UNIVERSITY OFNAIROBI

COLLEGEOFBIOLOGICALANDPHYSICALSCIENCES

SCHOOL OFCOMPUTINGANDINFORMATICS

RESEARCH REPORT

Title: A Methodology to Test the Richness of Forensic Evidence of Database Storage Engine: Analysis of MySQL Update Operation in InnoDB and MyISAM Storage Engines.

BY: JAMES ORENGO OGUTU

P53/72922/2014

SUPERVISOR: DR.ELISHA O. ABADE.

A research Report submitted in partial fulfilment of the requirements for the award of Master of Science Degree in Distributed Computing Technology.

November, 2016

ABSTRACT

Digital forensic investigation requires forensic evidence data to prove a claimed crime. Forensic evidence can either be volatile or persistent wherein persistent evidence is of great importance while investigating a case in a system that has once been shut down or powered off after the claimed violation since volatile evidence will disappear when the system is powered off. With the possibility of performing database forensic as a file system coupled with the fact that there are several storage engines that can be implemented in a database, there is need to know the forensic implication of using a particular storage engine with focus on how much forensic footprint it leaves behind. This work investigated the impact of MyISAM and InnoDB storage engines in generation of persistent forensic data in MySQL DBMS system. A comparison was done on the number of logs and files affected by an update operation in MySQL DBMS implementing either of the storage engines. It was found that more files were affected in InnoDB than in MyISAM implementation.

TABLE OF CONTENTS Page

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS AND ABBREVIATIONS

Artefacts	-	Traces left in the system as a result of an activity
Digital evidence	-	Proof of an activity
DBMS	-	Database Management System
RDBMS	-	Relational Database Management System
MAC time	-	Modify Access Create Time
ACID	-	Atomicity, Consistency, Isolating, and Durability
FTK	-	Forensic Tool Kit
ANSI	-	American National Standard Institute

CHAPTER ONE: INTODUCTION

1.1 Background to the Study

The need for an organized manner of storing information belonging to an organization was the motivation for coming up with the concept of a database. Databases have evolved from simple lists of items to more complex transactional databases that are run on complex computer systems. A database can be defined as: "persistent, logically coherent collection of inherently meaningful data, relevant to some aspect of the real world" (Robbins, 1994).

Databases play a pivotal role in businesses as they store variety of data from asset inventory to orders and financial transactions. With the up surge of databases in many business applications, there comes the need to maintain integrity, consistency and reliability of the data stored in the databases. Some of these functions have been integrated in the software systems that are used to run database applications which restrict access by requiring user to authenticate using passwords. These applications are called database management systems (DBMS). However, DBMSs can only ensure integrity, consistency and reliability with the assumption that the system is configured properly and authorised users use the system in the authorised way. But because this may not always happen in real world, there has been need to provide extra measures to mitigate the effect of unauthorised manipulation of the database by authorised or authorised database user acting maliciously. The role of database security expert then comes in to ensure that the data contained in the database is reliable and protected from unauthorized access and manipulation, and in the event that there is a disputed or unlawful manipulation, then an acceptable procedure and evidence can be used to prove that the claimed violation actually took place. The act of proving a past occurrence in database using acceptable data evidence constitutes database forensics. The aim of database forensics is to find out and prove what happened when and to prevent unauthorized data manipulation (Weippl, 2010).

Analysis of architectural components of a database system is critical for the understanding of database forensic analyst in which a forensic examiner identifies areas

to get evidence about a past activity in a database. Some of the areas in a database where evidence about past activities can be found include metadata, data files; redo logs, transaction logs and storage engines. With the constant security threat to databases and the possibility of data being altered, there has been a deliberate effort in the research community to try to resolve some breaches and challenges in the database world. An overview of some common database systems is given below.

