DATABASE REPLICATION: ENSURING MYSQL DATA INTEGRITY

Ivan ŠUŠTER
Darjan KARABAŠEVIĆ
Aleksandar ŠIJAN
Đorđe PUCAR
Luka ILIĆ
Goran JOCIĆ

Abstract: *In order to maintain stable, accessible data, database administration must prioritize efficiency, consistency, and fault tolerance. Data dissemination between servers is facilitated via database replication, especially in MySQL, which improves availability and readability. This paper explores the fundamentals of MySQL replication, emphasizing the configuration procedure that includes both the more recent Global Transaction Identifiers (GTID) and the more conventional replication through binary log files. It also introduces group replication, which guarantees high availability and fault tolerance in distant contexts, and provides helpful administration commands. The article examines MySQL replication kinds, contrasts replication and mirroring, and provides configuration instructions. It also offers information on GTID replication, including setup instructions. Additionally, it covers MySQL replication commands and goes over the advantages of group replication. All things considered, MySQL database replication is essential to guaranteeing data availability, scalability, and integrity in contemporary database systems.*

Keywords: *MySQL Replication, Database Management, GTID (Global Transaction Identifiers), Fault Tolerance,*
High Availability

INTRODUCTION

In the context of databases, it is important to maintain efficiency, consistency and fault tolerance so that data is stable and easily accessible. Database replication plays an important role in this process, as it enables the distribution and synchronization of data across multiple servers, thus improving data availability and readability.

Focusing on MySQL replication, one of the leading database management systems, the basic principles of replication are presented, with an overview of the different types of replication available within the MySQL system (Šušter & Ranisavljević, 2023; Zmaranda et al., 2021). This is followed by an analysis of the MySQL replication configuration process, with steps for setting up traditional replication using binary log files, including setting up a primary (or master) and replica (or slave) server.

Additionally, the use of Global Transaction Identifiers (GTIDs) for replication is presented, highlighting the benefits of this approach. Furthermore, an overview of useful commands for managing MySQL replication is provided and the concept of group replication is explained. It is modern functionality within MySQL that provides a high level of availability and fault tolerance in distributed environments.

REPLICATION OF DATABASE

Database replication involves creating and maintaining copies of database objects on multiple servers in a distributed database system. Replication is a fundamental mechanism that improves scalability and fault tolerance in databases (Pohanka & Pechanec, 2020). It allows data to be available from different servers, enabling parallel processing of queries and load balancing (Kemme, Alonso, 2010).

There are different types of database replication mechanisms, such as master-slave replication, one-way data replication, full replication, partial replication, and scalable replication for transactional web applications. These mechanisms serve various purposes such as backup, reporting, data distribution, and ensuring data availability and reliability (Yadav et al, 2013).

Based on the previous definitions, it can be concluded that database replication is a very important concept in database management systems, which offers advantages such as scalability, resilience to server failure, etc. and therefore there are different mechanisms or types of replication to satisfy different requirements and ensure data integrity in distributed environments.

Another definition of replication would be that it is the process of copying a database from one server to one or more other servers to improve database availability, as well as database scalability and performance. This process also includes the synchronization of the „main" base and the replicas of that main base (Drake, 2021).

TYPES OF REPLICATION IN MYSQL DATABASE

In MySQL, replication is asynchronous, replicas do not have to be connected constantly to receive new changes from the source or main database, and depending on the configuration, it is possible to make replicas of all databases, selected databases or only selected tables from a database.

Some of the advantages provided by MySQL replication are (Replication):

• Scaling - spreading the load across multiple replicas to improve performance. In this environment, all writes and updates must be done on the source or master server. Reading from the database can also be performed on one of the replicas. This model can improve write performance (because the origin server is dedicated only to writes and updates), while greatly increasing read speed as the number of replicas increases (Milani & Navimipour, 2016).

• Data security - since the replica can pause the replication process, it is possible to start the backup process on the replica without conflicting with the existing data on the source server.

