Relative Performance of Various Types of Repositories for MySQL Archive Backup and Restore Operations

https://doi.org/10.3991/ijoe.v18i13.33429

Pavel Petrov[1(✉)], Ivan Kuyumdzhiev[1], Georgi Dimitrov[2], Adelina Kremenska[3]
[1]University of Economics – Varna, Varna, Bulgaria
[2]University of Library Studies and Information Technologies, Sofia, Bulgaria
[3]Institute of Robotics, Bulgarian Academy of Sciences, Sofia, Bulgaria
petrov@ue-varna.bg

Abstract—The backup and restore operations are essential part of database system's administration process. Usually, the size of the database is a major factor that determines numerous practical issues, among which is what backup and recovery methods to be used. However, in addition, there are various types of storage systems (repositories) available in MySQL 8.0, for example MyISAM, Merge, Memory, InnoDB, CSV, Blackhole, Archive and Federated, which in general sense determine how to store each table into at least one file. Each of these storage systems have their own positive and negative characteristics, so when constructing logical archives, it is vital to compare the performance of those various types of repositories.

Keywords—database backup, database restore, storage engine, archiving, MySQL

1 Introduction

One of the most widely used database systems on the Internet is MySQL, with thousands of sites using it for a variety of purposes. It's also used in some Big Data projects or in handling large data volumes. Companies such as Facebook, Twitter, Symantec, Verizon, NASA, YouTube, Wikipedia, and many others can be added to the list of clients [1].

The variety of clients and the widespread use of MySQL imply that there are adequate reliable and efficient capabilities for backing up and restoring databases. The main characteristics of MySQL related to physical and logical data management will be considered sequentially in order to study through the lens of the goals set for the current development and to derive unresolved issues from the theory.

The size of the database is a factor that influences both the backup and recovery processes. The specific characteristics of MySQL that affect the size of the database, the methods for measuring the size, and obtaining information that allows it to grow while maintaining growth trends should all be considered in order to assess its significance.

The MySQL architecture allows for the selection of different repositories for each of the database's tables, resulting in three levels:

1) The connection layer is in charge of managing the server connections.
2) SQL Engine—analyzes and optimizes server queries.
3) Storage Engine—read and write data from and to storage.

According to Bell [2], one of the best features of MySQL is the use of different repositories, which allows to customize the database by selecting the most appropriate repository based on the requirements of the applications to each table. For example, for very active databases, using a repository that provides transaction control, or using a repository that stores data in RAM in cases where it is accessed frequently but does not change.

The characteristics of the most commonly used data warehouses will be derived in order to highlight the relationship between backup and recovery processes and the data warehouses used.

MyISAM, Merge, Memory, InnoDB, CSV (delimited text files), Blackhole, Archive, Example (template for creating new repositories), and Federated [3] are data warehouses that present each table of a hard drive as at least one file in MySQL 8.0. It has a. frm extension and is created by the server. Different repositories can create additional files containing table data and index information, and their names and structures depend on the repository [4].

Up until MySQL 5.5, MyISAM was the default data store type. Tables are created by compressing and optimizing indexing methods to improve speed and are thus used in the implementation of data warehouses and e-commerce applications [2]. The locking is done at the table level, and this storage is recommended when read speed is critical. Each table in MyISAM is represented on the hard disk by three files: the file describing the structure of the table, files with the extension .myd (for writing row contents), and files with the extension .myi (saving index information) [5].

Outstanding transactions can be rewilded in InnoDB, and tables of this type can grow to 64 TB (less than MyISAM – 256TB). The repository is used by default for all newly created tables, despite the smaller maximum size, and the developer (Oracle) recommends it. InnoDB is the first MySQL repository to support both transaction management and data reference integrity (the so-called ACID compliance—atomicity, consistency, isolation, durability). Tables in this format support record-level locking (only paid NDB storage supports this feature): while a transaction is being performed on certain records in the table, those records remain locked, but other transactions can be performed on other records. We accept therefore, according to some studies [6], that the InnoDB tables match (and sometimes exceed) the performance of Oracle tables.

