# DATABASE MANAGMENT SYSTEM SELECTION

**Stanišević Ilja**
*Valjevo Business School of Applied Studies,*
*Valjevo, Serbia*

**Slobodan Obradović**
*Faculty of Electrical Engineering*
*East Sarajevo, Bosnia and Herzegovina*

**Mladjen Vićentić**
*Valjevo Business School of Applied Studies,*
*Valjevo, Serbia*

**Abstract**

*In order to select optimal database managment system for information system developed for Valjevo Business School of Applied Studies, testing of three available options (i.e. MS SQL, MySQL and Access) was performed. Instead of using existing banchmark software providing general evaluation, the development team has built a simple software tool to test features relevant for the system. This paper describes applied methodology and displays realised results.*

**Keywords:** Database management systems, Microsoft SQL Server, MySQL, Access, benchmark, performance evaluation.

## INTRODUCTION

During realization of information system for Valjevo Business School of Applied Studies the question of selection the most adequate database management system has arisen. Members of the development team had various suggestions and proposals depending on their experience. Therefore, it was necessary to develop unique and objective testing criteria and tool to provide an adequate answer.

One way to resolve the issue was to rely on some of already existing benchmark tools. These tools should be relevant, portable, simple and general [1], and they are designed to be applicable to as many cases as possible [2]. The problem with these testing tools is that they take into consideration many different aspects, many of them irrelevant for our purpose [3]. Furthermore, the best evaluated system according to achieved general marks may not be the best one for our aim. Finally, the development team has agreed to develop a new testing tool according to the specific requirements of the school's information system.

The tool and criteria had to be based on the requests and demands characteristic for the particular system. This paper describes usage of the developed software testing tool and it presents the results achieved.

## DETERMINATION OF THE TESTING DOMAIN

The testing system was designed for relational databases since the school information system's database schema was relational [4]. Activities on the targeting information system should be happening in a real time. Therefore, basic evaluation criterium was response speed. According to that, the testing tool should measure response speed on generated queries, select queries (i.e. SELECT SQL command) as well as action queries (i.e. INSERT, UPDATE and DELETE SQL commands). Furthermore, the generated queries could have defined logical condition (i.e. WHERE clause). The user can determine if he wants to measure response speed to queries executed on one or more related tables (i.e. if the queries should contain JOIN clause). Since the number of existing data effects measuring results, tests could be performed with different number of already existing records. The data were generated in a way that 20% of the records meet the logical requirement defined through WHERE clause.

To determine the amount of data for testing, it was necessary to estimate the largest amount of data during transactions as well as the number of rows in the largest tables.

The largest transaction happens in the beginning of a schoolyear when new students

are enrolled and when existing students enroll for a new academic year. Large transactions also happen in an examination period when students register for the exams. Number of records for these transactions seldom exceeds 1.000.

To determine size of the largest table it was necessary to detect the most numerous objects in the database. It was detected by looking at the previous data. The table that contains the largest number of rows was the table containing data about the exam registrations. The annual number of rows for the table can be estimated using the following formula (1):

$$\boldsymbol{RN} = \sum_{i=u,m,d} (\Delta_i * \boldsymbol{\alpha_i} * \boldsymbol{SN_i} * \boldsymbol{EN_i}) \qquad (1)$$

where:

-**i** represents level and kind of study (u for undergraduate study, m for master study, d for distance learning study);

-**RN** represents number of exam registrations;

-**Δ** represents duration of the studies in academic years;

-**α** represents ratio of active students;

-**SN** represents number of students;

-**EN** represent number of examinations during academic year.

The school has accreditation for 165 students on the undergraduate (bachelor) level, 50 students on master level, and 32 students for distant learning studies. Average annual number of exams on undergraduate level and distance learning studies is 14, while on master level is 6. Undergraduate and distance learning studies last three academic years while master studies last two years. Ratio of active students for undergraduate studies is 0.38, for distance learning studies is 0.15 and for master studies is 0.57. Having all these values and including them in the equation (1), we get:

$$RN = RN_u + RN_m + RN_d \qquad (2)$$

where

$$RN_u = 3 * 0.38 * 165 * 14 = 2.633.4$$

$$RN_m = 2 * 0.57 * 50 * 6 = 342$$

$$RN_d = 3 * 0.15 * 32 * 14 = 201.6$$

Entering these numeric values into the equation (2), finally we have:

$$RN = 2633.4 + 342 + 201.6 = 3177 \qquad (3)$$

On the annual level it was estimated that the largest table in the database will have increase of 3,177 rows. For three years it will be 9,531 rows.

