sciendo DOI 10.2478/ntpe-2019-0068

2019
Volume 2
Issue 2
pp. 12-23

Safety Mechanisms in Relational Database as Part of the IT System of the Enterprise

Szymon Berski, Martyna Bilau
Czestochowa University of Technology, **Poland**

Date of submission to the Editor: 09/2019
Date of acceptance by the Editor: 11/2019

INTRODUCTION

Database systems available under open source licenses are an interesting alternative to widely available software for broadly understood as an enterprise resource management for small and medium manufacturing enterprises (Sarkinen, 2007). There is a huge variety of software supporting the management of a production enterprise based on a local or distributed database like ERP, MRP and CRM (Kiełtyka, 2016, Knosala et al., 2013). Their operation is sometimes limited, there is no possibility of self-expansion with minor functionalities, and the costs of service deployments and licenses can be large. Therefore, companies employing young engineers and can use their IT knowledge potential. Many enterprises employ young employees who, graduated engineering courses, have the necessary knowledge to create and develop applications based on relational or object-oriented database systems and knowledgeable in object-oriented programming or Java, SQL, C #, PHP, Python etc. languages.

The use of open software allows making the company management process more flexible also in the area of organization of the production. The specific application in the field of computer aided enterprise activities depends on many factors, e.g. the industry or the profile of the company's activity, the nature and type of production, the form of orders or the level of production monitoring. In many cases for the proper management of the enterprise it is necessary to use the databases software and in such cases the solution can be based on database systems available as free of charge, free software (open source products created e.g. by communities). There are many such software such as Microsoft SQL Server Express, PostgreSQL, Oracle MySQL, MariaDB, Apache Derby and Firebird SQL (Sarkinen, 2007, computerworld.pl, 2019). They can be used in many fields, e.g. in risk management (Berski, 2016, Berski & Wojtyto, 2018), production management (Habel, 2015). Due to the professional application in scientific databases (Hardono et al. 2017, Bautista, 2015), very high popularity (9 out of 10 web applications uses this database) and the speed

of action (computerworld.pl, 2019) in the work database system MySQL (under the GPL license) operating in a Linux environment was used to create a test database of the production company and server protection against data loss caused by hardware failure. Database security in enterprises can be increased at every stage of their processing. At the stage of database server support, data security can be provided through a number of mechanisms, e.g. through high-availability clusters (Chaurasiya et al., 2007, Berski & Szatan, 2018, Moreno et al., 2017) or replication on the level of the database server (Xiang, 2014, Bilau, 2015).

SAFETY MECHANISMS IN MYSQL DBMS

There are a number of security features in the MySQL system. The basic one is the system of permissions, which in MySQL DBMS consists of several levels and allows defining the access to data in the area of the database, table or a specific column for a given user (account). This system is based on the data contained in MySQL database system tables. These tables define the users' rights at various levels of access to the contents of the database. For example, the columns_priv table defines the permissions for columns, db for the database, tables_priv specifies the table permissions, the user table contains a list of all database server users.

The encryption of character strings in MySQL can be done by functions: PASSWORD(), MD5(), SHA1() and ENCRYPT().However, encryption of connection with the MySQL database is based on SSL (Secure Socket Layer).

One of the most important mechanisms supporting maintaining database security is regular backup. The basic tool used to create it is the mysqldump program to create a few selected databases or all databases.

Another mechanism is the mysqlhotcopy tool that allows to back up all database files. The program flushes the query buffer and closes and locks the tables. The script is more efficient than mysqldump because it mainly uses file system operations. To restore the copied database, the server should be turn off and replaced the data files in the MySQL directory with the backup files (Bilau, 2015).

An alternative to using the above programs is manual backup but to automate this procedure, it is recommended to use the CRON service (Cron, 2019) on Unix systems to execute desired tasks (in the background) at desired times.

