Case study
UDC 519.683.5:681.3.07
DOI 10.7251/IJEEC2002111S

MS SQL server and MySQL response speed comparation

Stanišević Ilja[1], Vićentić Mlađen[1], Obradović Slobodan[2]

[1]Academy of Applied Studies Western Serbia, Valjevo, Serbia
[2]Information Technology School ITS, Belgrade, Serbia

ilja.stanisevic@vipos.edu.rs, *mladjen.vicentic@vipos.edu.rs*, *slobo.obradovic@gmail.com*

Abstract—**In order to select database managmeent tool (i.e., DBMS) for information system developed for Valjevo Business School, testing of two available options (i.e. MS SQL and MySQL) was performed. Instead of using available benchmark software, simple software to test features relevant to the system was developed. The tool was developed using the same technology as the school's information system and was designed to test only relevant features. The aim was not to provide general evaluation of the tested DBMS but to evaluate specific features and to determine option more relevant for the specific situation and environment. This paper describes applied methodology and displays realized results. In the first part of the paper the technical and organizational foundation is described. Then there is a description of possible solutions to the problem with the justification of the applied methodology. The first step in realization of the research was to determine testing domain and investigation framework. The features and requirements of the technical environment are shown. The conceptual design and features as well as user interface and available options of the testing tool realized for the purpose of this research are described. Finally the results and conclusions are presented.**

Keywords - Database management systems (DBMS), Microsoft SQL Server, MySQL, query response speed, performance evaluation, benchmark, low-budget software development

I. INTRODUCTION

During implementation of the integral information system for Valjevo Business School the question was which database management system should be used. Different members of the development team had various suggestions, depending on their experience and their leaning. Therefore, it was necessary to develop objective and neutral method and tool for evaluation which should give adequate and argumentative answer to the question. In this paper development and application of the method and tool as well as the results achieved are described.

II. TECHNICAL AND ORGANIZATIONAL FOUNDATION

The information system of the Valjevo Business School is supposed to be operative within a school's local area network (LAN) based on client-server model. The network consists of four servers (one domain controller, one back-up domain controller, one database server and one internet controller). It also contains around 120 workstations. The domain controllers and the database controller operate under Windows 8 server operating system. At the same time, due to different age of the workstations, they are operated under various versions of Windows operating system (i.e., Windows 10, Windows 7 and Windows XP). This diversity is a result of the way the school procured ICT equipment. It is not based on a proper planning but on occasional purchases depending on the available funds provided by the Ministry or through individual projects. Despite different operating systems, all workstations have installed Windows .NET 3.5 framework. Therefore, for the sake of compatibility with older workstations, the information system as well as the testing tool are developed for .NET framework 3.5. The system itself has been developed in C#.NET programming language using Windows Forms libraries and Microsoft Visual Studio as a development environment.

This paper is a revised and expanded version of the paper presented at the XIX International Symposium INFOTEH-JAHORINA 2020.

Due to security reasons school management decided to make the school system available exclusively within the school network. Therefore, usage of the system via internet is not supported. The system has a particular module that controls log-ins and access rights. The domain controller with role defining and access rights control is also used so that the system can rely on operation system security primitives.

When it comes to the functional requirements of the system, it should support the following business processes:

- the administrative actions of the students' service (registry of students, enrollment activities, colloquiums and exams registrations, exams records and reports, issuing various certificates etc.);

- activities on organization of lectures (lectures assignment, plan of various teaching activities, schedule of lectures, records of realized teaching activities, reports of different activities on annual or semester level);
- mentoring activities (monitoring learning efficiency and success for each subject, term and department, monitoring individual as well as group success of students, monitoring achieved results of each examination and colloquium terms, reports on number of students who registered an exam, number of students who accessed the exam, etc.);
- HR office activities (maintaining personal records of the teaching staff, records of attendance, absence, professional trainings etc.)

According to these requested requirements it can be noticed that the system will not be heavy loaded with input, but will be weighted with many report generations. Actually, due to the lack of stuff, the most of everyday data input activities (like student's applications and requests, examination records, held lectures records, student's attendance records, issuing of various documents, etc.) are centralized and they would take place at only one spot, at the students' service office. These activities, as well as issuing various reports would happen on-line in real-time (i.e., at students' service office's windows). Therefore, the primary evaluation criteria should be response speed, or, to be exact, database retrieval speed.

Of course, there are many other activities performed by other roles within the system (like library records, lectures assignment, lectures and examinations schedule, maintaining all kind of registers, employee attendance records, mentors' activities records, payroll calculations etc.) but these activities occur occasionally, from time to time, so they are not critical for the system performance in everyday work.

III. METODOLOGY APPLIED

In order to be able to objectively determine the performance of a database management system, appropriate measurements must be made with a suitable benchmark software tool. These measurements should provide objective evaluation of the tested database management systems. According to Gray [1], high-quality database testing benchmark system should provide the following characteristics:

- relevance – it has to be adequate for the largest number of potential users;
- portability – it can be applied on many different (desirably all) existing database management systems;
- simplicity – it has to be simple, easy to use and not to consume too many resources;
- scalability – it has to be adequate for many different (desirably all) computer systems and architectures, large as well as small.

These characteristics can be seen as contradictory. For instance, portability feature is frequently contradictory regarding the simplicity feature. Accordingly, it is necessary to determine reasonable compromise among these characteristics [2].

There are many factors that can influence results of the measurements. Some of these factors include hardware components. Thus, core numbers, working frequency, CPU speed, the amount of RAM memory, hard disk drive speed, bus speed, memory access speed, quality of the mainboard, etc. can have a significant influence on the measured results. System software can also have an impact on the results (for instance the way operating system manages the memory, threading of locking). Other significant factors could be data schema, database configuration (dedicated cash memory for query processing, number of established connections to the database, network protocols dedicated for database access, index implementation, etc.), the amount of previously recorded data, type of database access application, etc. [3]

One possible approach to resolve the problem is to apply some of the existing benchmarking tools. There are many measuring software tools which can test and evaluate different database features. Some of the benchmarks are made for general purpose. On the other hand, some of the benchmarks are specialized for testing certain aspects of the tested database management system. Some are dedicated for transaction processing tests, the others are for relational databases or for object-oriented databases, there are benchmarks for testing XML based databases, for decision support systems (i.e., DDS) evaluation, for cloud-based databases, for evaluation of non-SQL databases etc. [2] [4] Due to the variety of benchmarking tools, it was necessary to establish a separate independent organization to deal with database benchmark standards. This organization is the non-profit corporation founded by Transaction Processing Performance Council – TPS and it issues standards for database benchmarks and verifies accuracy of benchmarking tools [4].