InnoDB tables store data in two ways: privately and in shared table space. Shared is made up of one or more large files that form a logically connected storage area with a size equal to the sum of the individual files' sizes [7]. The only file that is specific to each table in this case is the .frm file. Using the default settings in MySQL (up to version 5.6), a single data file named ibdata1 (also known as a shared table space) and two log files named ib logfile0 and ib logfile1 are created. These files contain information about the data in the tables of all databases on the server, as well as the operations

that were performed on them. The InnoDB file per table option was added to the default option [8] after version 5.6, which creates a data file with the .ibd extension for each table. Given the differences between physical and logical data archiving, the question arises as to which method would produce better archiving and recovery results: reading data from a single file that contains data from other databases or reading data from multiple files without redundant content.

2 Methodology for estimating backup and restoration capabilities on various types of repositories

It is critical to compare the performance of one type of repository to that of other possible types of repositories when creating logical archives. For instance, as an advantage of InnoDB, the official documentation [9] of MySQL states that it supports its own buffer, which caches frequently available data and indexes in RAM, which speeds up processing. The repository, according to the same documentation, is designed for processing efficiency when dealing with large amounts of data (with specialized database servers up to 80% of RAM is allocated to the InnoDB cache).

The repository's structure is efficient for long text and BLOB fields, and it allows for the creation and deletion of indexes with less impact on performance and accessibility. This, in our opinion, is one of the reasons why it is recommended for environments with a lot of write operations, whereas MyISAM is recommended for tables that are mostly read-only.

We can define the following by analyzing the characteristics of the two most widely used repositories:

1) We could determine which of the two methods of reading data from one file, which contains data from other databases from data, or reading data from multiple files, but without redundant content, would lead to better results in archiving and recovery—i.e. to detect the impact of the innodb file per table option on the backup and recovery process.
2) To determine how a faster read speed of MyISAM and a faster write speed of InnoDB affects the backup and recovery speed in the following way: faster backup MyISAM tables and faster recovery InnoDB tables. If the derived dependency is correct, it could be defined how this affects total backup and recovery time (whether the "write" advantage has a greater impact than the "read" advantage).

The Merge and Federated repositories serve the same function. While the first is a virtual table created by combining several tables with the same MyISAM structure from the same server [6], Federated can be based on tables from multiple servers [2].

Because the data and indexes are stored in RAM, memory tables only use a .frm file and are not otherwise represented in the file system [10]. The content is deleted when the server is stopped, and the tables are empty when the server is restarted. Blackhole tables, like Memory tables, are presented on the hard disk with a single .frm file, but they do not contain any data [11].

The Archive repository is designed to store large amounts of compressed historical data that is only occasionally available. There is no indexing, so the only way to get to the information is to scan the table [12].

As previously stated, one of the primary benefits of MySQL is the ability to select repositories for tables with different purposes within the same database. Although this gives developers more freedom and can lead to increased productivity, it is important to consider how the storage choice affects the backup and recovery strategy.

Merge or Federated tables are virtual and do not store the merged tables' actual data. As a result, using this type of repository to archive tables can only be expressed in terms of preserving their structure. The same is true for Blackhole tables; because they do not store data, archiving is limited to the contents of the .frm file.

We can infer alleged archiving features based on the main characteristics of Memory and Archive repositories. Memory tables keep data in RAM, which means that when creating an archive, it will be read directly from there and written to media, resulting in better performance than the previous version, which reads data from the hard drive first. The data will need to be read by the storage medium when recovering from the archive, but instead of being saved to the hard disk, it will be transferred to RAM. This leads to deem that the best time to create and restore archives is when memory tables are used.