The above methods register and save the state of the database at the time when the backup was created. If the database is subject to later changes, they will not be included in the copy. To ensure protection of all changes that take place in the database, the backup would have to be done very often, because in the event of a failure all changes that took place after the backup will be lost. Frequent backup is difficult and generating a lot of data, because in the case of large databases it can take up to several hours. In MySQL, it is possible to counteract this problem by operating in a connected binary log mode that records all commands that were performed in the source database and caused changes in its contents. Thanks to this, the data that is in the backup is up to date compared to the source database. To restore the contents of the database,

it should be used a backup copy as well as binary log entries, which will store much less information than the backup itself. These files are called the incremental copies, as they are aimed at gradually supplementing the basic copy. When the files are restored it is possible to return to older database entries, because many such files are created during the binary logging is enabled (MySQL, 2019). For properly recovering data, for example after a main server failure, the tables should be regularly tested. The testing process is performed by using CHECK TABLE table_name command, and if any problems will appear with the table, they can be solved with the REPAIR TABLE command. These commands work for all table types, they can also be used when the server is running and using the database contents. The result of the command checking the database should give the message Table is already up to date on the screen, or show OK in the Msg_text column (MySQL, 2019).

Another method used to check and repair tables will be the mysqlcheck program, also called from the command line, but it will also be used to check InnoDB tables. It is a type of tables that additionally supports transactions, was created separately from the MySQL system and has its own configuration options and a separate method of data storage (Bilau, 2015).

The database replication mechanism has been implemented in MySQL since its early versions and it is constantly being improved. The best effect of server replication can be obtained by using the same versions of RDBMS, because with different versions of the main server and backup servers there may be some errors occurs during use, which may even prevent the operation of replication for some functions of replication process. However, it is not excluded to use such solutions, especially for existing data resources, for example in manufacturing companies that use already existing servers. And such situation was analyzed in the work where the main server was a server based on Linux OS, and a backup server based on Windows OS. Nowadays, almost every enterprise uses databases that can be large, and then the backup becomes the administrator's responsibility. Large amounts of data, especially production data that require multiple processing, caused an over load on servers, and then the backup can act as an unloading mechanism. The main servers from which information is retrieved in the article will be called master and the backup servers which data is replicated will be called slave servers. The master server is responsible for storing data in the system, and backup servers for creating a copy of this data (Fig. 1).

The database created on the slave server will perform the function of "backup" data in the event of a failure of the master server (Pupezescu, 2015). The slave server also relieves the master server when performing calculations and operations on data, because it is on the additional server that these actions can be performed. Additional tests can be performed on the additional server, for example when providing a database for testers who test the resistance of the database to attacks of unwanted users, so between the master servers and slave servers there is isolation. The replication mechanism provides load balancing (Russell, 2010), i.e. spreading the load across multiple servers.

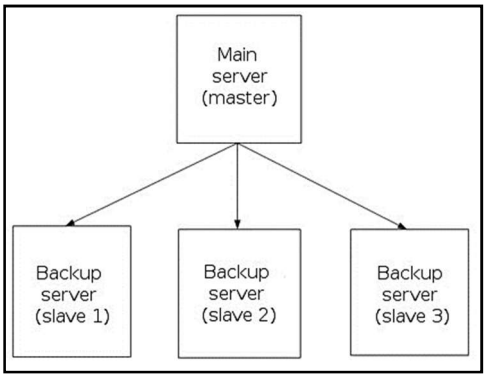

Fig. 1 Schema of servers replication

The main aim of the replication procedure is to get two identical databases, the MySQL database management system uses a very simple mechanism based on binary logs (binaries). The logs contain information about all operations performed by the server from the moment when this option was activated, because it is not enabled by default. So if the replication of the database begins, while it has already been filled with previous information before the transmission to the binaries is activated, it turns out that the replication of the database will not be complete, because the entries operations on data have not been made before. At the moment of starting replication and writing to the binary log, on the master server and on slave servers, the contents of the databases must be identical. If the mechanism works correctly then this information, i.e. binaries, stored in binary files, are read by the slave server and according to them the database with its entire contents on the slave server is copied (Russell, 2010). The next mechanism will be the so-called threads (Fig. 2), which are run on the master server and their task is to send the above-mentioned binaries to the backup servers.

