Comparative Analysis of MariaDB's Performance Efficiency as a Suitable Replacement for MySQL

Michael Witham, Isaiah Bender, Rahul Gomes
Department of Math & Computer Science
Minot State University
500 University Avenue W, Minot, ND 58707
michael.witham@ndus.edu, isaiah.j.bender@ndus.edu, rahul.gomes@ndus.edu

Abstract

This paper investigates the viability of MariaDB as a replacement for MySQL. A few major components of the two relational database management systems (DBMS) are compared. Their insertion and query speeds are tested using a large dataset and setting up two different experimental test benches. The Centers for Disease Control's U.S. Chronic Disease Indicators dataset was used. We selected this dataset due to its size and complexity as it had 519,718 rows and 34 columns. Identical MySQL environments were setup on both systems using XAMPP and a replacement MySQL package. The MariaDB environment was setup using an installer from MariaDB's website.

1. Introduction

Any data-oriented operation requires a database management system. From a local business storing just a handful of user's credentials to a large-scale hospital network with millions of medical records, a database management systems (DBMS) must meet the needs of its environment. While selecting a DBMS, most businesses consider MySQL. It is an attractive option due to its low cost. It's fast, easy to use, and supports structured query language [1]. Fewer will investigate the viability of MariaDB, MySQL's community-developed fork. Both have nearly identical features as MariaDB is based off MySQL's source code and built by some developers of the MySQL team as a drop-in replacement for MySQL [2]. However, MariaDB does not garner the same level of popularity as MySQL.

Storing and manipulating data has always been of massive importance since the earliest days of computer systems [3]. Overtime the modern database as we know it would develop out of the need for fast and efficient applications and systems that could meet the needs of the growing digital world. Databases find use in every aspect of the modern world and serve as the backbone of the Internet. Every website a person goes to or any facility that heavily utilizes technology has at least one database behind the scenes managing the massive amounts of data involved in the everyday operations [4]. Without an easy-to-use, fast, and reliable DBMS the World Wide Web as we know it would be completely different. [5]

The intent of the research conducted for this paper was to look at two relational database management systems (RDBMS) and compare their performances. MySQL and MariaDB were selected for a couple of major reasons. First, being that they are both wildly popular free and open source platforms (for our purposes). Second, is that MariaDB is a community developed offshoot of the MySQL source code and our thought was that there had to be a reason the MariaDB RDBMS exists despite the existence, popularity, and massive developmental backing of MySQL. In addition to comparing the performance of both MySQL and MariaDB we also decided to test the performance differences between the two environments on different sets of hardware. The idea was that either or both RDBMS' might show clear performance gains that were dependent on the hardware of the systems they were deployed on.

2. Background

When comparing two DBMS' it is unlikely one will find any two databases more similar than MySQL and MariaDB. This section of the paper delves into the background of both and what exactly spurred on the development of both database systems.

2.1 MySQL

MySQL was originally developed in 1994 and officially released in mid-1995. It was made because the creators thought mSQL (based on ISAM) was too slow and not flexible enough. They created a new interface but kept the same API as mSQL [6]. Development took a long time but eventually it was integration with another's work on InnoDB storage engine that finally made MySQL viable for mainstream use [7]. MySQL is a free yet full-featured relational database. [8] MySQL is also one of the most widely used database managements systems in the world with over 80% of websites utilizing it [9]. MySQL acquired Sun Microsystems in 2008, and in 2010 Oracle acquired Sun Microsystems. The same day Oracle purchased Sun Microsystems, one of the original creators of MySQL left and took many developers with him to create his own fork of MySQL called MariaDB. MySQL's open source nature [10] eventually lead to its source code being used for the development of MariaDB.

Some major features of MySQL are: Query caching, secure sockets layer (SSL) support, data definition language (DDL), nested SELECT's, Unicode support, embedded database library, performance schema that collects and aggregates statistics about server and query performance, support for multiple storage engines, commit grouping to gather multiple transactions from multiple connections.

2.2 MariaDB

MariaDB was designed as a drop-in replacement for MySQL initially released in January of 2009. The developers of MariaDB are composed primarily of former developers of the original MySQL. Fearing the changes to come following Oracles acquisition of Sun Microsystems, and by extension the MySQL intellectual property, the original programming team for MySQL left to create the community driven fork MariaDB. Due to it being built on the foundation of MySQL the commands, libraries, and databases are compatible between MySQL and MariaDB relational databases. The API was once again carried over from MySQL.