There are many database benchmarks available on the market today. Significant part of them is open-source or free software and their usage does not require additional funding. For instance, it is possible to use Quest [5], STS Soft [6], HammerDB [7] or any other. However, there is a significant issue related to usage of these benchmark tools as most of them are designed and developed for general purposes. Consequently, these tools test, measure and evaluate as many features as possible.

Many of these features are irrelevant for the objectives of our project. For instance, the school's system will have only few input spots so that evaluation marks dedicated for dealing with many connections to the database is in this case of no importance. Furthermore, the school's system will be available only within local area network and will not be available through internet, so that internet performance is completely irrelevant.

On the other hand, there are benchmark tools designed to be applied for certain specific purposes [8] as well as benchmarks which evaluate only small number of database management system features [9]. These tools are also not appropriate to determine the most suitable DBMS for the purpose of our project due to the limited scope of their application.

Having all this in mind, the development team has decided to adopt a different strategy. Instead of using the existing benchmarking tool(s), the team decided to develop a simple program to test and evaluate only features relevant to the development of the school's system. Furthermore, the program was to be developed in the same environment and using the same technology as the school's system itself. The tool should be perfectly adjusted to meet and test evaluation criteria requested in this case. Of course, this tool would not satisfy Gray's criteria of portability and scalability [1] but these features are not relevant for our purpose. Anyway, the tool would to a large degree meet criteria of relevance (as designed and developed for a particular case it would be most relevant for this case) and simplicity (only features relevant for the particular case would be tested, therefore the tool should be simple and easy to use).

IV. Testing Domain Determination

Since the school's database is a relational one hence the testing tool should be designed to test relational databases. The main adopted evaluation criterium is query response speed. Therefore, the testing tool is intended for measuring response speed to various SQL queries. This includes data retrieve queries (i.e., SELECT queries) as well as action queries (i.e., INSERT, UPDATE and DELETE queries). There were separately taken into consideration queries without any logical condition and queries containing simple or composite logical conditions (i.e., containing WHERE clause). Furthermore, there were separate measures for queries under single data table (i.e., relation) and under several tables (i.e., containing JOIN clause). The measurements were conducted under different amount of data already written in the database and with various amount of data to be written or changed. Queries were generated in a way that logical conditions were met by approximately 20% of records.

It was necessary to determine the amount of data in the relations. The largest number of records in transactions was detected during student enrollment process. This includes new students as well as enrollment of registered students for a new academic year. The number of these records varies between 150 to 250 records, depending on the number of students on each term. Therefore, testing transactions which refer to a larger number of records is not relevant for our research.

It was necessary to determine maximum quantity of records per single table. For that purpose, the objects that have the most instances were determined. Performing investigation of the previous records, it was established that the most numerous objects were colloquium applications. Number of these instances on annual level can be determined based on the following formula:

$$PK = \sum_{i=o,m,d} (\Gamma_i * \alpha_i * BS_i * BK_i) \quad (1)$$

where:

i stands for the type of students (i.e., o for bachelor, m for master and d for distance learning students);

PK stands for total number of colloquium applications;

Γ_i stands for duration of ith type of study in academic years;

α_i stands for active students' ratio on ith type of study;

BS_i stands for number of students on ith type of study;

BK_i stands for average number of colloquiums during an academic year on ith type of study.

The school has accreditation for total of 165 students for regular bachelor level studies, 50 students for master level studies and 32 students for distance learning studies of a bachelor level. This division is exclusive, since distance learning studies of a master level cannot be accredited in Serbia. One student of a bachelor level can be enrolled either as a regular student or as a distance learning student, but not both. On a bachelor level and distance learning studies average number of colloquiums on annual level is 14, while on master level studies it is 6 colloquiums (since on this level there are less subjects per academic year). Duration of bachelor and distance learning studies is 3 academic years while duration of the master studies is 2 academic years.

In each generation, a number of students just formally enrolls the school and stays inactive during the school activities. Either they want to hold pensions or some other benefit, or they want to have a student's status, or simply they just overestimate their learning capacity. These students have insignificant impact on the amount of data in the database. Therefore, the active student ratio is introduced into the equation (1). Student is considered active if he/she applies for at least one examinational activity. In real life, students usually either apply for the most of the activities or they stay utterly inactive for the whole academic year.

Significant number of students are registered but they are neither attending classes nor taking the exams. That fact had to be taken into consideration and therefore additional active students' ratio was introduced. Active students' ratio for the bachelor studies is 0.38, for the master studies it is 0.57 while for the distance learning studies is 0.15. The efficiency coefficient (i.e., number of students who successfully accomplished an academic year) has not been taken into consideration since that number is compensated with number of students who attend again either academic year or a particular subject. When we include these values into equation (1), for various type of studies we get the following equation:

$$PK = PK_o + PK_m + PK_d \quad (2)$$

where PK stands for total number of colloquium applications. Including determined numbers, we have:

$$PKo = 3 * 0.38 * 165 * 14 = 2{,}633.4$$

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

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

Finally, including these values into equation (2), we have the estimated total number of colloquium applications for one year:

$$PK = 2{,}633.4 + 342 + 201.6 = 3{,}177 \quad (3)$$

In this way it was estimated that the most burdened relation would have increase of 3,177 records on the annual level. For three-year period (which is average period between the school system general updates) the amount would be 9,531 records. Having in mind decreasing tendency of number of students which is present for a longer period (mostly due to declining birth rate in Kolubara region and in whole Western Serbia where the school is located) and current number of pupils in elementary and secondary schools in the same region, the number of records could only be decreased.