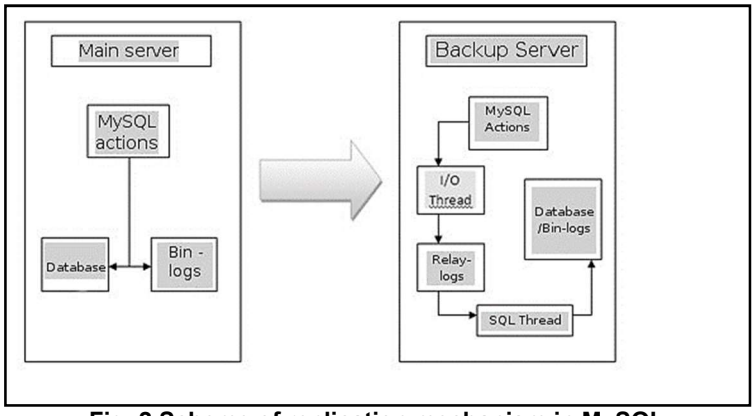

Fig. 2 Scheme of replication mechanism in MySQL

The thread appears on the server side and their launch is preceded by configuring the replication mechanism in the MySQL database management system. The next two threads are created on the side of slave servers. I/O Thread is a file that is responsible for receiving binaries from the master server, then saves these files locally in the temporary relay-log files. The second thread

is SQL Thread, which is responsible for parsing relay-logs and sending queries to the database (Bilau, 2015).
For the server version for the tests MySQL distinguishes three methods of server replication (RBR- row based replication, SBR- statement based replication, MFL-mixed format logging). They differ from each other in the data format in which data is written to the binaries. In SBR method queries that the server performed are saved to the file and also is very fast and was chosen to replicate the database for the purposes of this paper. More advantages and explanations about these methods were presented in literature (Bilau, 2015, MySQL, 2019). Replication has become a very popular mechanism that gives a wide range of opportunities (Ping, 2014) used, among others, by the most popular internet portals in the world. Most replication problems are provided at the beginning of its startup, as well as the diversity of its configuration on various systems and errors when using older versions of MySQL. The mechanism of data replication is not a solution without shortcomings, however its reasonable implementation in order to support enterprise database resources can protect the system against excessive load on the main database server, but also the unpleasant consequences of hardware or main server failure (Bilau, 2015).

AIM AND RESEARCH METHODOLOGY

The purpose of the work was creation a backup copy of the operating database of the selected IT system in a continuous mode, i.e. a copy of data relevant to the selected company will be constantly updated during operations in term of long lime functioning. The work will analyze the typical operation of the data in the conditions of main server failure. The scope of the tests included the following stages:
1. Installation of the MySQL RDMBS environment on a Linux system.
2. Creating a test database that modeling the chosen enterprise's resource flow.
3. Starting database replication on a different operating system, e.g. Windows.
4. Checking the mechanisms of replication in conditions of a controlled failure of the main server.
During the designing process of database model of a tool-producing enterprise X, the flow of elements in the company producing fine metal details was identified, There made some collections which will be represented by tables (relational model of the database). These are, among others, tables: employees alias pracownicy, orders (zamowienia), position_orders (pozycja_zamowienia), customers (kienci), suppliers (dostawcy), parts (czesci), parts list (wykaz_czesci), products (produkty). An intensional part was built, i.e. the structure of the database with attributes for particular tables, and they were combined with inter-table relations. The structure and a set of integrity constraints (primary, foreign keys etc.) are shown in Figure 3.
This model which enables enterprise data management together with graphic interfaces of the web servers, (also in mobile devices) is part of an IT system supporting the design, manufacture and sale of the company's X products.