It can be assumed that tables of the Archive type are at the other end of the spectrum—due to the large amount of non-indexed data, reading their contents when creating a logical archive will take longer than any other repository, slowing down the process of creating the archive. Recovery, on the other hand, would not necessitate the creation of indexes, reducing the time required to return the table to an active state. This leads to the conclusion that Archive tables take the longest time to create a logical archive and recover from it than other repositories that store files on hard disk.

3 Experimental results and discussion

The ability to recover data at a specific point in time was previously identified as a characteristic of logical archives. To implement this functionality, a log of operations must be kept, with a record of when they occurred and, if necessary, when they were applied in a specific time interval. MyISAM, Memory, InnoDB, Archive, and NDB are some of the MySQL repositories that support backup and recovery (storage available only for the paid version of the product). Based on the foregoing facts and analysis, we can conclude that the following statements should be considered when developing a strategy for archiving and restoring a database using the free version of MySQL (see Table 1):

1) Although MySQL offers a variety of data storage options, the unique characteristics of each repository should be considered when developing a backup and recovery strategy.
2) MyISAM tables provide faster access to read data due to efficient indexing methods, which should result in faster archiving when compared to InnoDB.
3) When compared to MyISAM, InnoDB tables have better data recording performance, which means better results when recovering from a backup copy.

4) When compared to other repositories, memory tables should have the shortest time to create and recover from a logical archive file.
5) Due to the lack of indexing, tables of the Archive type should have the lowest rate of archiving and relatively higher recovery from InnoDB and MyISAM.

Table 1. Relative backup and recovery speed for different types of MySQL repositories

Repository Type	Backup Speed	Recovery Speed
MyISAM	Higher Than InnoDB	Lower Than InnoDB
InnoDB	Lower Than MyISAM	Higher Than MyISAM
Memory	Highest	Highest
Archive	Lowest	Better Than MyISAM and InnoDB

Remark: The content in this table is based on our experience of archiving data used in educational institutions.

As previously stated, in order to achieve recovery at some point in the future, a list of ongoing operations should be kept. All commands that are executed on the database after MySQL version 3.23.14 are saved in an additional one or more files. Because this file is in binary format, it is referred to as a binary log [6]. To view it, use the mysqlbinlog tool, which converts it to a text file. Creating and maintaining logs adds to the load on the system and necessitates more hard disk space. Furthermore, because logs contain sensitive information, they can be misused to compromise security; as a result, some administrators [13] prefer to turn them off, limiting backup options.

Depending on the server configuration, the binary log may contain different information [14]:

1) Saves the SQL queries that have been executed;
2) Saves the values of the tables' changed rows;
3) Mixed mode, which alternates between the two depending on the situation.

One of the main questions here is whether and how the binary log settings affects the backup and recovery processes. We consider the possibility that if the values of changed rows are copied from time to time during archiving and recovery rather than being converted to SQL commands to be executed, the second method of binary log management will result in better performance. This consideration is based on the assertion that data copying is a process that requires less checking and work on MySQL's part than running a SQL query. The only way to verify the truth of the statement is to run tests and compare the results, as there is no answer to this question in the official MySQL documentation.

Each of the considered DBMS should take into account the method of measuring the size of the database and possible methods for simulating its growth based on trends in changing its tables when developing an algorithm for testing the effectiveness of the different types of archiving.

Bradford [15] recommends summing the sizes of tables and indexes in the database to calculate the actual size of the database, which is similar to the identical problem in MS SQL Server. This data can be accessed through the Tables table in Information

Schema [16], and only storage tables that are subject to archiving should be specified in the request condition. According to the same author, calculating the size of the database is critical for archiving and recovery because the size of the resulting logical archive is roughly equal to the size of the data with a 10–15% difference.

We propose the following logic in order to follow the trends in the development of the database and set in an approach for its effective management:

1) Creation of specific user procedures to record table operations in the event that short-term monitoring cannot provide reliable data.
2) Access to the binary log and analysis of the available data on command frequency. A simple text editor can also be used to edit MySQL archives, allowing to calculate the number of operations performed for each table over a given period. Considering this, we propose that the two approaches listed above be supplemented by the following:
3) Browsing through existing archive files.

