

DBMS RESPONSE SPEED TESTING SYSTEM

Stanišević Ilja

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

Slobodan Obradović

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

Mladjen Vičentić

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

Abstract

During the development of the information system for Valjevo Business School of Applied Studies it was necessary to evaluate available DMBS alternatives. Instead of using some of existing banchmark software tools, the development team has built a simple software adapted to test features relevant for the school's system. This paper describes applied methodology, design, development and evaluation of the realized response speed testing software.

Keywords: Benchmark, Performance evaluation, Database management systems, Microsoft SQL Server, MySQL, MS Access

INTRODUCTION

The need for objective database management system evaluation had emerged during realization of the information system for Valjevo Business School of Applied Studies. Instead of using some of existing benchmark tools it was decided to develop a software tool for measuring response speed during execution of various SQL commands. This paper describes criteria determination, development and functionalities of the realized testing tool.

It was necessary to perform adequate measurements to provide objective evaluation of the tested DBMSs. One approach is to apply some of many existing benchmark tools. Good and high-quality benchmark system for testing database management systems, according to Gray [1] should have 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 DBMSs;
- 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 (desirable all) computer systems and architectures, large as well as small.

These characteristics, by its nature, are frequently contradictory. For instance, feature of simplicity is opposite to portability feature. Therefore, it is necessary to establish adequate compromise among these characteristics [2].

Many factors can have an impact on the measuring outcome. The results can be depended on hardware (i.e. CPU speed, number of cores, memory access speed, the amount of RAM memory in the system, bus speed, hard disc drive speed etc.), on system software (the way operating system deals with the memory, threading or locking), on data schema, on amount of previously recorded data, on database access application, on database configuration (cash amount dedicated to queries, limit of established client connections to the tested database, the way of index implementation, network protocols used for database access...) [3].

Development of database technology led to development of number of measuring software tools (benchmarks) which test and evaluate various database features. Therefore, we have

transaction processing benchmarks, benchmarks for relational databases, for object-oriented databases, XML based databases benchmarks, decision support systems (DSS) benchmarks, benchmarks for non-SQL databases, for cloud databases [2] [4]. There is a separate organization dealing with relational databases benchmark standards, Transaction Processing Performance Council – TPC. The organization issues standards and verifies correctness of benchmark tools [4].

There are many database benchmarks on the market today, like Quest [5], STS Soft [6], Hammer DB [7], etc. Many of these tools are open source and freeware, therefore they can be used without a need for additional funding. Anyway, there is a significant problem related to reliance on these tools. Most of them are created and designed to be generally applicable. As a result, the tools test and evaluate many features irrelevant for our purpose. For instance, our system has only few input points, therefore performance of dealing with many users is not relevant. The system is planned to be used in the local area network, so internet performance is of no importance.

There are distinct benchmark tools intended to evaluate special kind of databases [8], as well as tools which evaluate particular database features [9] but neither of these is appropriate for determination of the best DBMS for the school system, due to its narrow applicability domain.

Taking all of the above in consideration, the development team has adopted a different strategy. Instead of using pre-made general purpose benchmark tool, the team has developed a separate testing software to objectively evaluate features relevant for the school system. Furthermore, the software will be developed and implemented in the exactly same hardware and software environment as the school system.

Of course, the testing tool like this would not fulfil requests for portability and scalability according to Gray [1], but these features have no importance for our purpose. On the other hand, the tool would at the most fulfil request for relevance (as being designed for particular system and environment) and simplicity (only features relevant for the school information system would be tested).

This will provide that the tests will show the most adequate DBMS for the school's system.

TECHNICAL PRESUMPTIONS

Information system for Valjevo Business School of Applied Studies was developed for exploitation in the school's local area network (LAN), according to client-server model. The network consists of four servers (domain control server, database server, internet access control server and back-up domain control server) and approximately 120 workstations. The servers run under Windows 8 server operating system. The workstations, due to different age of the stations (some are quite new, but some are almost obsolete), work on different operating systems, i.e. Windows 10, Windows 7 and Windows XP, but they all have .NET framework 3.5 installed.

The testing tool was realized using C#.NET programming language. The programming environment was MS Visual Studio Community 2017, version 15.9.14. The school information system was developed in the same environment using the same programming language. Due to compatibility the .NET Framework version 3.5 was adopted as a software framework since it can be installed and run to all school's computers, regardless operating system version and hardware obsolescence.

The school management has decided for security and control reasons that system should be available only through the local area network, i.e. access through internet would not be supported. Since there is domain control server which controls user's roles, prerogatives and access rights, the school system can rely on security primitives provided by the operating system.

The school information system should provide functionalities to support following activities:

- all activities performed by student service office (i.e. student's registry, enrollment track, exam's registration, issuing various certificates, etc.);
- lectures organization activities (management of classes, lectures record, track of attendance, records of the realization of the curriculum during

a semester as well as during a school year, etc.)

- mentor activities (success monitoring of the students for various subjects, departments, sections, school years, tracking attendance at the exams individually as well as for different groups of students, etc.);
- activities performed by the human resources department (registry of employees, track of staff attendance, etc.).