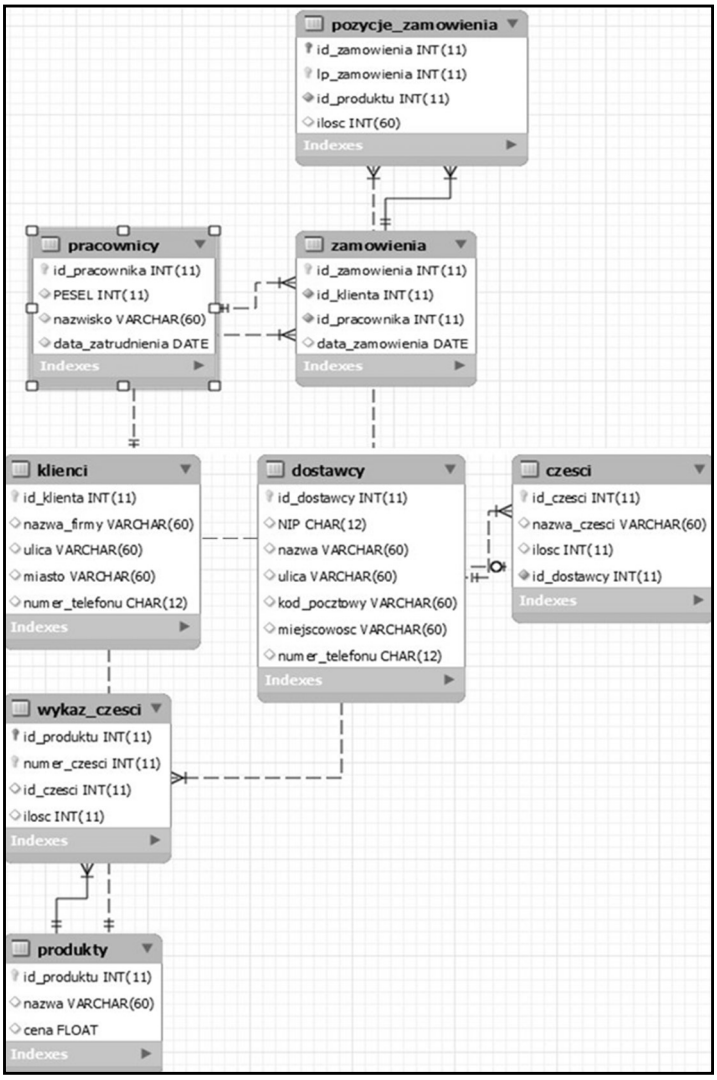

Fig. 3 Simplified model of the production enterprise X

Due to the reliability and speed of processing, the database schema was normalized to 3-rd NF (Codd, 1978), and the entire database was replicated on a separate server. The enterprise X database was created in MySQL version 5.6, on a server based on the Linux operating system, and its copy on a backup server based on the Windows Server 2008 operating system. The replication process was based on the replication mechanisms built into the MySQL system.

REPLICATION OF THE DATABASE

The replication process is an essential tool to increase the security and reliability of the database server. In Figure 4 the structure of the replication of the enterprise X database has been shown.

The SBR method of the replication process was chosen. The SBR method is the fastest and most efficient replication method (Russell, 2010) and such features should be characterized by a solution for replication the enterprise X database. The database will not use more complicated queries, for example using random functions that require the use of other replication methods that are slower than the SBR method, which is why SBR is the method used for replication in this work.

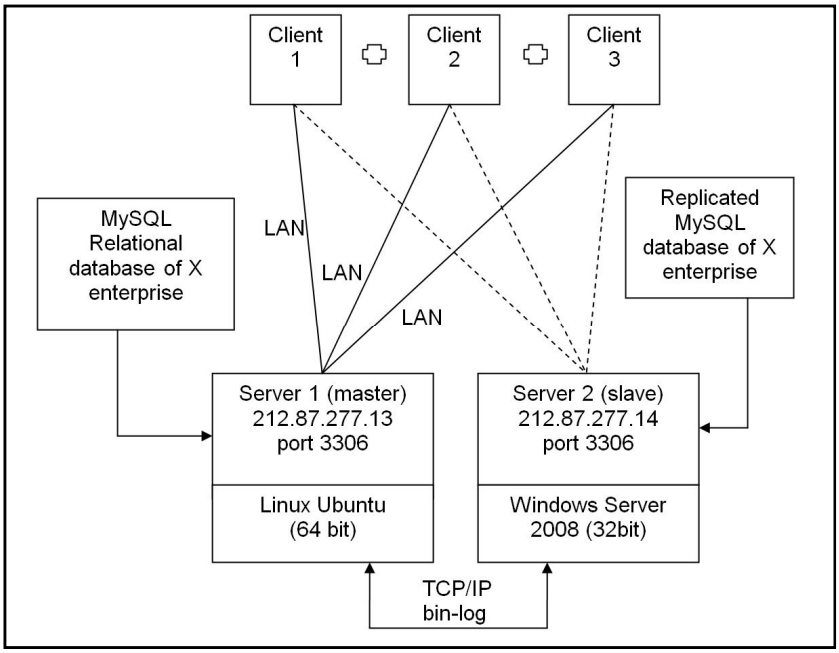

Fig. 4 The enterprise database replication scheme in the relational data structure

The main parameters of the servers configuration are presented in Table 1.