Applying similar estimates for other large tables in the database (held lectures records, employee's presence records, students' attendance records...) it was estimated that testing queries execution including tables larger than 10,000 would be irrelevant for our purpose.

On the other hand, majority of tables in the database (like student's register, employee's register, classrooms register, subjects list, list of graduate students...) are not that large. They usually have 10-1,000 rows.

Having previous estimations in mind, it was decided that all measurements would be performed on tables having 10, 100, 250, 500, 1000, 2500, 5000, 7500 and 10000 rows. This range would cover all the cases presented in the real database.

Three most common database management systems [5] were selected to be tested: Microsoft SQL Server (version 2016, Standard, 64-bit); MySQL (version 8.0.16. MySQL Community Server GPL) and Microsoft Access (version Professional Plus 2016.).

MS SQL Server is the most popular RDBMS in organizations, while MySQL is the third most popular [6]. In overall usage MySQL is the second most widespread DBMS, and MS SQL Server is the third (the first one is Oracle DB) [7]. It can be told that MS SQL Server dominates for business purposes within organizations while MySQL is leading for internet applications supported by Linux (and other UNIX-like operating systems), Apache server and PHP programming language. Important factor in MySQL popularity for internet applications is the fact that it is used by WordPress, the most popular tool for WEB page development. Despite many limitations, Microsoft Access is still the fifth most popular database [7]. The fact that Access is a part of Microsoft Office significantly contributes its popularity.

Selection of DBMSs was determined by the technical capabilities of the school, as well as previous experience of the development team

members in working with various databases. The school did not have a license for some other commercial DBMS (e.g. Oracle DB, IBM DB2, Informix...). Furthermore, the team members had no experience with some other open-source and freeware DMBS (e.g. PostgreSQL, Ingres, SQLite...). It was estimated that involving some of these systems in the project would require significant resources (either in additional funding, or in additional workload) and, therefore, would not be acceptable.

## TECHNICAL PRESUMPTIONS

In order to avoid influence of the network environment (eventual delays due to network protocols) and to avoid influence of various processors, several operating systems and different PC configurations, all testing were performed locally, on a single computer.

The computer was run under Windows 10 Professional 64-bit edition, operating system. The processor was 64-bit AMD Ryzen 7 27000X on 3.7 GHz, with 8 cores. The motherboard was Gigabyte X470 AORUS 5 Wi-Fi. The computer had 16 GB RAM memory, Kingston SUV 500/480 SSD disk used as a primary partition and a Western - Digital hard disk drive capacity 4 TB with 3 partitions. The working partition (i.e. the one containing the software and data) had capacity of 976 GB.

There are about 120 personal computers in the school running on different versions of MS Windows operating system (Windows XP, Windows 7, Windows 2008 Server, Windows 10). Some of PCs are relatively obsolete while others are brand new, but all of them has installed .NET Framework 3.5 which was accepted as a standard for the school information system. The same version was applied for the testing software. Due to the same reason (i.e. compatibility with older computers) executable version was built for 32-bit processors.

## RESULTS PRESENTATION

Separate measurements were performed for the following query types: INSERT, conditional (i.e. containing logical condition and WHERE clause) SELECT, UPDATE and DELETE, and finally conditional SELECT, command containing JOIN clause. The trivial cases of unconditional SELECT, UPDATE and DELETE were not tested since these cases are of no relevance in the school information system.

For each case there were 7 measurements, each measurement with different number of records (from 10 to 10,000 records, as already mentioned above). The highest and the lowest results were rejected, and arithmetic mean of the remaining five results was calculated and accepted as a final result.

The INSERT command was the simplest one, since there were neither WHERE nor JOIN clause involved. It should be told that every inserted record was a separate query, just like in the real system. Therefore, for each record it was necessary to establish a connection, send a request for the query execution and closing the connection. The respond time performing the INSERT command in seconds depending on number of records for all three databases is given in Table 1.

| Records | 10 | 100 | 250 | 500 | 1000 |
|---|---|---|---|---|---|
| MS SQL | 0.0595 | 0.4223 | 0.4253 | 0.4828 | 0.5889 |
| MySQL | 0.0667 | 0.4324 | 0.7876 | 1.0737 | 2.1799 |
| Access | 0.4837 | 4.3298 | 7.5662 | 12.034 | 24.098 |
| **Records** | **2500** | **5000** | **7500** | **10000** | |
| MS SQL | 1.4154 | 2.7311 | 4.0398 | 5.2746 | |
| MySQL | 6.2132 | 10.2802 | 20.3841 | 30.444 | |
| Access | 72.355 | 143.9219 | 274.5308 | 401.34 | |

***Table 1.*** *Results for INSERT command*

Graphical presentation of the results is given in Figure 1.