Similar estimations were performed for other objects consisting larger number of instances, like examination registrations, track of delivered lectures, attendance records etc. Estimation results were not larger than presented in any case. Of course, most of the relations (like professors' registry, registries of classrooms, subjects, departments, employees, graduated students etc.) have significantly smaller number of records, usually 10-1,000 records.

Therefore, it was decided that all measurements would be performed up to maximum 10,000 records in the relations. Measurements under higher burden of records would be irrelevant for our research.

Since the database for the school's system according to initial design was normalized relational database in the Boyce-Codd normal form [10], the tested database had to be of the same type. It was important to evaluate database management system performance dealing with related tables. Therefore, the ability of generating testing queries containing JOIN clause had to be included in the testing tool.

In the case of testing small amount of data like a single input of a record, delay caused by network protocols can significantly overcome DMBS response time which can lead to incorrect and misleading results. As a result of all these considerations, it was decided that measurements would be conducted within the range of minimum 100 records up to maximum 10,000 records. This range covers all relevant cases for this purpose.

V. TECHNICAL ENVIRONMENT

To be as adequate as possible, the measurements were performed in real working environment at the school where the school's information system would be running.

Within the school's local area network, dedicated database server has Intel Pentium CPU G2130 processor, which runs on 3.2 GHz and has 8GB of RAM memory. It runs under Windows Server 2008 Enterprise without Hyper-V SP2 64-bit version operating system. Hard disk drive is divided into two partitions containing 232 GB each. Database management system runs on the primary partition.

Regarding software used for the development of the testing tool, it was developed in the programming language C#.NET, the very same as the one that was used for the school's information system development. Consequently, influence of the programming language on the measurements results could be ignored. Development framework was .NET Framework, version 3.5 which was adopted as a standard for every work station within the school's local area network. Reason for this was combability among different Windows versions installed on working stations of the School's information system. Of course, the same framework version was used for the information system development. For the sake of combability, the final executable version of the testing tool, as well as of the information system, was adjusted for 32-bit processor. Integrated development environment used was MS Visual Studio Community 2017, version 15.9.14.

A selection of which database management systems should be tested was made according to technical characteristics of the school's equipment as well as to development team member's experience. The school had no valid license for some commercial DBMS like ORACLE, DB, IBM DB2, Informix etc. On the other hand, the development team members had no experience working on some other open source or freeware systems like PostgreSQL, Ingres, Informix etc. It was estimated that introduction of these DMBSs would require increase of resource consumption (either in additional funds or in the extension of working hours), therefore these systems were not taken into consideration.

For final testing two most common systems were selected: MS SQL Server (version 2016 Standard, 64-bit) and MySQL (version 8.0.16 – MySQL Community Server – GPL). Preliminary MS Access (version Professional Plus 2016) was also taken into consideration, but it achieved extremely poor results during preliminary measurements performed on a stand-alone workstation, so it was abandoned for the final testing in the network environment [11].

MS SQL Server is the most common DBMS for business purposes while MySQL is the third most common [12]. Generally, the most commonly used is Oracle DB while MySQL and MS SQL Server are ranked in the second and the third place consequently [13]. It can be concluded that MS SQL Server is predominant for the business-oriented applications in commercial organizations while MySQL is predominant for the development of internet applications combined with Apache server and PHP programming language. The fact that many content management systems like WordPress include MySQL contributes to MySQL popularity significantly. On the other hand, as part of Microsoft Office, MS Access despite many functional limitations is on the fifth place in the most popular database management systems list [13].

VI. TESTING TOOL DEVELOPMENT

The conceptual design of the testing system is shown in Fig. 1. The original design includes MS Access testing option which was abandoned later. This design enables relatively easy extension for testing other database management systems as well by simple addition of the adequate data provider and generating simple method to establish database connection within Database Interface layer [14].

The intention was to make the testing system as user friendly as possible, but at the same time to provide all required functionalities.

The topmost layer is the user interface which was implemented using MS Windows Forms library and its components. The reason to choose this technology instead, for instance, the more contemporary Microsoft Presentation Foundation MPF [15] [16] or Universal Windows Platform UWP [17] [18] lies in the fact that the school system was built in WinForms technology. Thus, the influence of chosen technology to measurement outcome is eliminated. In addition to this, the most common controls (i.e., buttons, text boxes, list and combo boxes and check boxes) are used which contributes to the ease of users training and makes the user interface simple and intuitive.

A user has ability to pick the type of DBMS he wants to test. He can burden desired tables with any number of records which occurs automatically and is not consuming measured time. Furthermore, the user can choose number of records to insert within the measured time. He can determine which SQL command he wants to perform (i.e., SELECT, INSERT, UPDATE or DELETE). He can also define logical conditions the resulting records need to meet. The conditions can be simple or composite (i.e., using operators AND, OR or NOT) one as well.