The major features of MariaDB are the same as the above features for MySQL with the added capability to simply replace MySQL in most environments where it is utilized. Since MariaDB is so similar most of the applications for MySQL work with MariaDB without any modifications required. [11] MariaDB also includes geographic information systems (GIS) and JavaScript object notation (JSON) features as well as performance gains across the board. The current MSI package for MariaDB also contains HeidiSQL which is an open source graphical client that you can use to interact with MariaDB. The developers for MariaDB worked with those of HeidiSQL to make sure it supports all MariaDB features and options. [12] MariaDB remains an opensource platform flaunting its adoption by industry titans such as Google, WordPress and Wikipedia.

3. Experimental Setup

3.1 Dataset

The dataset used for this project is one by the Centers for Disease Control and Prevention, called “U.S. Chronic Disease Indicators. [13]” It contains 519,718 rows and 34 columns of data. The database only has one table, so our testing is limited, but it should still be able to show differences in speed. The attributes in the table are shown below:

Year Start, Year End, Location Abbreviation, Location Description, Data Source, Topic, Question, Response, Data Value Unit, Data Value Type, Data Value, Data Value Alt, Data Value Footnote Symbol, Data Value Footnote, Low Confidence Limit, High Confidence Limit, Stratification Category 1, Stratification 1, Stratification Category 2, Stratification 2, Stratification Category 3, Stratification 3, Geo Location, Response ID, Location ID, Topic ID, Question ID, Data Value Type ID, Stratification Category ID 1, Stratification ID 1, Stratification Category ID2, Stratification ID2, Stratification Category ID 3, Stratification ID 3.

3.2 System Specifications

System 1 is a Windows 10 machine with an Intel i7 4790k central processing unit (CPU) with a slight overclock running at a frequency of 4.4GHz. It is a 4 core, 8 thread processor with 8MB of L3 cache, 156KB of L2, and 64KB of L1. The system is equipped with 16GB of double data rate (DDR) 3 random access memory (RAM) operating at 1866MHz in dual-channel mode. Storage was two traditional SATA 3 solid state drives in redundant array of independent disks (RAID) 0.

System 2 is a Windows 10 machine with an Intel i7 8700k CPU with once again a slightly overclocked frequency of 5.1GHz. It is a 6 core, 12 thread processor with 12MB of L3 cache, 256KB of L2, and 64KB of L1. The system is equipped with 16GB of DDR4 RAM operating at 2366MHz in dual-channel mode. Storage was a 5400RPM hard disk drive (HDD).

The commands to set up the databases were the same for both systems. In these examples we are assuming only one database is currently on the system.

```
> create database DBName;
> use DBName;
> CREATE TABLE data (YearStart INT, YearEnd INT, LocationAbbr VARCHAR(3), LocationDesc VARCHAR(40), DataSource VARCHAR(40), Topic VARCHAR(100), … , StratificationID3 VARCHAR(40));
> LOAD DATA LOCAL INFILE "CDCdatabase.csv" INTO TABLE DBName.data FIELDS TERMINATED BY ',' LINES TERMINATED BY '\n' IGNORE 1 LINES (YearStart,YearEnd, … ,StratificationID3);
```

These four lines above were executed in each database and both returned results for insertion times, which we will get to later in this paper. The commands listed are the actual commands performed except for some removed columns noted by "..."

4. Testing and Results

Commands were run a minimum of three times each in both the MySQL and MariaDB environments on both systems. In total four commands were executed for comparison of general query performance. Insertion times for both environments were also measured on both systems in both RDBMS environments. Differences in insertion times might be attributed to the differences in storage methodologies between the two systems. While system 1 had generally older, slower hardware than system 2, system 1's storage consisted of a striped solid-state storage array.

System 1's insertion times were only slightly different when comparing MySQL and MariaDB. MySQL was able to insert the full dataset in and average 33.26 seconds. MariaDB inserted the same dataset in a slightly faster average of 30.137 seconds. Times were collected from the onscreen output from both database systems. Dataset insertion was tested 3 times but in one instance MariaDB took 105.435 seconds. A fourth test revealed a time within one second of the average. MariaDB was 3.123 seconds or 9.39% faster on average at data insertion. Figure 1 provides a visual representation for the differences in times between both environments and shows that MariaDB while only slightly faster is clearly starting to take the lead.