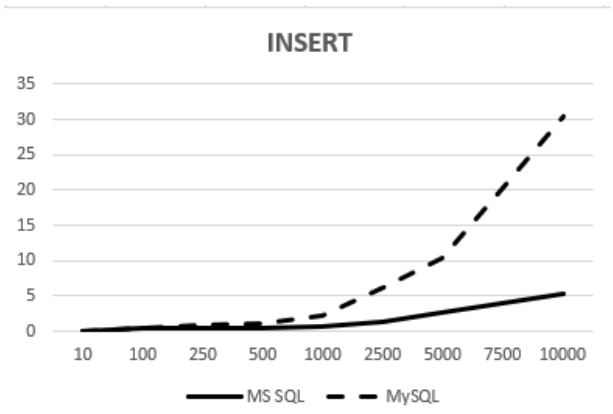


***Fig. 1.*** *Results for INSERT command (in seconds)*

There are not presented results for Access in the diagram, since it scored from 8 to 76 times slower result. If it was presented, the difference between MS SQL and MySQL would have been practically indistinguishable.

The respond time performing the UPDATE (including WHERE clause) command in seconds depending on number of records for all three DBMSes is given in Table 2.

time (in seconds)

| Records | 10 | 100 | 250 | 500 | 1000 |
|---|---|---|---|---|---|
| MS SQL | 0.0036 | 0.0039 | 0.0029 | 0.0050 | 0.0060 |
| MySQL | 0.0067 | 0.0070 | 0.0060 | 0.0070 | 0.0078 |
| Access | 0.0472 | 0.0480 | 0.0477 | 0.0493 | 0.0481 |
| **Records** | **2500** | **5000** | **7500** | **10000** | |
| MS SQL | 0.0078 | 0.0098 | 0.0146 | 0.0194 | |
| MySQL | 0.0117 | 0.0146 | 0.0216 | 0.0250 | |
| Access | 0.0520 | 0.0527 | 0.0567 | 0.0614 | |

***Table 2.*** *Results for UPDATE command*

It is noticeable that results are significantly faster then performing the INSERT command. It seems a bit confusing, since the UPDATE is much more complicated and time-consuming operation than the INSERT. The explanation for this is that UPDATE command is performed as a single query, demanding opening and closing the connection to the database only once for any number of records, while execution of the INSERT command requests separate query (i.e. opening and closing a separate database connection) for each record.

Graphical presentation of the execution of the UPDATE command results is given in Figure 2.

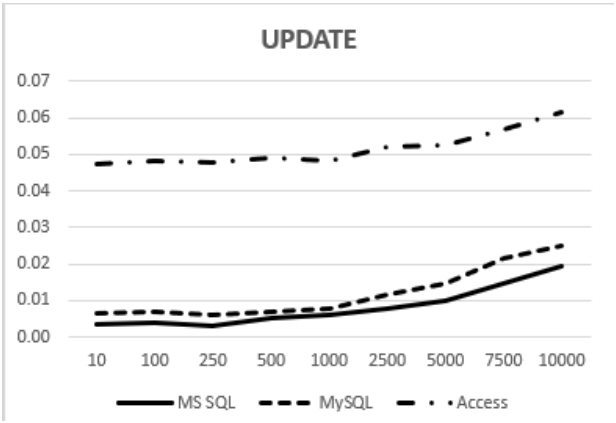


***Fig. 2.*** *Results for UPDATE command (in seconds)*

Just like in execution of the INSERT command, MS SQL Server achieved the best score, but the difference among different DBMSs is significantly smaller and barely depended on number of records.

The respond time performing the DELETE (including WHERE clause) command in seconds depending on number of records for all three databases is given in Table 3.

time (in seconds)

| Records | 10 | 100 | 250 | 500 | 1000 |
|---|---|---|---|---|---|
| MS SQL | 0.2344 | 0.2760 | 0.2987 | 0.3426 | 0.3876 |
| MySQL | 0.4504 | 0.5075 | 0.5299 | 0.5693 | 0.5978 |
| Access | 0.5460 | 0.5565 | 0.5987 | 0.6109 | 0.6301 |
| **Records** | **2500** | **5000** | **7500** | **10000** | |
| MS SQL | 0.3990 | 0.4216 | 0.4499 | 0.4942 | |
| MySQL | 0.6524 | 0.6988 | 0.8121 | 0.9857 | |
| Access | 0.7102 | 0.7783 | 0.9875 | 1.2282 | |

***Table 3.*** *Results for DELETE command*

Graphical presentation of the results in execution of the DELETE command is given in Figure 3.