We conclude that the development of specific user procedures can be expressed through the development of triggers or the installation of MySQL plugins. They have the ability to add their own variables and commands, tables in Information Schema, run in the background, and so on [17]. Audit log is an example of such an add-on; it receives events during request execution to record what is happening on the server. McAfee created Audit log, which focuses on security and database audit requirements. It is free to use and distributed under the terms of the GNU General Public License [18]. As a result, it's used in a variety of applications, including Percona [19] and the commercial version of MySQL – MySQL Enterprise Audit [20], which use this add-capabilities to track server events.

The mysqlbinlog tool can be used to access the operations stored in the binary log. It allows to convert a binary file into text that contains SQL commands and their timestamps. Thus, from an algorithmic standpoint, the second and third methods for database load analysis produce nearly the same expression—reading a text file and outputting the number of executed operations to a given object per unit time. We suppose that in some cases it is feasible to exclude the possibility of using add-ons and/or functionalities offered by the paid version or other software in order to achieve greater portability [21] of the developed application for evaluating the effectiveness of archiving strategies. Once the methods for measuring database size, issues related to archiving tables from different repositories, and tools for studying table growth rates have been specified, the approaches that will be used by the developed system for measuring archiving speed should be investigated. as well as recovery.

4 Conclusion

Although predicting how long it will take to create the archive is one of the most important questions in the archiving and recovery strategy, there is no way to give an accurate answer. The complexity stems from the fact that the size of the database, the amount of RAM, the data storage used, the MySQL configuration, the hard disk speed,

and the system load all have an impact on the execution time. As a result, the only way to predict backup and recovery speed is to run tests and measure it at various values of these variables. To reduce the impact of the testing application, it is recommended that no additional tools be used, and thus we propose out the possibility of using additives to measure the time to perform the monitored operations. Using the time command as a prefix to the backup command in MySQL on Unix-based operating systems will provide information about the exact time required to create the archive**.**

5 Acknowledgment

This research is financially supported by the Project "Innovations for Big Data in a Real World" iBigWorld – ERASMUS + Programe, KEY ACTION 203:STRATEGIC PARTNERSHIP FOR HIGHER EDUCATION PROJECT; 2020-1-PL01-KA203-082197 from the University of Library Studies and Information Technologies, Sofia, Bulgaria.

6 References

[1] MySQL. (2022). MySQL Customers. Retrieved March 2, 2022 from https://www.mysql.com/customers/
[2] Bell, C. (2013). Expert MySQL. Apress. https://doi.org/10.1007/978-1-4302-4660-2
[3] MySQL. (2022). MySQL 8.0 Reference Manual / Alternative Storage Engines. Retrieved March 2, 2022 from https://dev.mysql.com/doc/refman/8.0/en/storage-engines.html
[4] DuBois, P. (2009). MySQL. Fourth Edition. Addison-Wesley.
[5] Fadelelmoula, A. A. (2021). Exploiting Cloud Computing and Web Services to Achieve Data Consistency, Availability, and Partition Tolerance in the Large-Scale Pervasive Systems. International Journal of Interactive Mobile Technologies (iJIM), 15(15), pp. 74–102. https://doi.org/10.3991/ijim.v15i15.22517
[6] Vaswani, V. (2009) MySQL Database Usage & Administration. McGraw Hill Professional.
[7] DuBois, P. (2014). MySQL Cookbook: Solutions for Database Developers and Administrators. O'Reilly Media, Inc.
[8] Aqel, M. J., Naqshbandi, O., Sokiyna, M. Y., & Valentyn, P. (2020). Messaging System Design Based on Using Servers and Encoding System. International Journal of Interactive Mobile Technologies (iJIM), 14(10), pp. 107–127. https://doi.org/10.3991/ijim.v14i10.15189
[9] MySQL. (2022). Benefits of Using InnoDB Tables. Retrieved March 2, 2022 from http://dev.mysql.com/doc/refman/8.0/en/innodb-benefits.html
[10] Kuyumdzhiev, I., & Nacheva, R. (2019). Correlation Between Storage Device and Backup and Restore Efficiency in MS SQL Server. Serdica Journal of Computing, 13(3–4), pp. 139–154. https://doi.org/10.55630/sjc.2019.13.139-154
[11] Kuyumdzhiev, I. (2019). Comparing Backup and Restore efficiency in MySQL, MS SQL Server and MongoDB. International Multidisciplinary Scientific GeoConference: SGEM, 19(2.1), pp. 167–174. https://doi.org/10.5593/sgem2019/2.1/S07.022
[12] MySQL. (2022). MySQL 8.0 Reference Manual / The ARCHIVE Storage Engine. Retrieved March 2, 2022 from https://dev.mysql.com/doc/refman/8.0/en/archive-storage-engine.html
[13] Murach, J. (2012). Murach's MySQL. Mike Murach & Associates, Incorporated.
[14] MySQL. (2022). MySQL 8.0 Reference Manual / Binary Logging Formats. Retrieved March 2, 2022 from http://dev.mysql.com/doc/refman/8.0/en/binary-log-formats.html