System 2's insertion times were very different. MySQL inserted the full dataset with an average time of 50.62 seconds. MariaDB inserted the same set of records with a substantially lower average time of 22.156 seconds. Both times were reported back by the respective database running the command. Figure 2 shows that MariaDB provides a visual representation to just how much faster the insertion times on system 2 were for MariaDB when compared to MySQL.

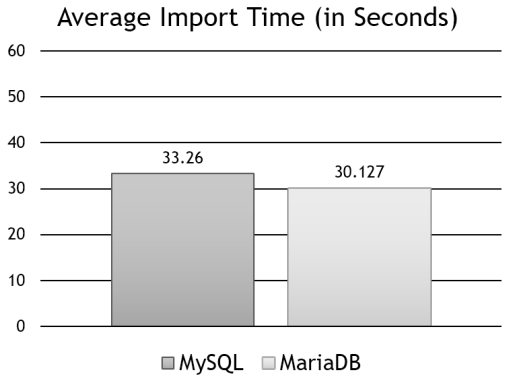

Figure 1: System 1 Import Times

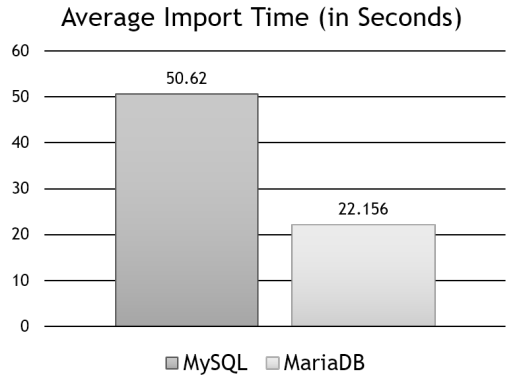

Figure 2: System 2 Import Times

4.1 Command 1 Results

Command 1 was *SELECT * FROM data ORDER BY YearStart DESC*; This command is built around the SELECT statement which is easily one of the most important commands in a structured query language (SQL) environment [14]. This is the methodology by which SQL databases are queried for data and by which it returns. The asterisk is used to indicate that all records from the data table should be returned by the query and that data should be sorted in descending order by their YearStart attribute.

4.1.1 System 1 Results

The results for command 1 on system 1 showed that the operation times were near identical between both MySQL and MariaDB. For MySQL the return times were 2.75, 2.81, and 2.81 seconds which averaged out to 2.79 seconds. MariaDB by comparison had return times of 2.784, 2.916 and 2.800 which averaged out to an operation time of 2.833 seconds. MySQL had a 1.54% faster descending sort query with a difference of 0.043 seconds. Figure 3 provides a visual on the actual differences in query times between MySQL and MariaDB.

4.1.2 System 2 Results

The results for command 1 on system 2 showed a very different outcome compared to system 1. MySQL returned times of 4.40, 3.79, and 3.88 seconds. With an average of 4.0233 seconds. MariaDB returned times of 1.099, 1.043, and 1.049 seconds. With an average of 1.0637 seconds. MariaDB was 116.36% faster than MySQL, with a difference of 2.9596 seconds. Figure 4 gives a visual representation of the query average time differences between both MySQL and MariaDB with MariaDB clearly outperforming MySQL.

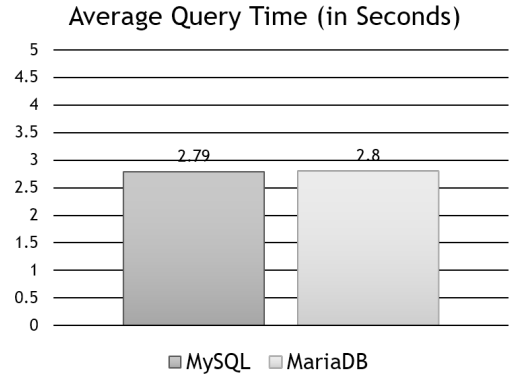

Figure 3: System 1 Command 1 Times

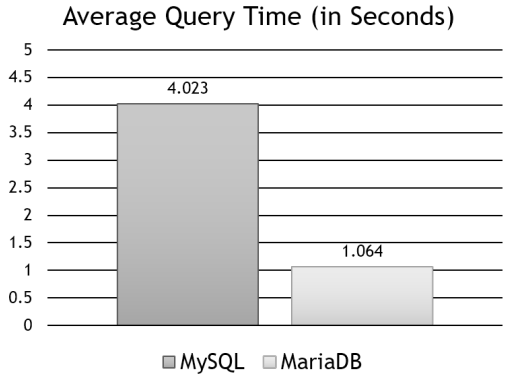

Figure 4: System 2 Command 2 Times

4.2 Command 2 Results

Command 2 was *SELECT * FROM data ORDER BY YearStart ASC*; This command is nearly identical to command 1 with the exception that the data is sorted in ascending order rather than descending. No real differences in time were expected although if there were any it might indicate a different handling of sorting methodologies.