1. **MS Access** -Was developed by Microsoft Corporation and stores data in its own format based on the Access Jet database engine
2. **MySQL** -Is an open-source DBMS developed by MySQL AB Company and uses multiple storage engines. It is the most popular DBMS among open source DBMSs (The Windows Club, 2015).
3. **Oracle** - An object oriented DBMS Developed by Oracle Corporation.
4. **Microsoft SQL Server** – Is a relational database server developed by Microsoft Corporation.
5. **Other DBMSs** include Ingress, Postgress, and InterBase.

MySQL has been chosen for this research because of its popularity amongst users. According to database storage engine ranking site, http://db-engines.com, MySQL has remained second in popularity after Oracle in the year 2014, 2015 and 2016 consecutively. Holc *et. al*, 2008 estimated that there were 500,000 downloads per day and 7.5 billion active user of MySQL world-wide. In addition to this, MySQL is open source. The motivation for MySQL popularity is fuelled by the fact that it is open source in addition to its versatility (Bassil, 2012).

The figure below is an architectural illustration of MySQL database system.

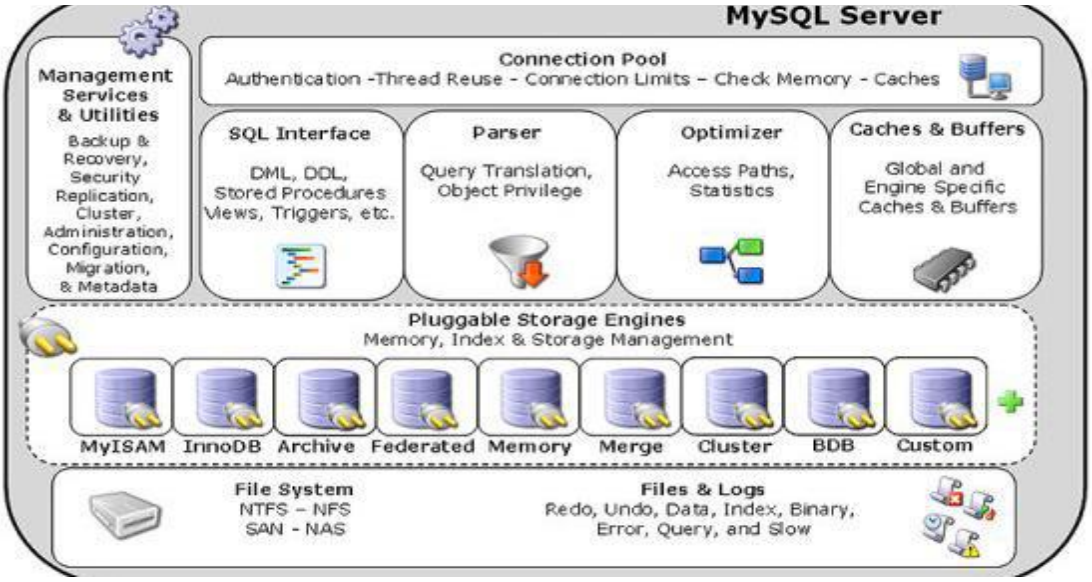


Figure 1: MySQL Architecture (Courtesy of: http://www.datadisk.co.uk).

The architectural illustration of MySQL DBMS depicts its major components in the hierarchy of their interaction from user side on the upper level to the file system at the lower level. The highest level interfacing with applications connecting to the database is the connection pool that offers authentication to users, manages the use of threads, establishes connection limits, and checks memory and caches. The management services and utility manages backups and recovery, security replications, clusters, administrator configurations, migration and metadata. The storage engine performs memory, index and storage management. The operations of the storage engine write data to the files and logs residing in the file system. By analysing what data has been written to the logs and files in the file system, one can get evidence of an operation that was executed in the database. Evidence data can either be volatile or persistent. Volatile evidence data is the type of data that disappears when the machine is switched off or restarted while persistent evidence data is that data that does not disappear with system shutdown nor restart.