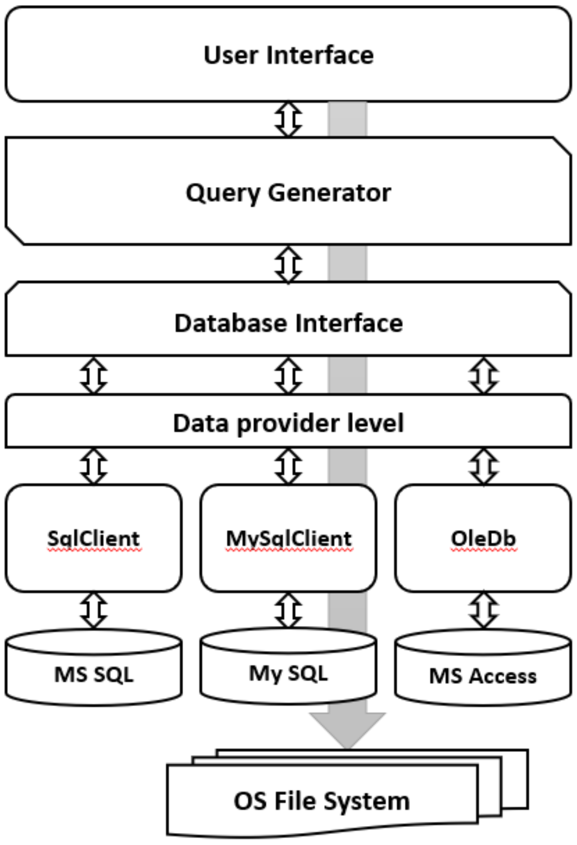
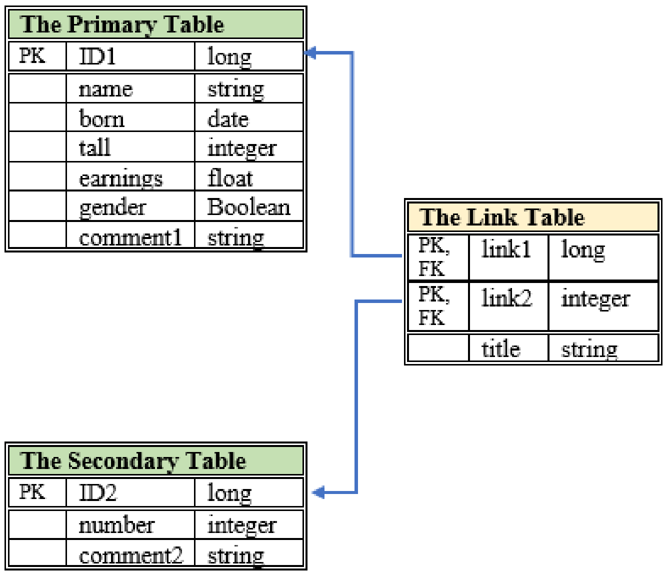


Figure 1. Testing system conceptual design

By simple checking the adequate check-box, the user can enforce inclusion of the JOIN clause, connecting three tables, according to the rules of relational database model [10]. The testing database consists of three tables. Between two of them there is a M:N relation. Therefore, there is the third table engaged as a link table to establish correct connection between the two tables. This structure is presented in Fig.2

When the user activates the adequate check-box, the system automatically generates data and inserts them into the second table as well as establishes connection between the two tables by filling corresponding data into the auxiliary table. The JOIN clause is then added into the generated query. In this way we can measure response of the database management system when more tables are involved [14].

We adopted this database structure as the most generic one since any other case contained in the specific school's database design can be derived from this structure. Of course, the school's information system would contain much more tables and more complex structure, however all of them could be reduced to the set of the structures as the one represented in Fig.2. This conclusion is not general, but it can be applied in this specific case. In case of a different database structure, requesting the different testing database, the testing database can easily be modified and implemented. It would cause only minor modifications in code within the Query generator.



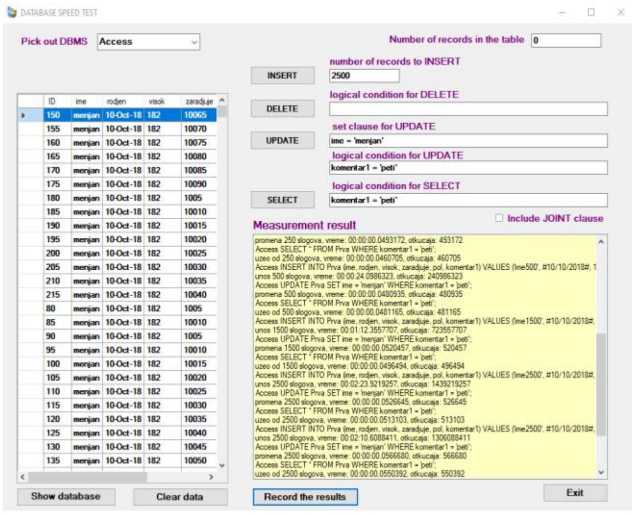
The Primary Table		
PK	ID1	long
	name	string
	born	date
	tall	integer
	earnings	float
	gender	Boolean
	comment1	string

The Link Table		
PK, FK	link1	long
PK, FK	link2	integer
	title	string

The Secondary Table		
PK	ID2	long
	number	integer
	comment2	string

Figure 2. Structure of the tested database

It can be noted that the primary table contains fields of all major data types. It was done in order to make results of the measurements independent of data types presented in the table. The School's information system would contain fields of different types as well, but according to the initial design, it the majority will be of basic types. This was the way to make testing database as realistic as possible.



Pick out DBMS: Access

Number of records in the table: 0

number of records to INSERT: 2500

logical condition for DELETE:

set clause for UPDATE: ime = 'menjan'

logical condition for UPDATE: komentar1 = 'peti'

logical condition for SELECT: komentar1 = 'peti'

☐ Include JOINT clause

ID	ime	rodjen	visok	zaradjuje
150	menjan	10-Oct-18	182	10065
155	menjan	10-Oct-18	182	10070
160	menjan	10-Oct-18	182	10075
165	menjan	10-Oct-18	182	10080
170	menjan	10-Oct-18	182	10085
175	menjan	10-Oct-18	182	10090
180	menjan	10-Oct-18	182	1005
185	menjan	10-Oct-18	182	10010
190	menjan	10-Oct-18	182	10015
195	menjan	10-Oct-18	182	10020
200	menjan	10-Oct-18	182	10025
205	menjan	10-Oct-18	182	10030
210	menjan	10-Oct-18	182	10035
215	menjan	10-Oct-18	182	10040
80	menjan	10-Oct-18	182	1005
85	menjan	10-Oct-18	182	10010
90	menjan	10-Oct-18	182	1005
95	menjan	10-Oct-18	182	10010
100	menjan	10-Oct-18	182	10015
105	menjan	10-Oct-18	182	10020
110	menjan	10-Oct-18	182	10025
115	menjan	10-Oct-18	182	10030
120	menjan	10-Oct-18	182	10035
125	menjan	10-Oct-18	182	10040
130	menjan	10-Oct-18	182	10045
135	menjan	10-Oct-18	182	10050