• Analytics - data can be added to the source server without interfering with or affecting the performance of the source server, while information analysis is performed on the replica.

• Remote data distribution - replication can be used to create a local copy of data to be used by remote locations without permanent access to the source server.

There are two types of replication in MySQL:

1. The traditional method - includes the use of a binary log file to replicate changes made on the master server to one or more slave servers. This method relies on recording changes to a binary log file and then repeating these changes on other servers to keep the data in sync. The way it works is that replica or slave server creates I/O thread which is used to copy data from the binary log file from the primary/master server to the relay log file on the slave server and than the SQL thread is used to execute the changes written in the relay log.

2. Replication with Global Transaction Identifiers (GTID) - is a more advanced and newer method that provides a unique identifier for each transaction on all servers in the replication topology. GTID simplifies the process of monitoring and managing replication, especially in complex multi-server replication setups. GTID assignment differentiates between source transactions and replicated transactions that are replayed on the replica. When a transaction is executed on the source server, a new GTID is assigned to it, provided that the transaction is recorded in the binary log file.

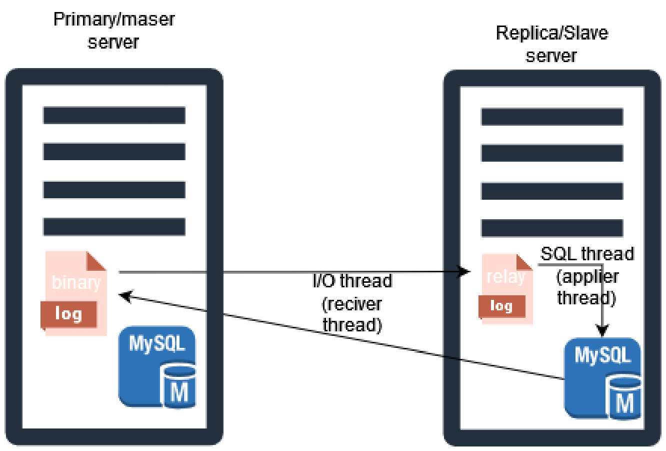


Figure 1 MySQL replication process

These two methods offer different approaches to database replication. The traditional method of using a log file is a well-established method that is widely used to replicate data changes between servers. In contrast, replication with GTID introduces a more efficient way of managing replication.

MySQL users can choose the approach that best suits their replication needs, whether it's the traditional method or GTID for improved transaction management and tracking on distributed servers.

MySQL also supports different types of synchronization during replication. The basic type of synchronization is one-way, asynchronous replication, where one server serves as the main or source server, while one or more other servers serve as replicas. And this type of replication in MySQL is the most used.

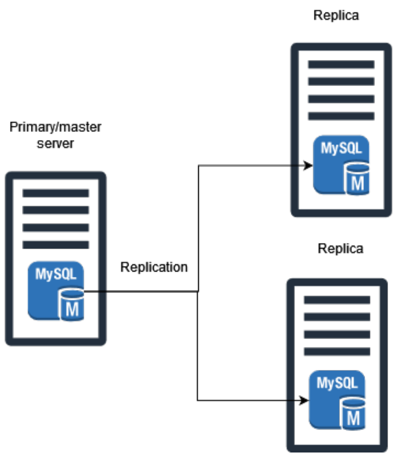


Figure 2 One way replication

In addition to asynchronous replications in MySQL, there are also synchronous replications that are characteristic of an NDB cluster. As well as semi-synchronous and delayed replication.

DIFFERENCES BETWEEN REPLICATION AND MIRRORING

While replication represents the process of copying data from the source server to the servers where the replicas are located, mirroring represents an identical copy of the database. The term mirroring means that changes made in the database are immediately mirrored and applied to copies or replicas.

In MySQL, with master-slave replication, the master or source server processes write and update operations and transfers changes to one or more replicas asynchronously, and this configuration provides advantages such as load balancing, but it does not guarantee data consistency in real time between the main base and the replicas.