Table 1 Essential parameters for the configuration files of the servers

Master server	Slave server
bind-address = 0.0.0.0	log-bin = slave-bin.log
collation-server=utf8_polish_ci	collation-server=utf8_polish_ci
server-id = 1	server-id=2
max_binlog_size = 500M	sync_binlog = 1
sync_binlog = 1	replicate-do-db = bp
replicate-do-db = bp	
innodb_flush_log_at_trx_commit = 1	

Source: (Bilau, 2015)

After the restarting servers and the replication process should be activated and the slave server status parameters: *Slave_IO_Running and Slave_SQL_Running should indicate YES and The Slave_IO_State* should be switched as a Waiting for master to send event.

VERIFICATION OF THE REPLICATION PROCESS

By replication, each modification on the primary server causes the same change on the backup server. To perform the verification process some changes have been made in data presented in Figure 5.

```
mysql> desc czesci;
+--------------+-------------+------+-----+---------+----------------+
| Field        | Type        | Null | Key | Default | Extra          |
+--------------+-------------+------+-----+---------+----------------+
| id_czesci    | int(11)     | NO   | PRI | NULL    | auto_increment |
| nazwa_czesci | varchar(60) | YES  |     | NULL    |                |
| ilosc        | int(11)     | YES  |     | NULL    |                |
| id_dostawcy  | int(11)     | NO   | MUL | NULL    |                |
+--------------+-------------+------+-----+---------+----------------+
4 rows in set (0.06 sec)

mysql> select * from czesci;
+------------+---------------+-------+-------------+
| id_czesci  | nazwa_czesci  | ilosc | id_dostawcy |
+------------+---------------+-------+-------------+
|          1 | frez_do_dibondu |    27 |           2 |
|          2 | frez_do_plexi   |    48 |           2 |
|          3 | frez_down_cut   |    23 |           1 |
+------------+---------------+-------+-------------+
```

Fig. 5 Checking the structure and content of the *parts* table

For the designed database it was chosen a table containing tools for machines. After completing the magazine (the extensional part of the model), a new tool will appear in this table (*id = 4* name of the tool: *frezMP*).
In order to check the replication reaction, an INSERT query adding the new record to the selected table was executed to database on aster server (Fig. 6).

```
mysql> insert into czesci values (4, 'frez_MP', 12, 3);
Query OK, 1 row affected (0.13 sec)
```

Fig. 6 Adding a new record in the 'parts' table

Next, in the slave server the contents of the table were checked if the data was the same as on the main server. The record added on the main server has led to an identical change on the backup server.
The replication system in the data layer (extensional) works correctly. In the next verification stage, replication of the base in the structure layer (intensional) was checked by changing the data type in the table (Fig. 7). This modification consists in changing the BIGINT field to a string field, i.e. CHAR (11). It is a text data type that is used to store a string of characters, in this case consisting of a maximum of 11 characters.

```
mysql> select * from czesci;
+------------+-----------------+--------+--------------+
| id_czesci  | nazwa_czesci    | ilosc  | id_dostawcy  |
+------------+-----------------+--------+--------------+
|          1 | frez_do_dibondu |     27 |            2 |
|          2 | frez_do_plexi   |     48 |            2 |
|          3 | frez_down_cut   |     23 |            1 |
|          4 | frez_MP         |     12 |            3 |
+------------+-----------------+--------+--------------+
4 rows in set (0.05 sec)
```