Measurement result

```
promena 250 slogova, vreme: 00:00:00.0493172, otkucaja: 493172
 Access SELECT * FROM Prva WHERE komentar1 = 'peti';
uzeo od 250 slogova, vreme: 00:00:00.0460705, otkucaja: 460705
 Access INSERT INTO Prva (ime, rodjen, visok, zaradjuje, pol, komentar1) VALUES ('ime500', #10/10/2018#, 1
unos 500 slogova, vreme: 00:00:24.0986323, otkucaja: 240986323
 Access UPDATE Prva SET ime = 'menjan' WHERE komentar1 = 'peti';
promena 500 slogova, vreme: 00:00:00.0480935, otkucaja: 480935
 Access SELECT * FROM Prva WHERE komentar1 = 'peti';
uzeo od 500 slogova, vreme: 00:00:00.0481165, otkucaja: 481165
 Access INSERT INTO Prva (ime, rodjen, visok, zaradjuje, pol, komentar1) VALUES ('ime1500', #10/10/2018#,
unos 1500 slogova, vreme: 00:01:12.3557707, otkucaja: 723557707
 Access UPDATE Prva SET ime = 'menjan' WHERE komentar1 = 'peti';
promena 1500 slogova, vreme: 00:00:00.0520457, otkucaja: 520457
 Access SELECT * FROM Prva WHERE komentar1 = 'peti';
uzeo od 1500 slogova, vreme: 00:00:00.0496494, otkucaja: 496494
 Access INSERT INTO Prva (ime, rodjen, visok, zaradjuje, pol, komentar1) VALUES ('ime2500', #10/10/2018#,
unos 2500 slogova, vreme: 00:02:23.9219257, otkucaja: 1439219257
 Access UPDATE Prva SET ime = 'menjan' WHERE komentar1 = 'peti';
promena 2500 slogova, vreme: 00:00:00.0526645, otkucaja: 526645
 Access SELECT * FROM Prva WHERE komentar1 = 'peti';
uzeo od 2500 slogova, vreme: 00:00:00.0513103, otkucaja: 513103
 Access INSERT INTO Prva (ime, rodjen, visok, zaradjuje, pol, komentar1) VALUES ('ime2500', #10/10/2018#,
unos 2500 slogova, vreme: 00:02:10.6088411, otkucaja: 1306088411
 Access UPDATE Prva SET ime = 'menjan' WHERE komentar1 = 'peti';
promena 2500 slogova, vreme: 00:00:00.0566680, otkucaja: 566680
 Access SELECT * FROM Prva WHERE komentar1 = 'peti';
uzeo od 2500 slogova, vreme: 00:00:00.0550392, otkucaja: 550392
```

Figure 3. The main user interface form

The user has some additional functionalities as well. He/she can display the content of the tested table or a result of the generated query in a DataGrid view control on the left side of the form or clear the contents of the table. There is a list box on the main form which displays generated query and measurements results (i.e., the time achieved during execution of the testing query). These results can be saved on the user's request (by causing an interrupt clicking the adequate button on the form) in a file on the hard disk for further investigation and additional analysis. The design of the main user form is displayed in the Fig. 3.

The role of the Query Generator layer is to generate a query corresponding to the user's input on the form, and to transfer the query to the Database Interface. Query Generator performs a simple parsing and takes care of syntax differences among different SQL dialects (like date/time formats, type and usage of brackets, dealing with logical conditions format etc.). The outcome depends on the DBMS type chosen from the combo box at the upper left corner of the main interface form. The choice of the DBMS type causes change of a global static variable which determines usage of the adequate data provider.

Database Interface layer establishes a connection to the database via corresponding data provider, sends a request for the generated query execution, picks up the result, puts it in the correct form control and breaks the connections. Finally, at the end of the measuring session, on the user's request, the OS file system is invoked to save achieved results to the hard disk drive.

This design, besides its functionality and user-friendliness, is very flexible and makes adding other database management systems very simple. All that is needed is including a proper data provider and a small method, like the one described above.

VII. THE RESULTS PRESENTATION

The measurements for the following cases were performed: for unconditional (i.e., without WHERE clause within the generated query) SELECT, DELETE and INSERT statement, for conditional (i.e., using queries containing WHERE clause) SELECT, UPDATE and DELETE statement and, finally, composite (i.e., among multiple tables using queries containing JOIN clause) SELECT statement. The measurements were conducted under various number of records in the tables from 100 records gradually up to 10,000 records in the tables. The explanation of these limits in number of records tested is given in the Section IV: Testing Domain Determination. For each combination total of seven measurements were performed. The highest and the lowest results were discarded. The arithmetic mean of the remaining results was accepted as the final result for each particular case.

It was necessary to take into consideration feature of .NET framework and JIT interpreter to load large blocks of the code into the central memory during initial startup of a program. This feature could lead to misleading and confusing results. It can happen that execution of a short and simple query over small number of data consume several orders of magnitude more times than complex and composite query performed on large scale of records. To avoid influence of such behavior of the used tool and platform, before each measurement session, several initial measurements were performed and these measurements were not recorded as a valid outcome.

Measured response times in case of the INSERT command for MS SQL Server and MySQL are given in Table I. The results are given in seconds depending on the number of records in the tested table.

TABLE I. RESPONSE TIME FOR INSERT COMMAND

MS SQL	0.21	0.59	0.77	1.02	1.31	3.34	6.49	9.76	13.09
MySQL	0.34	0.63	0.87	1.29	3.97	7.59	13.48	25.36	35.44
records	100	250	500	750	1000	2500	5000	7500	10000

It is important to notice that insert of the records was performed separately, i.e., during an insert of each record the database connection was established and it was deleted afterwards, when the command was successfully executed. This is the reason why the measured response time is longer than the time measured during execution of much more demanding queries, like the ones containing the UPDATE command, when the connection to the database was established and deleted only once. Of course, establishing a database connection is a significantly time-consuming operation.

This effect was not corrected as the way of performing the INSERT command is the same as in the School's system. Therefore, the results achieved with this approach can be accepted as valid for the purpose of this research.

Chart with the results achieved is given in Fig. 4.

It can be concluded that the result gets slower as the number of records increases, but that retardation is quite slow for smaller number of records (i.e., less than 750 records). As number of records increase beyond that number, slowing down becomes more intensive and the delay becomes more significant. It should also be remarked that the difference gets more notable in favor of the MS SQL Server as the number of records increases.