On the other hand, mirroring prioritizes data consistency in real time. This means that when a change is made to the master database, it is immediately mirrored to the „replicas”, ensuring that all copies of the database remain identical at any time.

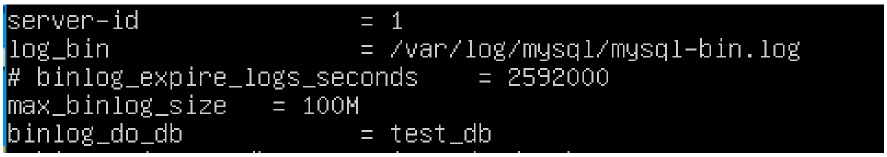
Essentially replication and mirroring share the goal of copying data to multiple servers, mirroring prioritizes synchronous replication so that data remains identical and available on any server but also increases network load. Because of this capability, it is a better way to back up data than asynchronous replication.

CONFIGURATION OF MYSQL REPLICATION

To configure the source or master server to use the binary log file for replication, we need to enable the binary log and assign the server a unique ID. In addition, it is necessary to configure the firewall so that the replica can communicate with the master server, this can be done with the command *sudo ufw allow from 192.168.1.105 to any port 3306*. It is important to note that each server located in the replication topology must have a unique ID which is a positive integer and is set using the server_id system variable.

The following MySQL command is used to set the server_id variable: *SET GLOBAL server_id = 1;* It is also necessary to set the name of the binary log file. These settings can also be done by changing the configuration file located at the location */etc/mysql/mysql.conf.d/mysqld.cnf* and it is necessary to change the following lines:

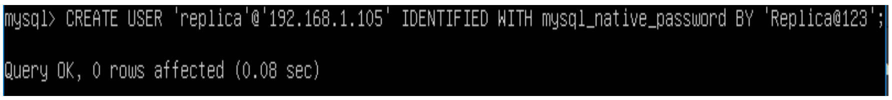
- bind-adress - the source server address is entered, it specifies the address from which MySQL will allow connections. On this line, instead of 127.0.0.1, enter 0.0.0.0.
- server-id - as already mentioned uniquely identifies the server. log_bin - the path to the binary log file as well as the name of the binary log file.
- binlog_do_db - enter the name of the database that will be replicated. If it is necessary to create replicas of several databases, add more lines with the names of the databases below this line. You can also specify the name of the database that will not be replicated using the binlog_ignore_db directive.



```
server-id               = 1
log_bin                 = /var/log/mysql/mysql-bin.log
# binlog_expire_logs_seconds    = 2592000
max_binlog_size   = 100M
binlog_do_db            = test_db
```

Figure 3 MySQL configuration file on master server

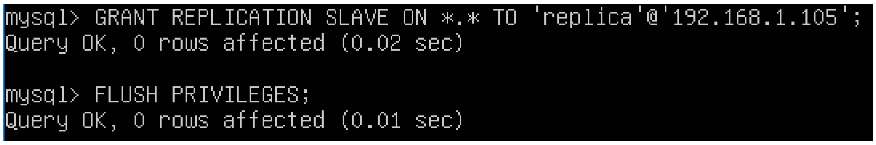
When all the changes have been entered, it is necessary to save the configuration file and restart the mysql service with the following command *sudo systemctl restart mysql.* When the MySQL server is up again, it is possible to log into MySQL using the command *mysql -u root -p* if we want to access MySQL as the root user.



```
mysql> CREATE USER 'replica'@'192.168.1.105' IDENTIFIED WITH mysql_native_password BY 'Replica@123';

Query OK, 0 rows affected (0.08 sec)
```

Figure 4 Creating a user on the master server who will access the server for replication



```
mysql> GRANT REPLICATION SLAVE ON *.* TO 'replica'@'192.168.1.105';
Query OK, 0 rows affected (0.02 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.01 sec)
```