1.2 Problem Statement

MySQL is one of the most widely used DBMSs. It uses structured query language (SQL) and is able to handle large databases in much faster way than existing applications. MySQL offers multi-user, multi-thread and robust server storage system. It has the capability to implement different storage engines depending on what type of operations

the system is targeting, that will leverage specific capabilities of a particular engine. The most popular storage engines for MySQL are MyISAM and InnoDB (Tocci, 2013). The strength of MyISAM is in the referential and read-only applications and full-text searches while its weakness is that it doesn't support transactional operations. InnoDB on the other hand supports transactional operations because of its ACID (Atomicity, Consistency, Isolating, and Durability) compliance. MyISAM was the default storage engine in earlier versions of MySQL but from MySQL 5.5 and above versions, InnoDB has remained the default storage engine (Oracle, 2014). The use of MyISAM and other storage engines can still be specified and be implemented in these later versions where InnoDB is the default storage engine.

The interchangeable use of MyISAM and InnoDB storage engines in MySQL applications has been of interest of researchers to explore the advantages and disadvantages of using either (Oracle, 2014). Some of the differences between MyISAM and InnoDB that have been documented are listed in the table below.

Feature	InnoDB	MyISAM
ACID transactions	Yes	No
Crash Safe	Yes	No
Foreign Key Support	Yes	No
Full text Searches	No	Yes
B-Tree Indexes	Yes	Yes
Raw-level Locking granularity	Yes	No (Table level)

Table (1.): Comparison between MyISAM and InnoDB storage engines

From the table above, it can be seen that comparative analysis of MyISAM and InnoDB storage engines done in the past have been focused on performance, versatility and data type handling capabilities of the two storage engines. While there are researches that have touched on forensics of databases implementing the various storage engines, none has given a comparative analysis of the forensic consequences of implementing either

MyISAM or InnoDB. This research undertook to perform a comparative analysis of MyISAM and InnoDB storage engines to find out if the implementation of either can influence the level of persistent forensic evidence data in database forensics. The focus of this research was be on the richness of persistent data as generated by the two storage engine implemented independently on separate identical MySQL installations. The richness was defined by the multiplicity of the locations evidence data is written when an identical update operation is performed. The storage engine implementation with higher number of evidence locations was deemed richer because it provides a forensic investigator with more location for sources of proof.

1.3 Research Objectives

The general objective of this research undertaking was to find out the forensic implications of using MyISAM or InnoDB storage engines in MySQL database. This would strengthen forensic discourse for choice of storage engine at policy formulation level. The product of this research is a report that details the implications of implementing either of the storage engines which enables policy makers to make informed decision with forensic consideration.

Main Objectives

1. To investigate the forensic implications of using MyISAM or InnoDB storage engines in MySQL database.
2. To study how MyISAM and InnoDB storage engines individually generate persistent forensic evidence data in the logs.
3. To perform an analysis to determine the number of occurrence of the persistent forensic data in each case and make a comparison.

1.4. Significance of the research

As the field of digital forensics matures, forensics should be part of policy consideration rather than being a post-occurrence undertaking that aims at trying to unveil or reconstruct a past activity. Rather, policy makers should bear in mind the possibility that

there would be need to carry out some investigation in future and therefore make a choice of forensically friendly system components. The findings of this research will give a forensic standpoint for policy makers if a choice is to be made between the two storage engines considered in this work.

1.5. Scope and limitation

This research focused on two storage engines namely MyISAM and InnoDB even though there are several other storage engines, these being just two among other several engines, more research need to be done in comparability of the storage engines before the results are generalized. However the choice of these storage engines is reasonable to forensic community because MyISAM and InnoDB are the only storage engines that have been fronted as default storage engines for MySQL and have therefore enjoyed popularity that all other storage engines have not. In the aspect of evidence data, this research limit itself to serializing of files that hat their metadata changed after UPDATE operation and did not consider persistent data not residing in the files, volatile class of data or comparing the actual contents of the files.