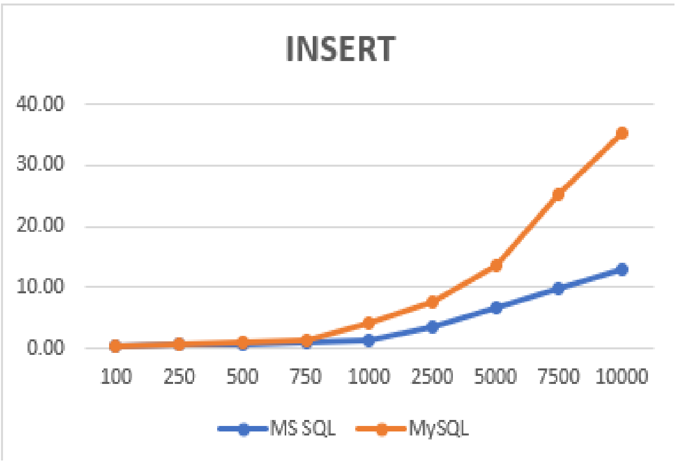


Figure 4. Response time for INSERT command

As explained above, the significantly shorter time than in the case of the INSERT command is due to single initiation of the database connection. Updating of any number of records is performed by execution of only one SQL query.

TABLE II. RESPONSE TIME FOR UPDATE COMMAND

MS SQL	0.18	0.30	0.32	0.36	0.42	0.89	1.40	1.95	2.48
MySQL	0.21	0.35	0.41	0.44	0.51	1.08	1.72	2.34	3.67
records	100	250	500	750	1000	2500	5000	7500	10000

Graphical representation of the results is given in the Fig.5. The results are given in seconds depending on the number of records in the tested table. It is noticeable that difference of the measured response time of MS SQL Server and MySQL is significantly smaller, but still increasing with the number of records in favor of MS SQL Server. Explanation for this lies in already mentioned number of database connections establishment. MS SQL Server is built using the same Microsoft technology as the developed testing tool. Therefore, it is natural that establishing a link to a database consumes less time than in the case of making connection to MySQL which is developed using different technology, i.e., MS SQL Server data provider is much more efficient than MySQL data provider.

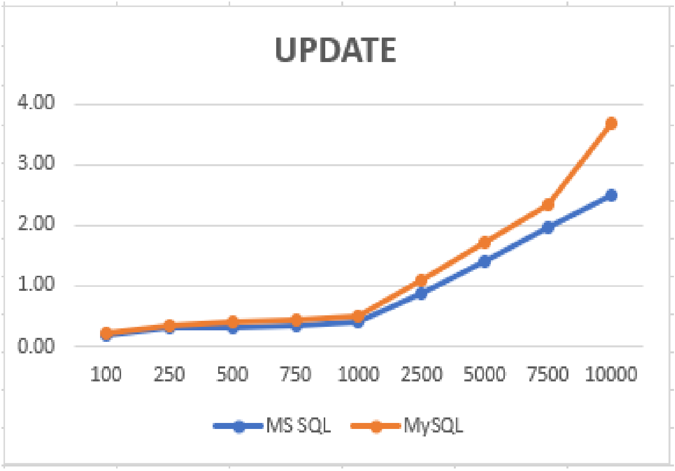


Figure 5. Response time for UPDATE command

Table III displays achieved response times during execution of the queries containing the DELETE command for MS SQL Server and MySQL depending on number of the records in the testing tables.

TABLE III. RESPONSE TIME FOR DELETE COMMAND

MS SQL	0.32	0.38	0.41	0.48	0.54	0.62	1.11	1.40	1.96
MySQL	0.51	0.58	0.63	0.67	0.73	0.83	1.21	1.87	2.51
records	100	250	500	750	1000	2500	5000	7500	10000

The difference between measured response times for MS SQL Server and MySQL is still in favor of MS SQL Server, but the difference is approximately constant and is not significantly dependent on the number of records in the tested tables. The graphical representation of the results for DELETE command is presented in the Fig.6.

The result for performing SELECT command under a single table (i.e., without JOIN clause) in seconds depending on number of records is given in the Table IV.

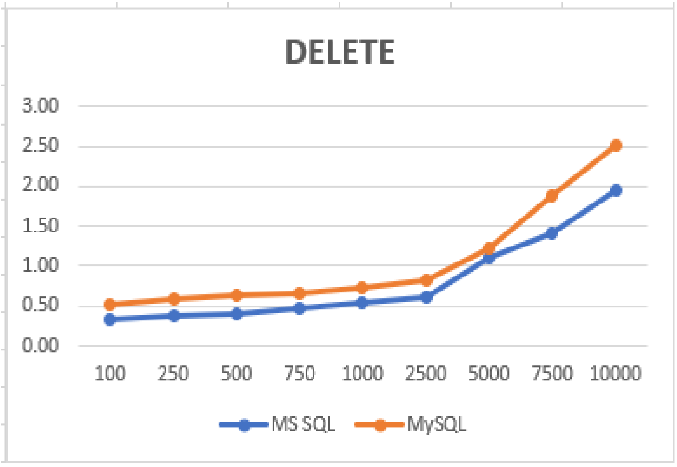


Figure 6. Response time for DELETE command

TABLE IV. RESPONSE TIME FOR SELECT COMMAND

MS SQL	0.26	0.32	0.35	0.39	0.48	0.54	0.69	0.90	1.21
MySQL	0.28	0.36	0.41	0.46	0.57	0.68	0.94	1.26	1.52
records	100	250	500	750	1000	2500	5000	7500	10000

The graphical representation of the results for SELECT command execution is given in the Fig.7.

