop More
☐	5	**no_queries**	int(11)			No	*None*			Change Drop More

Fig. 3: RESULTS table structure for the third Phase of C-SGBD

The results are analyzed within the same amount of the transactions (N) in order to obtain the global variation of the results and the system stability for every recorded value for N. For this purpose, we’ve calculated the following indicators:

- ➢ The maximum time recorded for running of 100 queries [in seconds]
- ➢ The minimum time recorded for running of 100 queries [in seconds]
- ➢ The average time recorded for running of 100 queries [in seconds]
- ➢ Standard deviation for the QS

The obtained results are presented in Table 2:

Table 2

C-SGBD Algorithm - Phase no 3

No of queries (set)	Min Value (sec for 100Q)	Max Value (sec for 100Q)	Average Value (sec for 100Q)	Standard Deviation
200	3.94462657	5.853013515	4.824274908	0.417957448
300	4.177803675	5.041297277	4.67937941	0.248076217
400	4.254061222	5.427081585	4.668644978	0.271440501
500	4.466841888	5.216475677	4.813941852	0.213201031
600	4.074904442	5.034558932	4.68699095	0.276048424
700	4.26140104	5.063520704	4.705557115	0.183611351
800	4.378765106	4.840855122	4.626365594	0.151185646
900	4.269561768	4.865469615	4.635016069	0.157424179
1000	4.479382324	5.146421051	4.710367508	0.17449132
1200	4.437171618	4.848783493	4.65450355	0.148543125
1300	**4.485152318**	**4.777329372**	**4.656403121**	**0.101873729**

1400	4.426936558	4.926513672	4.630674022	0.148233259
1500	4.282950846	4.862237549	4.636847632	0.128704877
1600	4.380867481	4.947503567	4.651077075	0.144093498
1700	4.431859634	4.929005342	4.671387593	0.121021237
1800	4.486450195	5.014217801	4.654694206	0.133319803

Analyzing the chart data, a significant decrease of the **standard deviation** can be seen from the initial queries series of N starting from 200, to a lowest value of **0.101873729**, reached **for N=1300**, followed by a significant increase of this value again.

It is important to evaluate the maximum decrease of the standard deviation and the point from where its values start to increase again consistently.

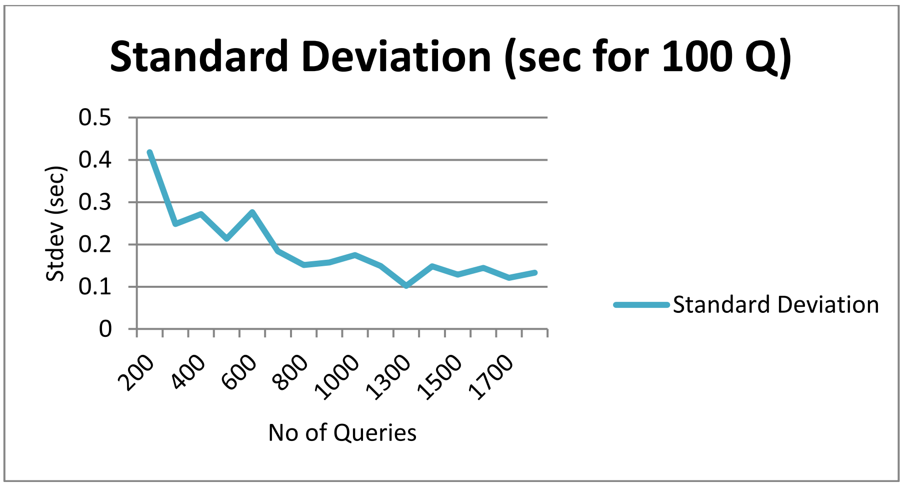


Fig 4: Standard deviation of the total DBMS response time for N queries

Considering C-SGBD methodology is searching for stability indicators of the system's response and behavior, the N value corresponding to the lowest standard deviation is indicating the most stable performances of the DBMS.