The limitation of the research was that the comparison was done based on observable changes in file metadata and not considering the actual content of the file or volatile data that may perhaps have stronger evidence than findings of this research.

1.6 Assumptions

It was assumed that:

1. The changes observed on the metadata of the target files were exclusively as a result of the UPDATE operation as implemented by the specific storage engine.
2. The installations of MySQL were properly configured and working correctly.
3. The analysis tool that was used was working correctly and gave the correct result.

The rest of this document is organised in the following order; chapter two discusses previous works relevant to the study, chapter three discusses the methodology, design, procedures and tools used in the study. Results and their interpretations are finally given in chapter five.

CHAPTER TWO: LITERATURE REVIEW

2.1Related Works

A database is a collection of information that is organized so that it can easily be accessed, managed and updated (Sheing, 2006). This section of the document will highlight past research works done on database systems that are relevant to database forensics.

With the definition of database as a collection of organised information, (Rodriguez, 2004) explains how this collection of organized data can be managed using a database management system (DBMS) which is a combination of data, hardware, software and users. Database hardware is a standard computer system with memory. The statements that are used for manipulating data in a database are structure query language (SQL) which performs retrieval and update of data. Retrieval is the collecting of data from the database for data that match the specification of the user query while updating involves modification, deletion and insertion of data into the database. This work also highlights various database architectures such as; functional, application and logical architectures. Database can also assume two or multi-tier architecture. The logical architecture, also known as ANSI architecture has three distinct layers of data abstraction which are physical, logical and user layer.

Relational database model has been discussed by (Sheing, 2006) as the foundation of the contemporary database. It consists of tables which are classes of data structures, relational algebra that are the methods used to build a new table from the initial one and constraints that are imposed on the data contained in the tables. Tables in a relational database have three distinguished features; table name, the heading of the content (each as a column entry), and the content of the table as list of rows. Referential integrity ensures that the entity being referred to in the table by users has a meaningful value. Weippal (2010) stated that the aim of database forensics is to find out what happened when and to revert unauthorised data manipulation.

Khanuja and Adane (2011) stressed that when carrying out database forensic, one needs to ensure that scientifically proven methods are used to gather, process, interpreter digital

evidence in order to give true reconstruction of the criminal activity. This work gives methodologies for tamper detection in database using audit logs. It also explores the vulnerabilities of using audit logs to perform database forensics in that a criminal can change their contents to hide his criminal activities.

This work points at places to look for evidence when undertaking database forensics; which are; system metadata, data files, redo logs, transaction logs and memory and trace files. Some temper detection methodologies are also described and they include: Notarization hash verification, more specialized forensic analysis algorithms such as monochromatic, RGB, Tilled-bitmap and 3D are also described. Finally they categorize artefacts in database forensics as resident or non-resident. Resident artefacts are those artefacts that are found within files and memory locations strictly reserved for SQL server while non-resident artefacts that are found in files not explicitly reserved for SQL server use.

Progress made in database research can be found in (Hauger and Oliviery, 2015). This work gives the two approaches to a database by forensic experts while performing database forensic analysis; one way is to look at a database as files residing in file system therefore database forensic can just be performed like forensics of other important software applications like emails or web browsers in this aspect, database forensic viewed as a sub discipline of file system forensic and some techniques like imaging and file carving are applicable. The other approach to database forensic is to view a database as complex multidimensional system with numerous interconnected components that should be analysed together to expose accurate truth, crucial components considered here are data model, data dictionary, application scheme and the application data.

A framework for performing database forensic analysis is given in (Khanuja and Adane, 2012). This framework gives a guide in how to undertake forensic activities of identifying, preserving, collecting, analyzing, validating and interpreting digital evidence in a database and finally generating a report.