From the list of required functionalities can be noticed that the system will be less burdened by inserting and updating data. The emphasis will be put on retrieving data and generating various reports. Most of data entry activities on everyday basis will be performed from one spot at the student service office. On the other hand, data search activities will be performed by all users of the system.

All activities on the system will be happening in real time (many of them on the desk window). Therefore, the rapid response of the system is required and the most important criterion for the database management system is its speed.

DESIGN OF THE SYSTEM

The conceptual design of the testing system is displayed on Figure 1.

The topmost layer is the user interface. It was built using Windows Forms library, and controls provided within that library. The reason for choosing the WinForms library (instead of, for instance, Microsoft Presentation Foundation – MPF as a graphical subsystem for rendering user interface) is the fact that WinForms is used for development of the user interface in the school information system, so this option is more suitable for our purpose. Furthermore, the most common controls are used (i.e. text boxes, buttons combo and list boxes) which makes the interface user friendly and intuitive for a user. He can be trained to use the system in a very short time.

The user interface enables the user to define parameters for basic SQL selection and editing commands. He enters only specific parts of a command, such as field names, logical operators or constant values (e.g. “name =

‘Peter’ and year = 2019”). All other parts of the command are automatically generated by the software. This solution speeds up the whole measurement process.

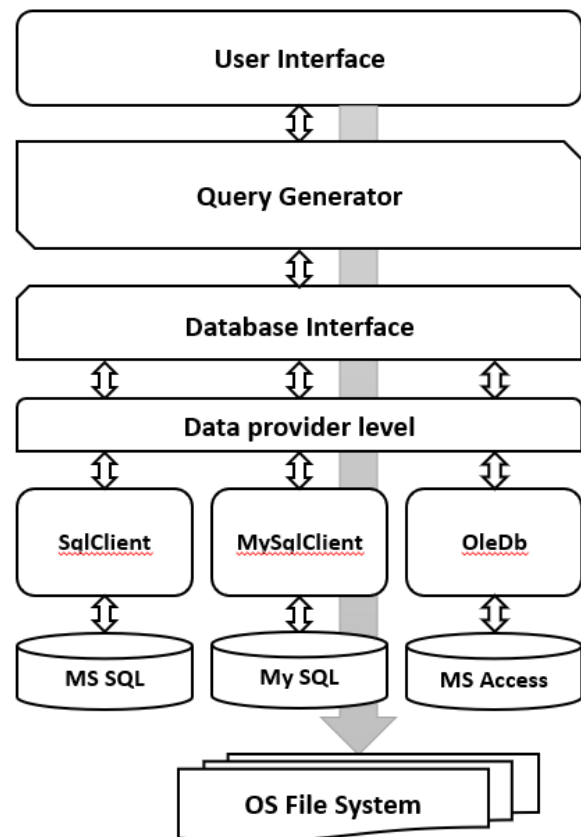


Fig. 1. The testing system design

The user has two additional options. One displays results of query/content of the database (depends on previous action) in a database view control. The other one deletes all data in all tables and clears the database, which is convenient for repetitive measurements. At the end, the user interface can invoke saving content of the list box where results are displayed in a separate file, by sending a request to the native operating system file system.

User interface designed in this way is simple and intuitive for usage, but yet has an option for all available functionalities. Interface form containing all functionalities is presented in Figure 2.

Interrupt caused by a click on some of the main command buttons invokes the query generator. It creates adequate SQL command, depending on the values entered in the check and text boxes on the user interface form. The

queries consist of a basic command (i.e. SELECT, INSERT, UPDATE, DELETE) and an optional part (i.e. JOIN and/or WHERE clause defined by user).

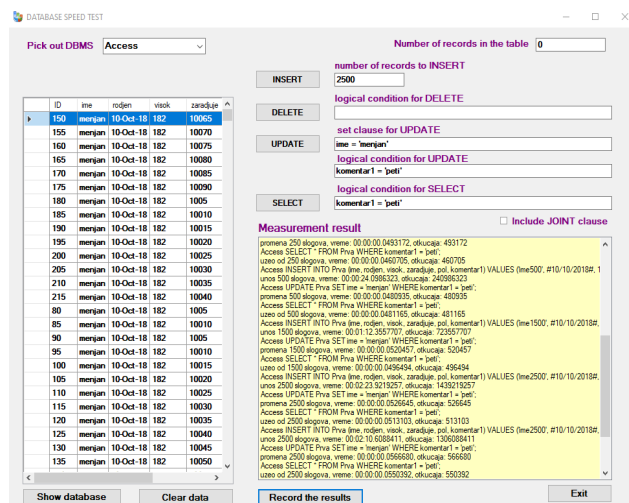


Fig. 2. User interface form

The query generator transfers the queries to the database interface. Depending on the chosen DBMS, the database interface calls the adequate method appropriate for the chosen database. This layer performs a simple parsing and takes care of syntax differences among SQL dialects used by tested DBMSes (for instance, the syntax to indicate data/time values, or to include necessary brackets where requested).