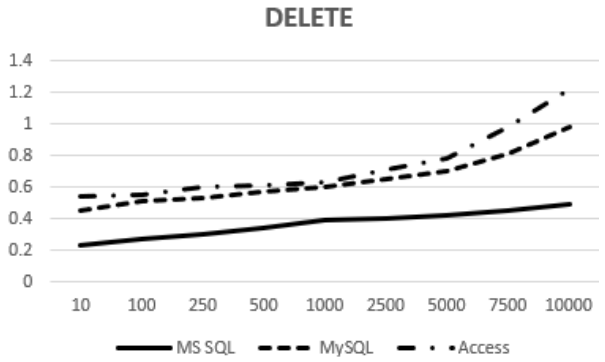


***Fig. 3.*** *Results for DELETE command (in seconds)*

The measured response time on executing the DELETE command including logical condition is almost linearly increasing depending on number of records. The most rapid response is achieved by MS SQL server. Interesting fact is that MySQL's result on deleting records is much closer to Access then to MS SQL.

The most important command is SELECT, since it will be the most frequently used in day-to-day work. The respond time performing the SELECT (including WHERE clause) command in seconds depending on number of records for all three databases is given in Table 4. Graphical presentation of the results in

execution of the UPDATE command is given in Figure 4.

time (in seconds)

| Records | 10 | 100 | 250 | 500 | 1000 |
|---|---|---|---|---|---|
| MS SQL | 0.0027 | 0.0100 | 0.0101 | 0.0149 | 0.0190 |
| MySQL | 0.0030 | 0.0101 | 0.0201 | 0.0400 | 0.0601 |
| Access | 0.0077 | 0.0471 | 0.0575 | 0.0761 | 0.0948 |
| **Records** | **2500** | **5000** | **7500** | **10000** | |
| MS SQL | 0.0200 | 0.0300 | 0.0500 | 0.2328 | |
| MySQL | 0.0696 | 0.0960 | 0.1801 | 0.4341 | |
| Access | 0.1050 | 0.1513 | 0.5504 | 0.6098 | |

***Table 4.*** *Results for SELECT command*

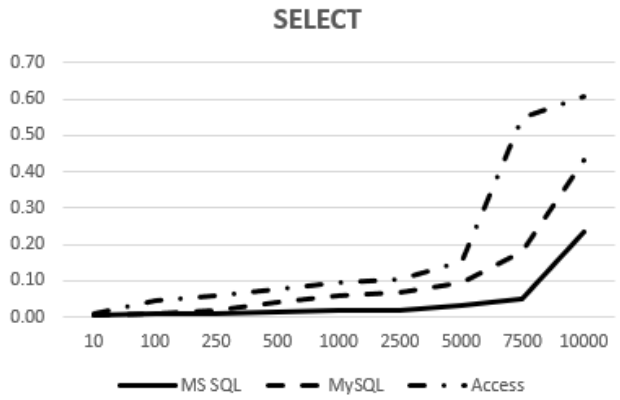


***Fig. 4.*** *Results for SELECT command (in seconds)*

As anticipated, the result gets slower as number of records increases, but that retardation is very slow for fewer number of records (i.e. less then 5,000 records). As number of records increases beyond that number, slowing down becomes more extensive and the delay gets more significant.

time (in seconds)

| Records | 10 | 100 | 250 | 500 | 1000 |
|---|---|---|---|---|---|
| MS SQL | 0.0109 | 0.0117 | 0.0168 | 0.0230 | 0.0401 |
| MySQL | 0.0101 | 0.0215 | 0.0301 | 0.0512 | 0.0691 |
| Access | 0.0542 | 0.0612 | 0.0748 | 0.0989 | 0.1421 |
| **Records** | **2500** | **5000** | **7500** | **10000** | |
| MS SQL | 0.0804 | 0.1220 | 0.2769 | 0.3921 | |
| MySQL | 0.1029 | 0.2102 | 0.2942 | 0.4334 | |
| Access | 0.2118 | 0.4398 | 0.6113 | 0.8069 | |

***Table 5.*** *Results for SELECT command including JOIN clause*

The most important case, according to the school information system's requirements is performing the SELECT command including logical condition (i.e. WHERE clause) and establishing relationship among several tables (i.e. existence of JOIN clause). Many of the queries performed by the target system would be of that kind, therefore the response time of performing this kind of queries is the most important for our purpose. The measured results for this type of query is given in Table 5. The graphical representation of the results is provided in Figure 5.