By the virtue that DBMS write data items to multiple location and copies such as in tables' indexes, logs materialized views and temporary relations view then when data is

deleted in one location, it is not completely destroyed but its traces are left in some database locations. These data traces can be recovered by performing forensics which will extract data and information from database, logs, cache, data files, and table space e.t.c. The undo logs which show older version of data and the redo log that stores information used during crash to restore the status (recovery) are also high lightened.

Alexander (2014) describes what to look at when performing database forensic analysis on MySQL server. The work describes places where artefacts may exist in MySQL server which include server files such as database transaction logs, query logs, query cache and key cache.

InnoDB as common and popular database storage engine is explored by (Fruhwist, 2010).The storage pattern of InnoDB in MySQL is described. InnoDB stores information about each table in the directory of the database as a .frm file with the table name as the file name and the size of the .frm is limited to four GB beyond which is transacted. By default InnoDB stores all data of all tables in a single file. The InnoDB storage format parts are; fill header, page heading, infinum and .supernum, user records, free space, page directory and fill trailer are also discussed. This work finally proposes a tool that reads hexadecimal data from the form then uses it to reconstruct the table and then uses the table information to locate the data in the data storage file.

A database forensic approach using log files is presented in (Charana and Khanuja, 2014). It highlights the procedure for carrying out digital forensic using log files and applying standard forensic steps of identification, acquisition and presentation, examination and analysis, and finally documentation. However, due to complexity of databases, database centric forensic follow steps of acquisition and presentation, collection and analysis. The logs maintained by MySQL are also discussed, they include error log, general query log, binary log, update log and slow query log. This work gives a two part framework for database forensic analysis. The first part depicts the user performing an action in the database and the second part is collecting and analysing forensic data from the central database.

The binary log has an update log that contains information needed to recreate the database since the server was last restored or the logs were flashed. A list of every query that changed the data can also be found by passing the log-bin-option to the SQL server (Mysqld). Whenever SQL server starts a new session, it opens a new log file in addition to the old one such that if the previous log file was aden-bin.000001 then the new file will be incremented to aden-bin.000002.SQL statements in the binary log file can be viewed by using sqlbinlog command with full path of the binary log file for example, program data/mysql/data/binlog. Procedures for accessing error log, querylog, redolog, undolog and index logs are given.

While the existence of log files content in database are important for forensic reconstruction, their existence can also be seen in bad side in that they can be exploited to expose privacy. Grebhain, (2013) gives a user defined deletion process for data in MySQL database in order to maintain privacy. As opposed to deletion by overwriting and setting a delete bit, this approach deletes data from all inventories created by the system. It gives a propagation strategy that performs deletion on multiple areas of a database.

2.2 Justification

While several researchers have undertaken comparative works between InnoDB and MyISAM storage engines, little has been done in forensic perspective to explore the forensic implication of using either of the storage engines. The aim of this research was to bring out the strength of either of the storage engines in generating persistence forensic evidence. It is in the view of this work that the storage engine that produces the most persistent footprints will enable forensic analyst to retrieve more evidence for a case even if the database system was once switched off. The results of this research will contribute to the forensic discourse of having forensic consideration at policy making level rather than undertaking forensic as a last resort when an inversion has already taken place.

2.3 Conceptual Framework

The concept of this research was based on fact that MySQL DBMS can be implemented with a variety of storage engines. It is also documented that the various storage engines have different ways in which they operate and manipulate data, MySQL 5.6 Reference

Manual (2013). It is in this sense that different storage engines are expected to generate forensic evidence differently. The conceptual framework depicts the instances if InnoDB and MyISAM storage engines interaction with MySQL DBMS and the scenarios of logs artefacts analysis for each storage engine instance. The end results are two file systems analyses for each storage engine implementation. Each corresponding file analysis, that is; before the UPDATE and switch off and then after UPDATE and switch off were compared and conclusion made on which storage engine implementation affects more files out of the listed file targets.