Fig. 7 Checking the operation of the replication mechanism

The structure initially looked like in Figure 8. After logging in to the database, it was checked that the data type in the PESEL table is BIGINT(20), which should be changed in the structure layer, because it takes up a lot of memory space and is not the optimal data type for storing such information.

```
Server version:         5.5.44-0ubuntu0.14.04.1-log (Ubuntu)

mysql> use bp;
Database changed
mysql> desc pracownicy;
+------------------+-------------+------+-----+---------+----------------+
| Field            | Type        | Null | Key | Default | Extra          |
+------------------+-------------+------+-----+---------+----------------+
| id_pracownika    | int(11)     | NO   | PRI | NULL    | auto_increment |
| PESEL            | bigint(20)  | NO   |     | NULL    |                |
| nazwisko         | varchar(60) | NO   |     | NULL    |                |
| data_zatrudnienia | date       | NO   |     | NULL    |                |
+------------------+-------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

Fig. 8 The structure of the database before the change

The structure change consisted in changing the data type in the PESEL table with bigint (20), and char (11). This type of text occupies as much space in the memory as is necessary in a given case, so it will take up less space in memory and the base structure will be optimized. The command:

```
alter table pracownicy modify PESEL char(11) not null;
```

was used to change the data type (Fig. 9).

```
mysql> desc pracownicy;
+-------------------+-------------+------+-----+---------+----------------+
| Field             | Type        | Null | Key | Default | Extra          |
+-------------------+-------------+------+-----+---------+----------------+
| id_pracownika     | int(11)     | NO   | PRI | NULL    | auto_increment |
| PESEL             | char(11)    | NO   |     | NULL    |                |
| nazwisko          | varchar(60) | NO   |     | NULL    |                |
| data_zatrudnienia | date        | NO   |     | NULL    |                |
+-------------------+-------------+------+-----+---------+----------------+
4 rows in set (0.00 sec)
```

Fig. 9 Changing the type of data in the database

After making changes to the structure of the "employees" table, the slave server 212.87.227.14 was logged in and the replication status of the base was checked (Fig. 10):

```
mysql> \s
--------------
C:\Program Files\MySQL\MySQL Server 5.6\bin\mysql.exe  Ver 14.14 Distrib 5.6.25,
 for win32 (x86)

mysql> use bp;
Database changed
mysql> desc pracownicy;
+-------------------+-------------+------+-----+---------+----------------+
| Field             | Type        | Null | Key | Default | Extra          |
+-------------------+-------------+------+-----+---------+----------------+
| id_pracownika     | int(11)     | NO   | PRI | NULL    | auto_increment |
| PESEL             | char(11)    | NO   |     | NULL    |                |
| nazwisko          | varchar(60) | NO   |     | NULL    |                |
| data_zatrudnienia | date        | NO   |     | NULL    |                |
+-------------------+-------------+------+-----+---------+----------------+
4 rows in set (0.02 sec)
```

Fig. 10 State of enterprise X replication on the backup server

The replication system in the structure layer also works correctly. Changes to the data type that were made on the main server are also performed correctly on the backup server.

The replication system presented in the paper worked over 6 years on such configured servers. During this time there were several failures related to the lack of power supply to the master server and slave server. After the failure of the slave server, the database copies (*mysqldump*) created in automatic mode and the update of the database system configuration files were used. After the failure of the slave server the relay-logs are incorrect and the replication mechanism is imposable (Fig. 11).

```
mysql> start slave;
Query OK, 0 rows affected (0.02 sec)