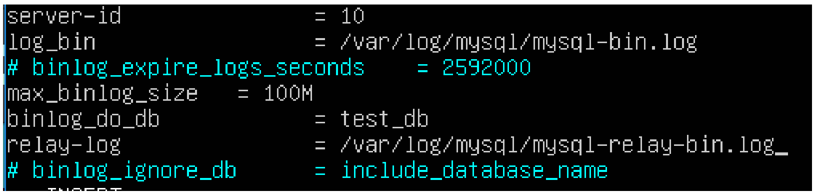
Figure 5 Grant of privileges for replica user

Furthermore, it is necessary to create a database, after creating the database it is necessary to execute using the command USE test_db; we choose the base with which we want to work, that is, where the table will be created and a new row added to the table. After that, it is necessary to create a user for replication, that is, a user that replication will use to monitor changes on the master server.

The created replica user in this case is used IDENTIFIED WITH mysql_native_password for user authentication, the default authentication mechanism would require the use of an encrypted connection between the source server and the replica. This option should be used in production.

When the user is created, it is necessary to grant him privileges to work with the database that needs to be replicated. In order to get the position of the binary file, i.e. the position for setting the slave server, it is necessary to execute the show master status command and remember the name of the binary log file as well as the position, after that, the master server setup process is complete.

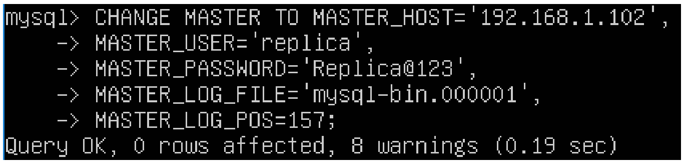
When we have made the settings on the master, it is necessary to set the replica or more if necessary. The first item when setting up the replica is to change the configuration file, which is located in the same location as the source server, ie. /etc/mysql/mysql.conf.d/mysqld.cnf, where you need to change the lines in the file it is important to enter a server-id that is not the same as that of the master server, as well as to add a new relay-log line indicating the location and name of the relay log file on the server. Another important note is that it is necessary to delete the auto.cnf file located at the /var/lib/mysql/auto.cnf location, the reason for deleting this file is that it serves to assign the UUID automatically, and since the virtual machine was cloned from installed MySQL server, an error will occur when starting the slave. After that, it is necessary to restart the MySQL service and log in to MySQL.



```
server-id                 = 10
log_bin                   = /var/log/mysql/mysql-bin.log
# binlog_expire_logs_seconds    = 2592000
max_binlog_size   = 100M
binlog_do_db              = test_db
relay-log                 = /var/log/mysql/mysql-relay-bin.log_
# binlog_ignore_db        = include_database_name
```

Figure 6 MySQL configuration file for replica

When all that is finished, it is necessary to execute the command that enables the replication to be connected to the master server, the command is shown in the picture below. For MASTER_HOST - the IP address of the master server is entered, MASTER_USER - the user that was created earlier on the master server, MASTER_PASSWORD - the password that was assigned to the user of the replica that was created on the source server, MASTER_LOG_FILE - the log file that the replica will monitor and from which it downloads changes to database, which is also specified in the MySQL configuration file and MASTER_LOG_POS - the position obtained earlier when the status of the master server was displayed.



```
mysql> CHANGE MASTER TO MASTER_HOST='192.168.1.102',
    -> MASTER_USER='replica',
    -> MASTER_PASSWORD='Replica@123',
    -> MASTER_LOG_FILE='mysql-bin.000001',
    -> MASTER_LOG_POS=157;
Query OK, 0 rows affected, 8 warnings (0.19 sec)
```

Figure 7 Master configuration in replica server

One of the notes is that instead of master and slave names, SOURCE and REPLICA can be used, so it could also be written as CHANGE REPLICATION SOURCE TO SOURCE_HOST, etc. After executing the previous command, the command to start the slave server START SLAVE can be executed and then the server status can be checked using the SHOW SLAVE STATUS command.