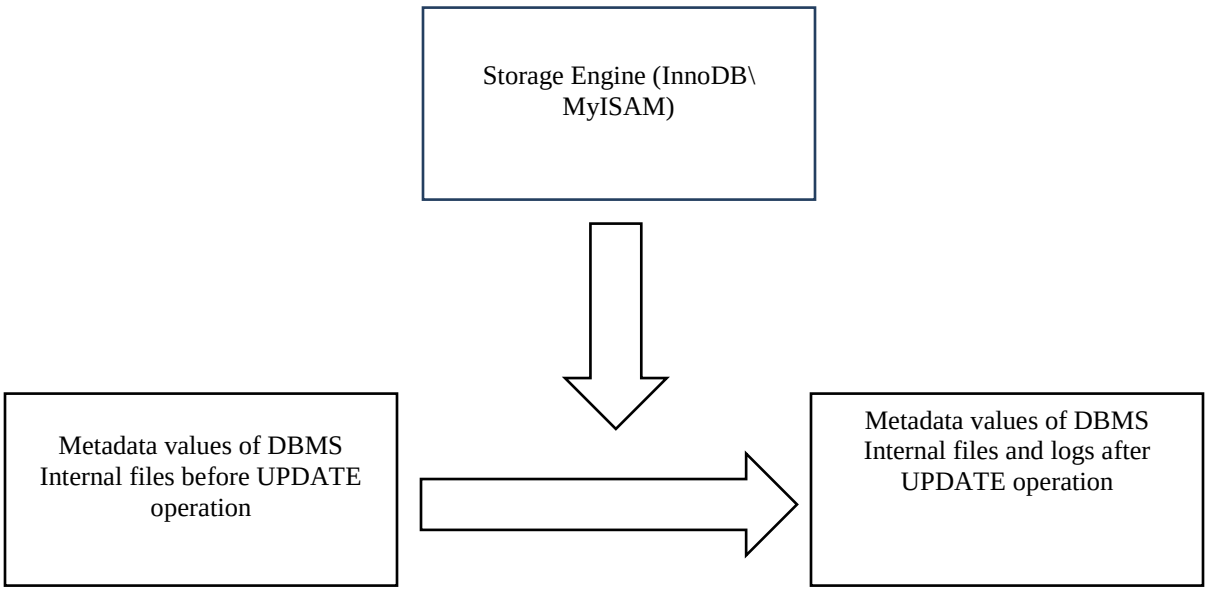


Figure **2: Conceptual Framework**

This conceptual framework shows that the storage engine implemented has an influence in how an update operation writes to MySQL internals and consequently affects the values of metadata. With the implementation of either InnoDB or MyISAM storage engine, there was a presumed distinct system that is autonomous in the way it generates and writes forensic data. Each instance of storage engine had the analysis of the artefacts done in comparison to the same artefacts from the other storage engine implementation.

CHAPTE THREE: RESEARCH METHODOLOGY

3.1 Methodology

The methodology for this research was qualitative. This is because the research entailed exploring and analysing the effect of the used storage engines to MySQL internals. Hancock *et* al. (1998) elaborated that the application of qualitative methodology is appropriate in research undertakings where the result or observations to be made cannot be expressed in numbers; rather, they are explained and illustrated in words.

While tools were be used to read file information, the outcome is explained in words after observation. The observations made were then be compared appropriately.

3.2Design

This research assumed experimental design. The subjects of the experiment were the following files; Transaction logs, Redo logs, Index logs, Query logs, Error logs Undo logs and Master Data File. The treatments were the two storage engines (MyISAM and InnoDB).After installing the storage engines to separate instance of MySQL and performing identical update operation on both, the observed changes brought about by respective storage engines were compared. The specific parameters looked at were; MAC times, size change of the file and the MD5 has value.

3.3 Research Tools

The use of LogExpert and SQL-Recovery as tools for the research was dropped because for these tools to give meaningful result, they would require large log data as well as long database operation time in order to have several functions and operation executed in the database. This would require time and more resources to implement. Going with the general objective of this work which was :" To find out the forensic implications of using MyISAM or InnoDB storage engines in MySQL database", the objective would d still be achieved by performing forensic analysis of a database as a file system and target the same files that were to be analysed using log editor.