mysql> show slave status\G
*************************** 1. row ***************************
               Slave_IO_State:
                  Master_Host: 212.87.227.13
                  Master_User: repuser
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mysql-bin.001545
          Read_Master_Log_Pos: 107
               Relay_Log_File: szymekwS-relay-bin.003128
                Relay_Log_Pos: 4
        Relay_Master_Log_File: mysql-bin.001545
             Slave_IO_Running: No
            Slave_SQL_Running: Yes
              Replicate_Do_DB: bp
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
```

Fig. 11 Slave server status after failure

The I/O thread is off and slave server could not receive the statement. The following procedure (manual or automatic depends on software realization) should be done:

- lock database on master server,
- backup replicated database or getting the previously copied database,

- reading the *master log_file* and *master log position* in analyzed event parameters were: *mysql-bin.001618* and position *107*,
- stop slave action,
- load the backup of the database on slave,
- indicate correct log file and position in this file,
- unlock tables on master;
- start slave thread on slave,
- the I/O thread should be on and status of the server should have value *waiting for master to send event* (Fig. 12).

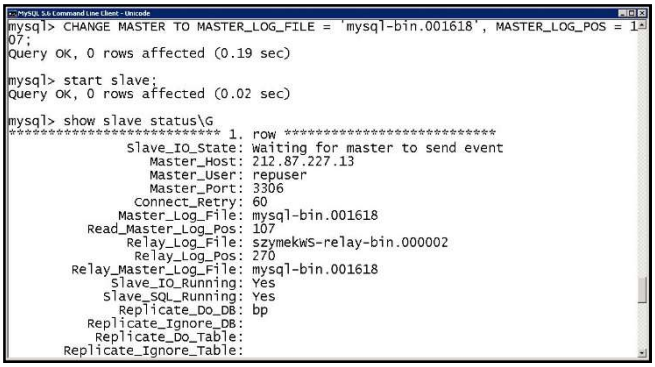

Fig. 12 The I/O thread is on and slave server is waiting for the statement

CONCLUSIONS

In the work the replication process of the enterprise X database using the MySQL relational database management software was done. The use of this DBMS enable the replication of the database, i.e. the creation of its identical copy on the second machine with other software, which allows to avoid data loss, in the conditions of a system failure.

After performing the replication and analyzing its operation, it can be concluded that:

- the database replicated on the backup server is identical to the database on the main server, both in the data layer and in the structure layer (different type of queries), so in the case of physical damage to the main server, the user is guaranteed data security, which is recovered entirely,
- the replication method (SBR) used is a fast and efficient when using simple SQL queries used in the created enterprise model,
- thanks to the replication mechanism in MySQL system with some essential security mechanisms (firewalls, ssl) the enterprise avoided data loss and long-term failure of the it system through the 6 analyzed years.

MySQL is a free software under the General Public License, efficiently operating on both Windows and Linux, that is currently the most popular operating systems. The MySQL program is constantly being improved, its newer versions appear in very short intervals, which proves the high potential of this database management system, which is applicable in an increasing number of enterprises.

After configuring servers and initiating replication, one can conclude that the entire process has been performed correctly by mapping the database that is located on the main server exactly on the backup server. All changes that were made on the main server during replication operations performed correctly on the second server. Thanks to the tools and mechanisms embedded in the MySQL system, data can be managed fully efficiently and safely.

ACKNOWLEDGEMENTS

The research was carried out as part of statutory research No. BS/PB-201-302/2018 financed by: Ministry of Science and Higher Education.

REFERENCES

Bautista, E. and Serna, N. L. (2015). An MDE-based graphical tool for the validation of MySQL replication models, Latin American Computing Conference (CLEI), Arequipa, pp. 1-12. doi: 10.1109/CLEI.2015.7360009.

Berski, S. (2016). Aplication of spatial data bases in crisis management. Seria Monografie nr 62 . Częstochowa, pp. 110-124.

Berski, S., Szatan, R. (2016). Zastosowanie klastra wysokiej dostępności w bezpieczeństwie relacyjnych baz danych, PRACE NAUKOWE Akademii im. Jana Długosza w Częstochowie, Technika, Informatyka, Inżynieria Bezpieczeństwa, Częstochowa, 2016, t. IV, pp. 49-57, DOI 10.16926/tiib.2016.04.04.