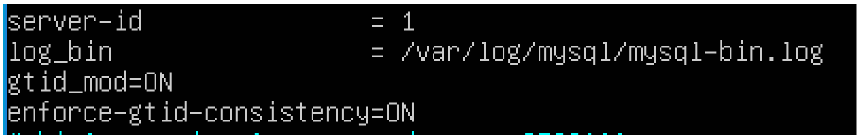
From the status, you can see some important information, for example, at which IP address is the master server, who is the user on the master through which we access the master, that the master server works on port 3306, which is the default port for MySQL, you can also see the name of the binary log file in which mentions are written, as well as the relay log file on the slave server. In addition, you can see two lines Slave_IO_Running and Slave_SQL_Running which represent the processes that are created on the slave server and which were discussed at the beginning that the IO thread serves to listen for changes in the binary file on the master and transfer them to the relay binary file on the slave u, while the SQL thread reads the changes from the relay binary and executes them on the slave server. Errors that occur can also be seen here. Also SQL_Delay represents the time after which the slave server will apply the changes that happened on the master, this was also discussed at the beginning of the work, these are delayed replications that can be configured also in the mysqld.cnf file.

When all the settings have been completed and it has been verified that the master and slave servers function without errors, replication testing can be performed, i.e. whether the changes are mirrored from the master server to the slave server.

GTID REPLICATION OF MYSQL DATABASE

A Global Transaction Identifier (GTID) is created and associated with each transaction that is successfully executed on the origin server. It ensures that every transaction can be easily tracked in the system. An example of setting up a simple master-slave replication using GTID will be shown below.

Similar to setting up the master server, to use the binary file it is necessary to modify the configuration file of MySQL (/etc/mysql-mysql.conf.d/mysqld.cnf). It is necessary to change bind-address = 0.0.0.0, add the line gtid_mode=ON and enforce-gtid-consistency=ON. Other changes are shown in the image below.

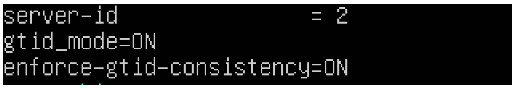


```
server-id                 = 1
log_bin                   = /var/log/mysql/mysql-bin.log
gtid_mod=ON
enforce-gtid-consistency=ON
```

Figure 8 MySQL configuration file for GTID replication on master server

After the change, restart the MySQL server and add a rule for the firewall so that the replica can communicate with the base using the following command *sudo ufw allow from any to any port 3306*, then open the MySQL prompt where it is necessary to create a user with privileges to read changes from the master server same as it was created in example with binary log files.

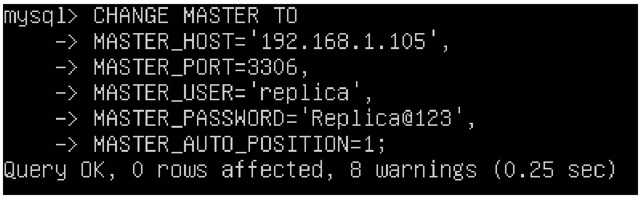
On the slave server it is also necessary to enable the GTID in the configuration file as on the master server (it is necessary to change the bind-address = 0.0.0.0) and it is important to change the server-id so that it is not the same as on the master server.



```
server-id                 = 2
gtid_mode=ON
enforce-gtid-consistency=ON
```

Figure 9 MySQL configuraion file for GTID replication on replica/slave server

After restarting the MySQL server, configure the replica in MySQL to use the master with GTID as the source for creating the replica, as well as to use the GTID automatic transaction position.



```
mysql> CHANGE MASTER TO
    -> MASTER_HOST='192.168.1.105',
    -> MASTER_PORT=3306,
    -> MASTER_USER='replica',
    -> MASTER_PASSWORD='Replica@123',
    -> MASTER_AUTO_POSITION=1;
Query OK, 0 rows affected, 8 warnings (0.25 sec)
```

Figure 10 Configuration of master server on replica server

This configuration entry, when MASTER_LOG_FILE and MASTER_LOG_POS are omitted means that the replica is configured to work with GTID, which also means enabling auto position. After the configuration, start the slave server with the START SLAVE command; the command to check the status of the slave server can also be started - SHOW SLAVE STATUS\G;

COMMANDS FOR WORKGIN WITH MYSQL REPLICATION

After the replicas are created, it is highly likely that knowledge of some of the commands used to manipulate replications in MySQL will be needed. Depending on the type of needs and requirements, it is important to have additional commands that can be used when working with replications.