By using file system forensics approach, all the target files in the data base have their metadata and content changes viewed and analysed using forensic tool for later comparison. Autopsy (Sleuth Kit) for Windows is used in this analysis.

Autopsy is an open source forensic tool available for both windows and Linux platforms. In this research we used Autopsy version 4.0 because of its stability and the ability to give full view of files and folders and their metadata such as name, file type, size, modified time, accessed time, created time, and the MD5 hash of the file. These parameters will be the basis for viewing and analysing the changes that have taken place in the file in question.

Our target files are as follows:

- Transaction log
- Redo log
- Index log
- Query log
- Error log
- Master Data File (MDF)

Applications, Tools and Equipment used in this experiment are as follows:

- MySQL 5.5.
- MySQL Workbench 6.3 CE.
- Access Data FTK imager.
- Autopsy sleuth Kit
- Two desktop computers installed with Windows 7 operating system.

3.4.1Procedure

1. Two freshly wiped computers were prepared and installed with window operating system.
2. On each machine, MySQL (Version 5.5) was installed. One installation of MySQL with InnoDB storage engine and the second one with MyISAM storage engine.
3. MySQL Workbench (MySQL graphical user interface) was installed to each computer to be used for database operation.
4. Identical databases (named 'Customer accounts') were created in each MySQL installation and in each database an identical table (name, records and settings) is created and populated with similar records.
5. MySQL instances were stopped and the machines powered off for a short time. The machines were then powered on and at this time it was assumed that all the volatile evidence data got discarded when the machines were powered off.
6. Using FTK imager activated from USB derive, an image of the logical drive where MySQL was installed was created and the image saved in a clean, wiped external evidence hard drive. This was done to both installations.
7. MySQL instances were started again.
8. With the records in each table known and identical in both MySQL installations, a specific record was updated to a common value in both systems.
9. MySQL instances were stopped and the machines were powered off to discard volatile data.
10. A second set of images was taken using FTK imager and saved as in (6) above.
11. At this point there are four images to be subjected to analysis. Two images based on MyISAM and another two based on InnoDB storage engines. These images were then ingested and analysed one by one using Autopsy analysis procedures.

3.4.2Measurement Metrics

The measurement metrics for this research was the number of files whose metadata changed after performing the update operation. More counts of affected files meant

potentially richer permanent forensic data while less count meant weaker permanent forensic evidence data

CHAPTER FOUR: RESULT, ANALYSIS AND INTERPRETATION

4.1Introduction

The focus of the proposal was to identify the internal files of MySQL DBMS that bear permanent evidence of a database operation (UPDATE in this case) as influenced by the storage engine used by analysing the metadata of the files under investigation. The experiment implemented an UPDATE operation in two database installations; one implementing InnoDB Storage engine and the second one implementing MyISAM storage engine.