4.2.1 System 1 Results

The results for command 2 on system 1 once again showed near matching times. For MySQL the operation times were 2.86, 2.79, and 2.83 seconds with an average time of 2.827 seconds. MariaDB queries returned times of 2.859, 2.757, and 2.767 seconds with an average time of 2.794 seconds. MariaDB was 1.79% or 0.033 seconds faster with the ascending sort query on a simple attribute. Figure 5 demonstrates the differences between the two environments.

4.2.2 System 2 Results

The results for command 2 showed similar results as command 1, with MySQL returning times of: 4.08, 4.11, and 3.77 seconds with an average of 3.9867 seconds. MariaDB had times of 1.042, 1.052, and 1.046 seconds with an average of 1.0467 seconds. For command 2 MariaDB was faster by 116.81% or 2.9403 seconds. Figure 6 demonstrates the difference in performance on this command.

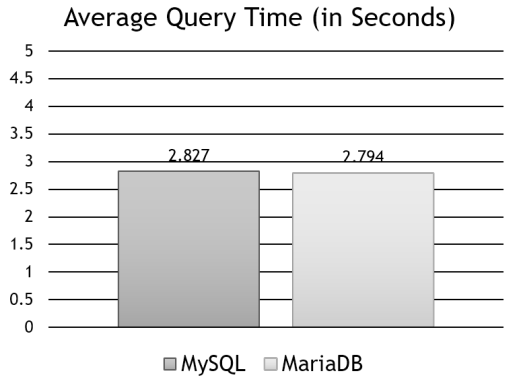

Figure 5: System 1 Command 2 Times

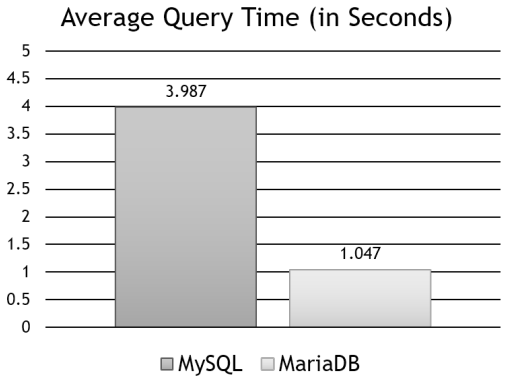

Figure 6: System 2 Command 2 Times

4.3 Command 3 Results

Command 3 was *EXPLAIN data*; This was a command that gave a summary of exactly how the MySQL or MariaDB query optimizers would choose to run the command following the explain keyword [15]. Due to the syntax used for our command the

expectation would have been an immediate return since it would simply be checking table existence from which we would take command runtime.

4.3.1 System 1 Results

The results for command 3 on system 1 showed different times. For MySQL the runtimes were 0.01, 0.01, and 0.01 seconds with an average of course of 0.01 seconds. MariaDB return times were 0.008, 0.006, and 0.008 seconds with an average operation time of 0.007 seconds. MariaDB was 42.86% or 0.003 seconds faster than MySQL. Figure 7 provides a visual of the two environment average times. While times were close it is hard to claim that MariaDB was indeed faster than MySQL due to limitations of the MySQL environment that only allowed measurement to a hundredth of a second whereas MariaDB was able to measure to a thousandth of a second.

4.3.2 System 2 Results

Command 3 for MySQL had times of 0.02, 0.01, and 0.01, with an average of 0.0133. MariaDB showed times of 0.095, 0.009, and 0.008, with an average of 0.0373 seconds. Command 3 only showed a difference of 94.86%, or 0.024 seconds. Figure 8 provides a visual representation of the average runtimes of command 3 showing MySQL to be significantly faster.