There are global variables in the system which contains values of separate connection strings for each tested database. The database interface then invokes appropriate method and sends the query as a parameter. The method then establishes connection to the wanted DBMS using corresponding data provider, sends a request for execution of the received query, picks up a result, closes the connection, and finally, if the query was for SELECT command, it fulfills the database table with the query result (i.e. received relation). The user can then by clicking the appropriate button see the resulting data table in a DataGrid View control on the user interface form.

This design enables relatively simple extension to include additional DBMSes. All that is required is reference to the adequate data provider and writing the corresponding method within database interface.

Data providers, each for a single database, receive requests from database interface and send them to the right DBMS for the execution.

The query and response time are recorded in a list-box on the main interface form, and the next query can be performed. Of course, this approach measures not only response time of the DBMS but the time needed for software processing of a query within the testing system as well. In ordinary benchmark tools this would be a serious deficiency, but in this case, it is an advantage since the same methods are designed to be used by the school information system for accessing the database. Therefore, this approach provides us a more adequate evaluation of the tested database management systems.

Finally, at the end of a session, all results can be recorded in a text file on a hard disk drive for further processing by simple click on the corresponding button.

THE SYSTEM FUNCTIONALITY

The testing software was intended to be user friendly as much as possible, but still to keep all required functionalities.

In the beginning, a user is able to choose the wanted DBMS (i.e. MS SQL Server, MySQL, Access) by simple selection of the appropriate combo-box option. Next, the user has option to choose among four basic SQL commands (i.e. SELECT, UPDATE, INSERT and DELETE). Optionally, he can activate automated filling of the tables with given number of rows by simple entering desired number of rows. The user can include logical condition in generated SELECT, UPDATE or DELETE query by simple entering content of WHERE clause. The clause will be then automatically added into the query. Additionally, he can define fields editing by entering SET clause when UPDATE query is to be performed.

Finally, since the relational database defined by the school system project is strictly normalized and is in Boyce-Codd normal form [10], it is important to evaluate DBMSs response speed for SELECT queries containing JOIN clause.

By checking the appropriate check-box on the form, user can initiate addition of a JOIN clause into the query. There are two tables and a multiple functional dependency (i.e. M: N relation) between them, so the third table, the link table, is used. In case of checked check-box, the testing system automatically loads generated data into the secondary table and establishes relation between depended tables by filling the auxiliary table. The JOIN clause is added to the generated query, so the response to the query when all three tables are involved can be measured. Described database schema is presented in figure 3.

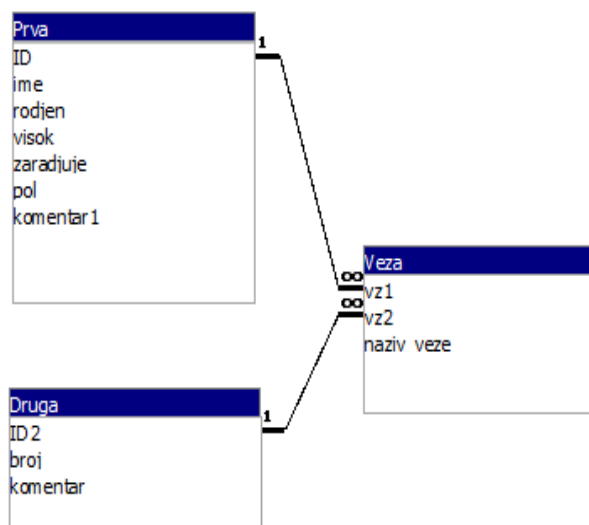


Fig. 3. The testing database schema

This structure was adopted for the test database as the most generic one, i.e. all other cases can be derived from this structure. Of course, the school information system will contain much complicated structure, but, at the end, it always can be reduced to the set of structures presented in Figure 3.

Response records containing the executed query, number of records and duration of execution are automatically added into the list box control. At the end of a session by clicking the corresponding button on the form, content of the list control can be saved in a separate file on the hard disk drive, and used later in additional processing of the results.

CONCLUSION

By realization of this testing system, we have got a tool to evaluate different DBMSes according to the specific requirements of the

software system we were going to develop. The testing tool was not intended for general evaluation of tested DBMSes but to provide a guideline that lead us to choose of the most suitable database management system for our particular need.

It can be concluded that this tool completely answered the task. The development team had the opportunity to test the acceptable alternatives regarding DBMS selection in the real environment and to, according to the objective results, make justified decisions

REFERENCE

- [1] Gray, J. (ed.). "The Benchmark Handbook for Database and Transaction Processing Systems 2nd edition". Morgan Kaufmann, 1993;
- [2] Darmont, Jérôme "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] Darmont, Jérôme, "Data Processing Benchmarks", Encyclopedia of Information Science and Technology, Third Edition, pp.146-152, 2014
- [5] Quest, Benchmark Factory For Databases, available at: <https://www.quest.com/products/benchmark-factory/>
- [6] STS Soft, Database Benchmark, available at: <http://stssoft.com/products/database-benchmark/>
- [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.