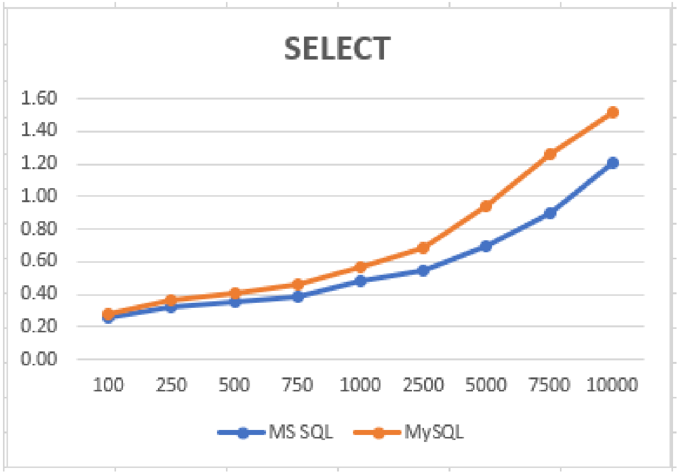


Figure 7. Response time for SELECT command

The results show that MS SQL Server is slightly faster than MySQL in performing the SELECT command over a single data table and that MySQL's delay increases with the increase of the records in the table, but neither that delay nor its increase are significant.

The most important case concerning the School's information system is execution of a query containing the JOIN clause, the queries that investigate more connected tables. Most of the queries requested from the School's system will be of that type, almost every report or document issued by the system. The outcome from response time measurement for the execution of SELECT queries containing logical conditions and JOIN clause (i.e., queries that involve more connected tables) is given in the Table V.

TABLE V. RESPONSE TIME FOR SELECT WITH JOIN COMMAND

MS SQL	2.13	3.10	3.49	3.97	4.40	6.21	9.60	12.80	15.40
MySQL	2.18	3.19	3.74	4.23	4.92	6.53	10.12	13.98	17.24
records	100	250	500	750	1000	2500	5000	7500	10000

As anticipated, the determined response speed is much slower than for the queries that execute SELECT command over a single table. It is shown that the final time given in seconds is significantly dependent on the number of records. The response time gets longer as the number of records gets larger. On the other hand, it is interesting that the resulting times for MS SQL Server and MySQL server are not significantly different. Nevertheless, MS SQL server is still slightly faster than MySQL.

The chart based on the numbers represented in the Table V is displayed in the Fig.8.

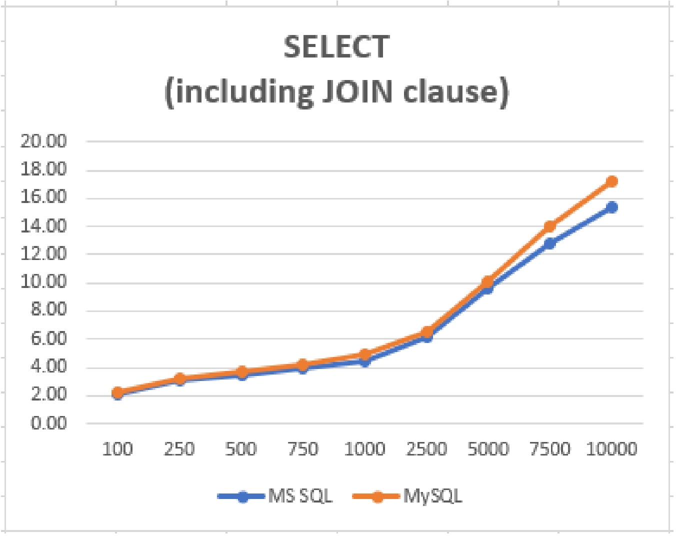


Figure 8. Response time for SELECT command

Similar research was preliminary conducted but on a single workstation instead in a network environment [11]. Databases, database management systems, integrated development environment and testing tool were running on the same workstation. In this way, the influence of the network protocols was avoided.

It is of interest to compare performance of the tested database management systems in the local area network environment and on the single workstation in order to determine delay caused by the execution of network protocols. Measured results representing response time for each described case of SQL commands in the network environment and on the single workstation in seconds are given in the Table VI. As a resulting time, the arithmetic mean of all achieved times (for number of records from 100 to 10,000) was accepted.

The chart comparing delays caused by the execution of network protocols for MS SQL Server and MySQL is displayed in the Fig.9.

TABLE VI. COMPARATION OF RESPONSE TIME MEASURED IN A LAN ENVIRONMENT AND ON A SINGLE WORKSTATION

	INSERT	UPDATE	DELETE	SELECT	SELECT /JOIN
MS SQL in LAN	4.06	0.92	0.80	0.57	6.79
MS SQL on local WS	1.72	0.01	0.37	0.04	0.11
network delay	2.34	0.91	0.43	0.53	6.68
MySQL in LAN	9.89	1.19	1.06	0.72	7.35
MySQL on local WS	7.98	0.01	0.64	0.10	0.14
network delay	1.91	1.18	0.42	0.62	7.21

It can be noticed that MS SQL Server is less sensitive on network influence while performing SELECT, UPDATE and SELECT containing the JOIN clause. At the same time, surprisingly, MySQL performs faster execution of the INSERT command. During execution of DELETE command, both systems have approximately equal delay in the network environment. Finally, it can be concluded that the network delay is not decisive advantage for any of the tested systems.

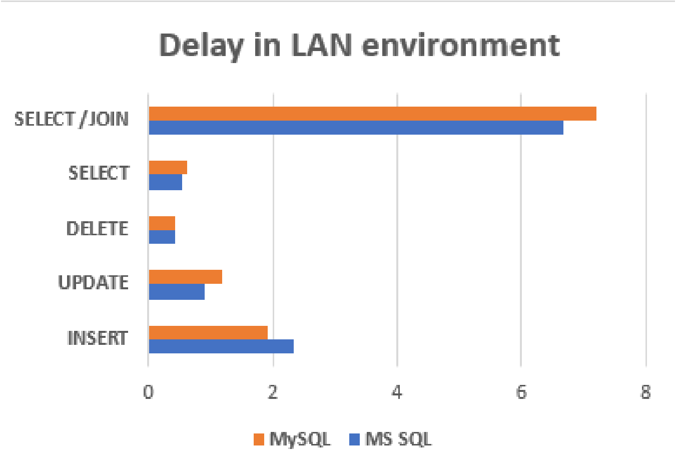


Figure 9. Delays caused by the execution of network protocols (in seconds)

VIII. CONCLUSION

Although the distinction between the evaluated performances of the two tested database management systems is relatively small (except for execution of INSERT command performed on the tables containing large number of records) it has to be emphasized that MS SQL Server has slight advantage in practically every evaluated case. This was the reason why Microsoft SQL Server was chosen for the development of the School's information system.