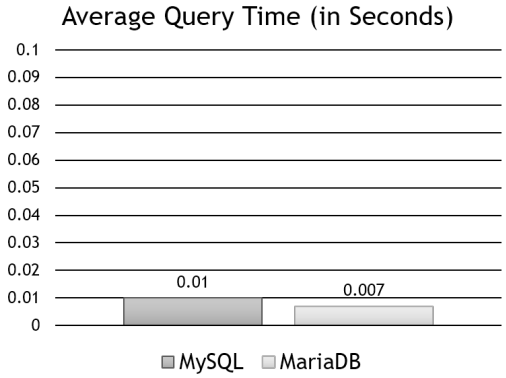

Figure 7: System 1 Command 3 Times

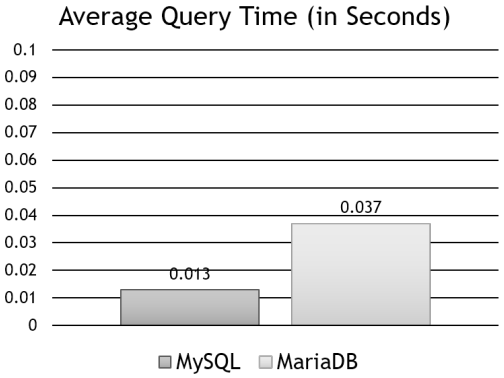

Figure 8: System 2 Command 3 Times

4.4 Command 4 Results

Command 4 was *SELECT * FROM data ORDER BY GeoLocation ASC*; Similar to the first two commands again we returned to measuring the query times for a select statement. The most commonplace operation for a database. GeoLocation was a more complex attribute than the YearStart attribute that was used in the previous commands. An ascending sort was also conducted on the operation.

4.4.1 System 1 Results

The results for commands 4 on system 1 showed the largest variation in runtimes. MySQL times were 21.86, 21.63, and 21.75 seconds with an average of 21.747 seconds. MariaDB times were 17.856, 17.856, and 17.775 seconds with an average operation time of 17.739 seconds. MariaDB was 22.59% or 4.008 seconds faster at ascending sort query on a more complex attribute. Figure 9 displays the differences in performance.

4.4.2 System 2 Results

Command 4 is where the benefits of MariaDB really start to show, especially on a system using traditional HDD's. MySQL had times of 65.02, 62.62, and 62.46 seconds, with an average time of 63.3667 seconds. MariaDB on the other hand was able to execute the command in just under 2 seconds each time, with times of: 1.209, 1.200, and 1.202 seconds. MariaDB had an average time of 1.2037. MariaDB blew MySQL away on this query with a query speed 192.54% faster than MySQL. They had a difference of over one minute, at 62.163 seconds. Figure 10 displays just how extreme the differences in performance were.

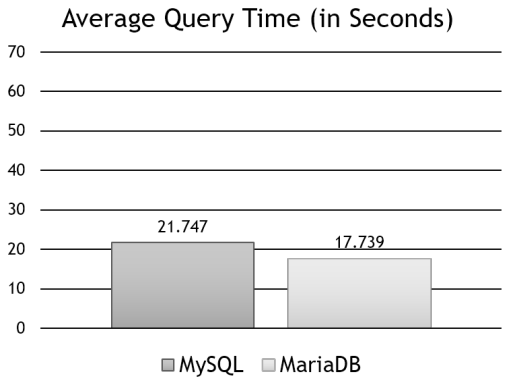

Figure 9: System 1 Command 4 Times

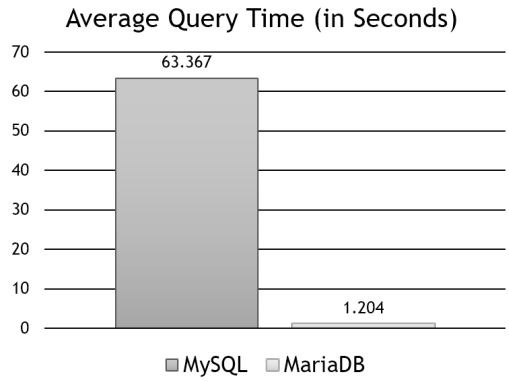

Figure 10: System 2 Command 4 Times

5. Discussion

The creation of the two databases were very similar and very easy. Since MariaDB was built by the creators of MySQL using the MySQL source code the commands were identical for creating the databases. To start MySQL, you first need to run the command prompt, then navigate to the folder where it was created and enter the command to start MySQL then enter your password and run your commands. In MariaDB they give you a shortcut that automatically sends you to the installation location so all you need to type is your password when prompted.