[15] Bradford, R. (2012). Effective MySQL Backup and Recovery. McGraw Hill Professional.
[16] Asrial, A., Syahrial, S., Maison, M., Muhaimin, M., & Kurniawan, D. A. (2020). E-Assessment for Characters Independence. International Journal of Interactive Mobile Technologies (iJIM), 14(15), pp. 125–141. https://doi.org/10.3991/ijim.v14i15.12995
[17] Schwartz, B., Zaitsev, P., Tkachenko, V. (2012). High Performance MySQL: Optimization, Backups, and Replication. O'Reilly Media, Inc.
[18] GitHub McAfee Engineering. (2021). mysql-audit License. Retrieved March 2, 2022 from https://github.com/mcafee/mysql-audit/wiki/License
[19] Percona. (2022). Audit Log Plugin. Retrieved March 2, 2022 from https://www.percona.com/doc/percona-server/8.0/management/audit_log_plugin.html
[20] MySQL. (2022). MySQL 8.0 Reference Manual / MySQL Enterprise Audit. Retrieved March 2, 2022 from https://dev.mysql.com/doc/refman/8.0/en/audit-log.html
[21] Shichkin, A., Buevich, A., Sergeev, A., Baglaeva, E., Subbotina, I., Vasilev, J., & Kehayova-Stoycheva, M. (2018). Training Algorithms for Artificial Neural Network in Predicting of the Content of Chemical Elements in the Upper Soil Layer. In AIP Conference Proceedings (Vol. 2048, No. 1, p. 060004). AIP Publishing LLC. https://doi.org/10.1063/1.5082119

7 Authors

Pavel Petrov is working as Associate Professor in the Department of Informatics and Computer Science at the University of Economics – Varna, Varna, Bulgaria. His research interests include distributed web systems, big data, and cloud computing. (email: petrov@ue-varna.bg).

Ivan Kuyumdzhiev is working as Associate Professor in the Department of Informatics and Computer Science at the University of Economics – Varna, Varna, Bulgaria. He is Director of the Center for Research and Application of New Information and Communication Technologies (CIPNICT). (email: ivan_ognyanov@ue-varna.bg).

Georgi Dimitrov is working as Professor in the University of Library Studies and Information Technologies, Sofia, Bulgaria. He is Vice Dean of Department of Information Technology. (email: g.dimitrov@unibit.bg).

Adelina Georgieva Kremenska is a PhD Candidate, Institute of Robotics, Bulgarian Academy of Sciences, Acad. Georgi Bonchev str., 1113 Sofia, Bulgaria. (email: a.kostova012@gmail.com)

Article submitted 2022-06-18. Resubmitted 2022-08-12. Final acceptance 2022-08-12. Final version published as submitted by the authors.