In the use of the system so far, no delays due to DBMS systems were detected. All requested tasks were performed on time. Therefore, it can be concluded that the choice was right as well as it was properly justified.

It must be stressed out that the purpose of this research was not to provide general estimation for the tested systems nor to determine which system was better. The purpose was to determine which database management system is more suitable for a certain information system in a pre-defined environment, and which one would better meet given requirements.

Testing performed in the different environment and under different conditions and criteria could provide a different conclusion. Some important aspects and features were excluded from these evaluation criteria, like response time in the internet environment or performance when there is a large number of online users connected to the database. Although these features are significant in general, for the purpose of this research they were irrelevant so their absence is justified.

It is possible to object that the results of the research are predictable, since, as the result, the Microsoft's SQL Server proved faster and better used under Microsoft technology. Still, there were members of the development team who had different impression as a result of their experience in using Oracle's MySQL Server. Therefore, this research was necessary to argumentatively justify the decision of selecting SQL Server for the school's system.

During this research it was shown that it is, in the decision-making process, possible to rely on small but efficient self-made testing tools adapted for real-life situation and adjusted for concrete requirements instead of depending on often inadequate and possible misleading benchmarking tools designed for general purpose.

The main contribution of this research, however, is to show and promote that it is possible to resolve a problem not only by using complex general-purpose tools, but by developing a smaller and simpler goal-oriented tools designed for a single specific purpose. Such development is much cheaper, faster and requires less resources. This strategy is important in the cases of software development under low-budget conditions [19].

In many developing countries there is a lack of important resources. There is a problem to provide sufficient financial, personal, technical or software resources. Still, there is a need for software development in these societies. One of the ways to overcome this problem is a strategy change [20]. Instead of developing complex, large, resource-consuming, general-purpose systems and tools, it is possible to develop smaller, cheaper, not general and not the best but still acceptable solution. This is a possible way to improve the development of information systems in less wealthy societies and organizations.

REFERENCES

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

[2] J. Darmont, "Object Database Benchmark" Encyclopedia of Information Science and Technology, I-III, Idea Group Publishing, pp. 2146-2149, 2005.

[3] MySQL, "MySQL Performance Benchmarks – Measuring MySQL Scalability and Throughput", A MySQL Technical White Paper, March 2005, available at http://www.jonahharris.com/osdb/mysql/mysql-performance-whitepaper.pdf.

[4] J. Darmont, "Data Processing Benchmarks" Encyclopedia of Information Science and Technology, Third Edition, Idea Group Publishing, pp. 146-152, 2014.

[5] Quest, Benchmark Factory for Databases, available at: https://www.quest.com/products/benchmark-factory/.

[6] STS Softy, Database Benchmark, available at: https://www.quest.com/products/benchmark-factory/.

[7] HammerDB, available at: https://www.hammerdb.com/.

[8] 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.

[9] McKnight William, Dolezal Jake and Barker Roger, "Cloud Database Performance Benchmark, Product Profile and Evaluation: Actian Vector and Microsoft SQL Server", MCG Global Services, February 2018.

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

[11] Stanišević I., Obradović S., Vićentić M., „Database Managment System Selection", International Scientific Conference UNITECH 2019, pp. II-33:II-38, Gabrovo, Bulgaria, EU, 15-16 November 2019.

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

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

[14] Stanišević I., Obradović S., Vićentić M., „DBMS Response Speed Testing System", International Scientific Conference UNITECH 2019, pp. II-39:II-43, Gabrovo, Bulgaria, EU, 15-16 November 2019.

[15] MacDonald M., *"Pro WPF 4.5 in C#: Windows Presentation Foundation in .NET 4.5",* Apress, New York, NY, USA, 2013.

[16] Microsoft Docs, Microsoft Presentation Foundation, avaiable at: https://docs.microsoft.com/en-us/dotnet/desktop/wpf/?view=netframeworkdesktop-4.8.

[17] Chatteryee A., "*Building Apps for the Universal Windows Platform: Explore Windows 10 Native, IoT, HoloLens, and Xamarin*", Apress, New York, NY, USA, 2017.

[18] Microsoft Docs, Windows Dev Center, Universal Windows Platform, avaiable at: https://docs.microsoft.com/en-us/windows/uwp/get-started/universal-application-platform-guide.

[19] Stanišević I., Obradović S. „Characteristics of the Optimal Method for Software Design in a Low-budget Environment", Megatrend Review, Vol 8,No 2, UDC 005.8:004.41, ISSN 1820-4570, pp. 597-524, 2011.

[20] Stanišević I., Obradović S., Yordanova M. „Development of Business Software Adapted to Local Conditions", Interantional Scientific Conference „Strengthening the Competitiveness and Economic Bonding of Historical Banat – SCEBB", Vršac, Serbia, pp. 255-260, 15th September, 2011.

Ilja Stanišević received his B.Sc. diploma from the Faculty of Electrical Engineering, University of Sarajevo. He received his M.S. degree from the Faculty of Electrical Engineering, University of East Sarajevo and Ph.D. degree from the Faculty of Computer Science in Belgrade. He works as a Professor of applied studies in the Academy of Applied Studies Western Serbia. His interests include information systems development, databases and programming. He participated in many commercial ICT projects.

Mlađen Vićentić achived his Diploma and M.S. degree from the Faculty of Economics, University of Belgrade. He received his Ph.D. from the Faculty of Economics, University of Kragujevac. He works as a Professor of applied studies and as a department manager in the Academy of Applied Studies Western Serbia, Department in Valjevo. His interest of research includes data exploration and analysis, methods of regression and correlation analysis.

Slobodan Obradović received his Diploma and M.S. degree in electrical engineering from the School of Electrical Engineering, University of Belgrade. Ph.D. degree from the Faculty of Electrical Engineering, University of East Sarajevo. He works as a Professor in the Information Technology School ITS in Belgrade.