Some of the commands have already been shown in the previous chapter when setting up replications, but it is possible to change them whenever there is a need to change some of the settings, for example the command CHANGE MASTER TO, which is used to set up the slave server, can be used later if it is necessary to change some of the parameters.

In addition, the START SLAVE command was also used; which starts the slave after entering the parameters via the previous command CHANGE MASTER, in addition to the START command, there is also a STOP SLAVE command for stopping the slave server, as well as for restarting, i.e. deleting all configurations on replication, the RESET SLAVE ALL command. These commands can be used in the case when it is necessary, for example, to upgrade a slave server to a master, where it is necessary to stop slaves, delete binary log files and change the configuration.

Some other useful commands that can be used in backup situations are STOP SLAVE IO_THREAD which stops reading changes from the binary log file on the master server. This can also be useful when the SQL thread needs to catch up and apply all the changes that have been read from the origin server by the IO thread up to that point. In addition, it is also useful when it is necessary to make some changes on the master server as well as on the replica.

It is also possible to stop the SQL thread with the STOP SLAVE SQL_THREAD command, which prevents the execution of changes that the IO thread transferred from the master server to the replica. The following two commands can be used to view the binary files, the first is to view the binary log file SHOW BINLOG EVENTS IN ,mysql-bin.000001';

which can be executed on the master server. Where you can see the name of the log file, the position, the type of change that occurred on the server or in the database, the identification number of the server on which the change occurred, as well as the end of the change position. Instead of this command, it is also possible to display the binary file using the command *sudo mysqlbinlog /var/log/mysql/mysql-bin.000001* where you can see the details of the binary file such as e.g. the time the change was made, one can also see the thread_id which thread made the change, exec_time the time the change was made, etc.

On the slave server, it is possible to display changes in the relay log file also using the mysqlbinlog command, given that these two log files have the same format, or it is possible to use the SHOW RELAYLOG EVENTS IN ‚mysql-relay-bin.000002' command; Another command that can be used in case replication stops working and it is necessary to recover replication is the command to skip the changes that happened on the master server and it can be executed only while the threads on the slave server are stopped, otherwise a SET error occurs GLOBAL sql_slave_skip_counter=2, which skips the next 2 changes that occurred on the master server.

These commands allow administrators to manipulate replicas from configuring replicas to monitoring replication status and troubleshooting. Effective use of these commands is critical to maintaining the reliability and efficiency of replication in MySQL.

GROUP REPLICATION IN MYSQL

MySQL Group Replication is a feature in MySQL that allows multiple servers to work together, replicating data across nodes to ensure high availability and fault tolerance. This feature allows a group of MySQL servers to coordinate and agree on changes that occur in the database, ensuring that each server has an identical copy of the data. This setup provides benefits such as automatic failover, in case one server fails, another server in the group can take over to maintain continuity of availability.

The main features of group replication include:

• Transactions are replicated synchronously to maintain data consistency across all members in the group.

• Group replication ensures error detection and recovery, thus ensuring database availability.

• Each server in the group can perform both write and read operations, which enables load balancing and scalability.

• New servers can be added or joined to the group dynamically, as well as member servers can be removed from the group.

Another important thing is how to execute changes in group replication ie. when a read-write transaction is ready to commit, the origin server (on which the change was made) broadcasts the changes and their identifiers to all servers in the group. Such broadcasting of changes is atomic, which means that either all servers allow the changes or reject them (Group Replication Background).

MySQL batch replication and traditional replication differ in several key aspects. Traditional replication in MySQL usually involves a master-slave configuration where one server or master replicates data to one or more slave servers. This replication is asynchronous, which means that there may be a delay in data replication between the master and slave servers. On the other hand, MySQl group replication involves multiple servers working together or synchronously to copy data on all nodes simultaneously.