From an installation standpoint both were easy to setup. Installing MariaDB was as simple as downloading the package from their website, running the installer, selecting desired

features, and launching the program. MariaDB comes with its own database environment that can be worked with immediately. Simply launch its command line utility and provide necessary credentials all this without the need to launch a server. Installing MySQL was more difficult. Initially, one downloads a MySQL package from the Internet. Following that, a server is necessary for MySQL to become functional. For our purposes we downloaded XAMPP and launched its included MySQL server. From there we opened windows command prompt and navigated to the unzipped folder where we were able to launch the MySQL application. It should be noted that the SQL database that XAMPP comes with, despite claiming to launch MySQL, is in fact MariaDB, hence the need for us to download a separate MySQL package. As far as what was easier to install the winner seems to be MariaDB due to its simplicity to both launch and install.

From a query speed standpoint, the clear winner was MariaDB. MySQL and MariaDB performed very comparably on older hardware with the two only undercutting each by tenths of a second and not even consistently enough to establish a trend. In the test of querying data sorted by GeoLocation MariaDB did manage to outperform MySQL by a fairly large margin. On newer hardware MariaDB truly shined, outperforming MySQL in every test by a large margin. Regarding insertion speed MariaDB seemed to be more optimized on both new and old hardware beating out MySQL. MySQL had only slightly slower read times than MariaDB on older hardware running from a solid-state array but on newer hardware with a slower spinning disk hard drive MariaDB blew MySQL out of the water with insertion times nearly twice as fast. Rather or not this was some benefit brought about by the processor or faster RAM is unclear, what is clear is that MariaDB is clearly taking advantage of some aspect of newer system hardware than MySQL is.

Experiments show that MariaDB performed better if not identical to MySQL. MariaDB mostly outperformed MySQL by a margin of 9.39% to 42.86% on insertion operations based on the size and test benches. Query efficiency ranged from 1.54% to 192.54% faster than MySQL based on the test benches and query parameters. Results show that MariaDB performs better than MySQL and should be used in querying and managing larger datasets.

The conclusion of our testing reveals that there is nothing to lose by upgrading to MariaDB but everything to gain. For older systems that may choose to forego upgrades they are likely not sacrificing anything in terms of performance. For newer systems there is only performance to be gained by upgrading to the simple replacement MariaDB.

6. Conclusion

Our experiment was limited in scope and while the results pulled are certainly meaningful in the future we would like to perform testing on a database with a higher-degree of complexity. With a database consisting of multiple complex tables we would have the opportunity to test additional common commands such as unions and intersections.

References

[1] DuBois, P. (2008). *MySQL*. Pearson Education.
[2] Bartholomew, D. (2012). Mariadb vs. mysql. *Dostopano*, *7*(10), 2014.
[3] Ramakrishnan, R., & Gehrke, J. (2000). *Database management systems*. McGraw Hill.
[4] Bartholomew, D. (2013). *Getting started with MariaDB*. Packt Publishing Ltd.
[5] Cabral, S. K., & Murphy, K. (2011). *MySQL administrator's bible*. John Wiley & Sons.
[6] Widenius, M., Axmark, D., & Arno, K. (2002). *MySQL reference manual: documentation from the source*. " O'Reilly Media, Inc.".
[7] Pachev, A., & Pachev, S. (2007). *Understanding MySQL Internals*. " O'Reilly Media, Inc.".
[8] Davis, M. E., & Phillips, J. A. (2007). *Learning PHP & MySQL: Step-by-Step Guide to Creating Database-Driven Web Sites*. " O'Reilly Media, Inc.".
[9] Dyer, R. J. (2015). *Learning MySQL and MariaDB: Heading in the Right Direction with MySQL and MariaDB*. " O'Reilly Media, Inc.".
[10] Watson, R. T., Boudreau, M. C., York, P. T., Greiner, M. E., & Wynn Jr, D. (2008). The business of open source. *Communications of the ACM*, *51*(4), 41-46.
[11] Razzoli, F. (2014). *Mastering MariaDB*. Packt Publishing Ltd.
[12] Bartholomew, D. (2014). *MariaDB cookbook*. Packt Publishing Ltd.
[13] data.cdc.gov/Chronic-Disease-Indicators/
[14] Welling, L., & Thomson, L. (2003). *PHP and MySQL Web development*. Sams Publishing.
[15] Schwartz, B., Zaitsev, P., & Tkachenko, V. (2012). *High performance MySQL: optimization, backups, and replication*. " O'Reilly Media, Inc.".