Berski, S., Wojtyto, D. (2018). Zastosowanie wolnego oprogramowania do administrowania danymi przestrzennymi w zarządzaniu kryzysowym. PRACE NAUKOWE Akademii im. Jana Długosza w Częstochowie, Technika, Informatyka, Inżynieria Bezpieczeństwa, Częstochowa, vol. VI, pp. 443-453, DOI 10.16926/tiib.2018.06.48.

Bilau, M. (2015). Safe Management of Data in MySQL Databases Engineering. Thesis under the supervision of Szymon Berski, Technical University of Częstochowa, unpublished material.

Chaurasiya, V. Dhyani P. and Munot, S. (2007). Linux Highly Available (HA) Fault-Tolerant Servers. 10th Int. Conf. on Information Technology Orissa, pp. 223-226. doi: 10.1109/ICIT.2007.58.

Codd, E. F. (1972). Further Normalization of the Data Base Relational Model. IBM Research Report RJ909 (August 31, 1971). Republished in Randall J. Rustin (ed.), Data Base Systems: Courant Computer Science Symposia Series 6. Prentice-Hall.

computerworld.pl, (2019). 6 darmowych baz danych z komercyjnymi możliwościami. https://www.computerworld.pl/news/6-darmowych-baz-danych-z-komercyjnymi-mozliwosciami,387854.html [Accessed 4 Jun. 2019].

CronHowto (2019). https://help.ubuntu.com/community/CronHowto [Accessed 04 Jun 2019].

Habel, J. (2015). Zastosowanie baz danych w analizie danych historycznych sprzedaży na potrzeby klasyfikacji pozycji materiałowych i sterowania zapasami. Enterprise Management, Number 4 ,pp. 6-14.

Hardono, Surjandari, I., Rachman, A., Panjaitan, Y. A. B. and Rosyidah, A. (2017). Development of theses categorization system search engine using PHP and MySQL. Int. Conf. on IT Systems and Innovation, Bandung, pp. 194-199.doi: 10.1109/ICITSI.2017.8267942.

Kiełtyka, L. (2016). Wybrane systemy wspomagania zarządzania w przedsiębiorstwach o charakterze usługowym. Wyd. Glinkowska B., Wydawnictwo Uniwersytetu Łódzkiego, Łódź, pp. 107-123, doi: 10.18778/8088-492-2.10.

Knosala, R. et al, (2013). Komputerowe wspomaganie zarządzania przedsiębiorstwem. Ed. PWE, WARSZAWA.

Moreno-Vozmediano, R. Montero, R. S. Huedo, E. Llorente I. M. (2017). Orchestrating the Deployment of High Availability Services on Multi-zone and Multi-cloud Scenarios, Journal of Grid Computing [online], vol.16 Issue I, pp. 39-53. Available at: https://link.springer.com/article/10.1007%2Fs10723-017-9417-z [Accessed 22 Jun. 2019].

MySQL 8.0 Reference Manual (2019) ttps://dev.mysql.com/doc/refman/8.0/en/binary-log.html [Accessed 04 Jun 2019].

Ping, Y., Hong-Wei, H. and Nan, Z. (2014). Design and implementation of a MySQL database backup and recovery system. Proceeding of the 11th World Congress on Intelligent Control and Automation, Shenyang, pp. 5410-5415 doi: 10.1109/WCICA.2014.7053638.

Pupezescu, V. and Rădescu, R. (2015) Enhanced protection level by database replication in the easy-learning online platform, 9th Int. Symposium on Advanced Topics in Electrical Engineering, Bucharest, pp. 929-932. doi: 10.1109/ATEE.2015.7133935.

Russell, J.T. Dyer. (2010). MySQL Replication: An Administrator's Guide to Replication in MySQL. A Silent Killdeer Publishing.

Sarkinen, J. (2007). An open source(d) controller. INTELEC 07 - 29th Int. Telecom. Energy Conf., Rome, pp. 761-768. doi: 10.1109/INTLEC.2007.4448885.

Welling, L., Thomson, L. (2004). MySQL Tutorial, MySQL Press.

Xiang, K. (2014). An improvement in MySQL cluster in e-commerce scenarios, Proc. of 2nd Int. Conf. on IT and Electronic Commerce, Dalian, pp. 286-289 doi: 10.1109/ICITEC.2014.7105620.

Abstract.
The work examines the effectiveness of the replication mechanism of the X production company database. In order to extend the functionality of the computer database of the enterprise, a model of its resource flow was created and an intensional and extensional part was created for a test database based on real enterprise resources. The model has been simplified to 3-rd normal form. The implementation was done in the MySQL database system. Two different operating systems were selected for testing: Windows and Linux. The database management system (DBMS) is working under the GPL license. MySQL DBMS offers many security mechanisms, and to secure the database, system of users permissions to objects have been selected and also an encryption of access passwords for users and connections to servers was used. A replication engine based on the binary log and the model "main server and backup server" was used to create a copy of the enterprise data.

Keywords: enterprise database, MySQL, data replication, data security