CONCLUSION

Today, database replication is an indispensable component in database architecture, critical to maintaining high availability, scalability and fault tolerance of the system. MySQL database replication allows data to be distributed across multiple servers, thus improving read performance and overall application efficiency, enabling faster access to data from different locations. Also, database replication offers solutions and strategies for data recovery in the event of a system failure, thereby significantly reducing risk of information loss. Thanks to it, systems are able to respond to the increase in data requirements, optimize load distribution and improve user experience.

REFERENCES

"Group Replication Background" in MySQL Documentation Available: https://dev.mysql.com/doc/refman/8.0/en/group-replication-background.html (Accessed: 1.3.2024.)

"Replication" in MySQL Documentation Available: https://dev.mysql.com/doc/refman/8.0/en/replication.html (Accessed: 27.2.2024.)

Drake, Mark. 2021. "How To Set Up Replication in MySQL" in DigitalOcean. Available: https://www.digitalocean.com/community/tutorials/how-to-set-up-replication-in-mysql (Accessed: 25.2.2024.)

Kemme, B. and Alonso, G., "Database replication", Proceedings of the VLDB Endowment, vol. 3, no. 1-2, p. 5-12, 2010. https://doi.org/10.14778/1920841.1920847

Milani, B. A., & Navimipour, N. J. (2016). A comprehensive review of the data replication techniques in the cloud environments: Major trends and future directions. Journal of Network and Computer Applications, 64, 229-238.

Pohanka, T., & Pechanec, V. (2020). Evaluation of replication mechanisms on selected database systems. ISPRS International Journal of Geo-Information, 9(4), 249.

Šušter, I., & Ranisavljević, T. (2023). Optimization of MySQL database. Journal of process management and new technologies, 11(1-2), 141-151.

Yadav, S., Singh G., & Yadav D., "Mathematical framework for a novel database replication algorithm", International Journal of Modern Education and Computer Science, vol. 5, no. 10, p. 1-10, 2013. https://doi.org/10.5815/ijmecs.2013.09.01

Zmaranda, D. R., Moisi, C. I., Győrödi, C. A., Győrödi, R. Ş., & Bandici, L. (2021). An analysis of the performance and configuration features of MySQL document store and elasticsearch as an alternative backend in a data replication solution. Applied Sciences, 11(24), 11590.

Notes on the authors

Ivan ŠUŠTER, B.Sc., is a M.Sc. student at the Faculty of electronic engineering, University of Niš. E-mail: ivansu995@gmail.com

Darjan KARABAŠEVIĆ, Ph.D. is a Full Professor and Dean of the Faculty of Applied Management, Economics and Finance, University Business Academy in Novi Sad. E-mail: darjan.karabasevic@mef.edu.rs

Aleksandar ŠIJAN, M.Sc., is a Ph.D. candidate at the Faculty of electronic engineering, University of Niš and a Teaching Assistant at the Faculty of Applied Management, Economics and Finance, University Business Academy in Novi Sad. E-mail: aleksandar@mef.edu.rs

Đorđe PUCAR, M.Sc., is a Ph.D. candidate at the Faculty of Economics and Engineering Management, University Business Academy in Novi Sad and a Teaching Assistant at the Faculty of Applied Management, Economics and Finance, University Business Academy in Novi Sad. E-mail: djordje@mef.edu.rs

Luka ILIĆ, M.Sc., is a Ph.D. candidate at the Faculty of electronic engineering, University of Niš and Teaching Assistant at the Faculty of Applied Management, Economics and Finance, University Business Academy in Novi Sad. E-mail: luka.ilic@mef.edu.rs

Goran JOCIĆ, M.Sc., is a Ph.D. candidate at the Faculty of Economics and Engineering Management, University Business Academy in Novi Sad and a Teaching Assistant at the Faculty of Applied Management, Economics and Finance, University Business Academy in Novi Sad. E-mail: goran.jocic@mef.edu.rs