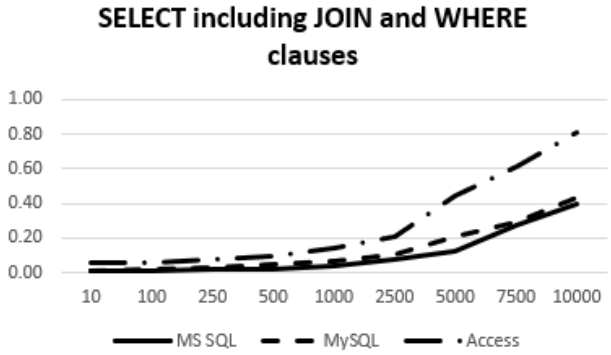


***Fig. 5.*** *Results for SELECT command with JOIN clause (in seconds)*

The results show a little difference among tested DB management systems for tables that contains less then 2500 records. For larger tables there is still no significant difference between MS SQL Server and MySQL, while MS Access starts to noticeably lag behind.

According to all performed measurements, undoubtedly best result in all cases is achieved by MS SQL Server. While under light load (i.e. less than 1,000 records) the difference between MS SQL Server and MySQL is almost negligible, under higher load the difference becomes more serious in MS SQL Server favor. This is especially true of INSERT and SELECT command (with no JOIN clause included) execution. Delay in performing INSERT command would be noticeable during inserting enrollment data. Delay in executing SELECT command is much more serious since the command is the most commonly used and it would affect performance of the school information system in general.

MS Access achieved by far the worse results in our testing and it was shown as not suitable for our purpose at all. That sort of database may be convenient for smaller tables and smaller amount of data, but even for the system size like ours, it is completely inappropriate.

## CONCLUSION

As a final conclusion of the presented testing, a Microsoft SQL Server, 64-bit 2016 version has been chosen as a database management system for information system of Valjevo Business School of Applied Studies. In the system exploitation so far, the choice has been proved justified, since response speed for all workstations in the school local network was quite satisfactory. Neither delays nor cease of functioning caused by the database management system has been recorded until today.

It should be emphasized that the purpose of this research was not to provide general and comprehensive comparative analysis or to provide overall evaluation of the tested DMBSs. The purpose was to establish which DBMS is the most appropriate for defined specific environment and conditions according to the concrete requirements. Testing the same DBMSs under different conditions (e.g. in internet environment or with many users logged on the system and using database simultaneously) could lead to entirely different results.

This research proved that it is possible, with a little effort, to make a testing tool to determine the most suitable DMBS for a specific condition instead of usage of standard general-purpose benchmarks which test many irrelevant features and functionalities. That enabled the best performance of the new information system, which was the primary objective of this analysis.

## REFERENCE


[1] Gray, J. (ed.). "The Benchmark Handbook for Database and Transaction Processing Systems 2nd edition". Morgan Kaufmann, 1993;

[2] Darmont, Jérôme, "Data Processing Benchmarks", Encyclopedia of Information Science and Technology, Third Edition, pp.146-152, 2014.

[3] Cudre-Mauroux Philippe, Kimura Hideaki, Lim Kian-Tat, Rogers Jennie, Madden Samuel, Stonebraker Michael, Zdonik Stanley B., Brown Paul G. "SS-DB: A Standard Science DBMS Benchmark" XLDB 2010, Stanford University, CA, Oct. 6-7, 2010.

[4] Alagić Suad, „Object-oriented Database Programming", Springer-Verlag, 1989.

[5] Saikia Amlanjyoti, Joy Sherin, Dolma Dhondup, Mary Roseline R., "Comparative Performance Analysis of MySQL and SQL Server Relational Database Management Systems in Windows Environment", International Journal of Advanced Research in Computer and Communication Engineering, Vol. 4, Issue 3, pp. 160-164, March 2015.

[6] Emison J.M., "2014 State of Database Technology", Information Week, San Francisco, CA, Rep. R7770314, 2014.

[7] DB-Engines.com, DB-Engines Ranking, on-line, August 2019., available on: http://db-engines.com/en/ranking_trend