We can, therefore, conclude that, for N=1300 queries, the system response is optimal, from the response time stability point of view. For lower values of N, the system response was significantly lower as performance stability. For higher values of N, the system response was significantly more stable, but the stability performance followed a slow decrease in the standard deviation levels. The identified value is very important, because C-SGBD can use this result in the subsequent phases.

5. Conclusions

In this paper, we've presented the experimental activity involved in the database querying phase of the stability evaluation procedure of the third phase of C-SGBD [8]. The experiment presented consisted in generating multiple sets of N queries, and analyzing the DBMS behavior and stability.

Over 15 million single queries were run to a real runtime operational DBMS, in order to register the query response time and to analyze the way the system performs in terms of stability. We've obtained a value for N=1300 queries (the number of QS) that reached the optimal value for stability, in terms of standard deviation of the response times within the same range of N.

The result will permit us to use this value as a proven indicator in the C-SGBD methodology research activities that will permit us obtain the stability indicator for a DBMS.

The study provides a different approach of the DBMS performance evaluation and can be useful for a runtime system administrator, in the process of assuring a certain level of system stability. The studied procedure, as part of the C-SGBD methodology, can provide a significant starting point in the effort of elaborating adapted benchmarking techniques for a runtime system using a relational DBMS.

An intuitive direction in the future research in this domain can analyze the dependency between the number of the queries included in a set (N) and the DBMS capacity and runtime workload. The procedure can easily be adapted and evolved to detect the proper text columns within a running database to perform the described tests and to provide an automated result.

R E F E R E N C E S

[1] Abdullah, T. and Resul, K., "A performance evaluation of in-memory databases", Journal of King Saud University - Computer and Information Sciences, 2017, pp. 520-525

[2] Kristi L. Berg, Tom Seymour, Richa Goel, "History Of Databases", International Journal of Management & Information Systems — First Quarter 2013 Volume 17, Number 1

[3] Petrescu, M.; Popeangă, D.; Vasilescu R., Performance Evaluation in Databases. Analyses and Experiments, Buletinul Științific din Timișoara, Transactions on Automatic Control and Computer Science, 4th International Conference on Technical Informatics CONTI'2000, Timișoara 2000

[4] Petrescu, M., Popeangă, D., Vasilescu, R, "The TPC-C Database Performance Benchmarking System, Analysis and Experiments", CSCS-13, June 2001, București

[5] Petrescu, M., Popeangă, D., "Relation Between Query Optimization and Execution Plan in Relational Databases", CSCS-14, July 2003, Bucuresti

[6] Raab, F., Kohler and W., Shah A. "Overview of the TPC-C Benchmark. The Order-Entry Benchmark", TPC, [Retrieved September 10, 2020] http://www.tpc.org/tpcc/detail5.asp

[7] Boicea, A., Rădulescu, F. and Agapin, L., "MongoDB vs Oracle - Database Comparison", Third International Conference on Emerging Intelligent Data and Web Technologies, DOI: 10.1109/EIDWT.2012.32, pp. 330-335
[8] Popeangă, D, Boicea A., Rădulescu F., Rădulescu I., and Petrescu M., S., "C-SGBD – A Procedure for DBMS Performance Stability Evaluation", Proceedings of the 36th International Business Information Management Association (IBIMA), ISBN: 978-0-9998551-5-7, 4-5 November 2020, Granada, Spain
[9] TPC-C Most Recently Published Results, Version Results As of 18-Feb-2023 at 12:21 PM, https://www.tpc.org/tpcc/results/tpcc_last_ten_results5.asp
[10] TPC Express Big Bench, TPCx-BB, Standard Specification, https://www.tpc.org/tpc_documents_current_versions/pdf/tpcx-bb_v1.6.0.pdf
[11] TPC-E, https://www.tpc.org/tpce/ accessed February 2023