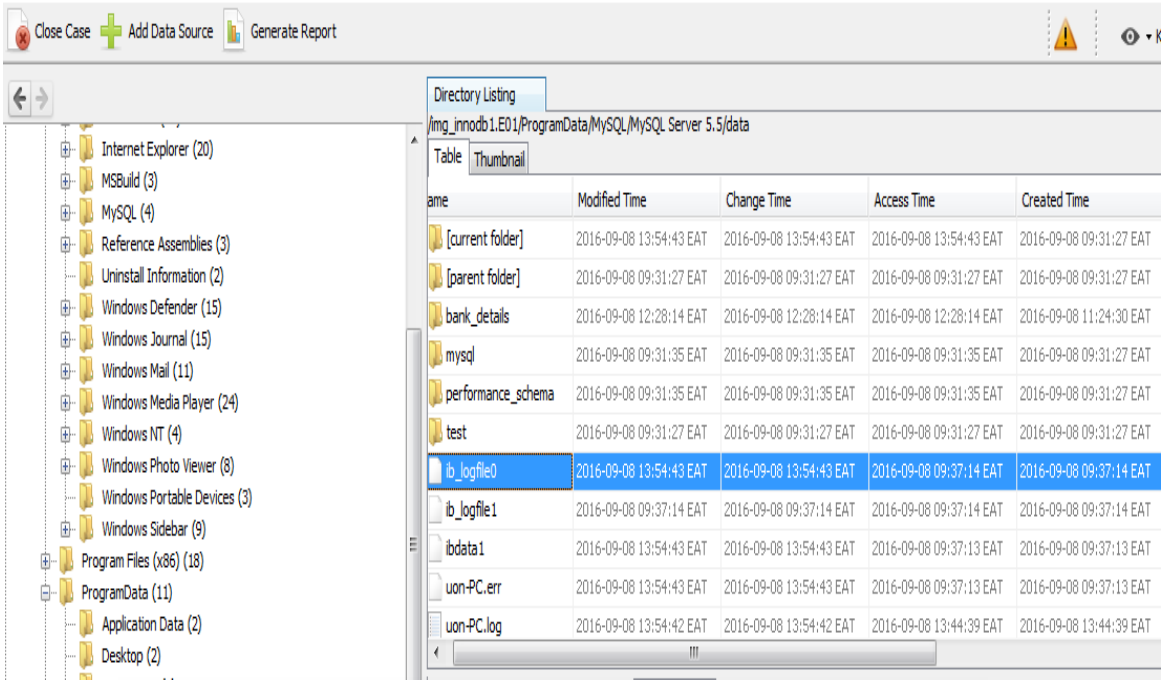
Our target files are as follows:

- Transaction log
- Redo log
- Index log
- Query log
- Error log
- Master Data File (MDF)

After ingesting the image, Autopsy performs analysis and gives file information in the following order;

- Size -Represents size of the file
- Modified -Shows when the content of the file most recently changed
- Accessed -Shows when the file was most recently opened for reading
- Created -Represents the time of creation of the file
- Changed/Change -When the file was first created or had the meta data changed
- MD5 -the MD5 hash value of the file

The figure below shows Autopsy navigation window where file system can be explored and metadata viewed with transaction log ib_log0 selected.



Figure, 3. Autopsy result window

4.2 Analysis of the Results.

In this section the analysis results are presented. The section is divided into two parts; the first part presents the result of the analysis of the installation implementing InnoDB storage engine while the second part presents the analysis result of the installation implementing MyISAM. In each section, the results are organized in the order of the target files. In each case a brief description of the target file is given, the forensic analysis result of the file as present in the image taken before the UPDATE operation followed by the forensic analysis result of the file as present in the image taken after UPDATE operation. Finally, a brief observation and comparison for the result of the file in consideration will be illustrated.

4.2.1 InnoDB Image Results.

i. Transaction log

Transaction log keeps log of all queries that have changed something in the database. It is the equivalence of binary log in mysql 5.5(MySQL 5.6 Reference manual) it is located in C:\program files/program data/mysql/mysql server5.5/data/mysql/ndb_binlog. The

logical paths to these files in the result may look different from the norm because all are based on the image data is the base drive.

Transaction log before update

Name	/img_innodb1.E01/ProgramData/MySQL/MySQL Server 5.5/data/mysql/ndb_binlog_index.frm
Type	File System
Size	8778
File Name Allocation	Allocated
Metadata Allocation	Allocated
Modified	2016-09-08 08:12:40 EAT
Accessed	2016-09-08 09:31:35 EAT
Created	2016-09-08 08:12:40 EAT
Changed	2016-09-08 09:31:38 EAT
MD5	6f38a874f706c1883bd40fb7b0e72be2

Transaction log after UPDATE

Name	/img_InnoDB After Update.E01/ProgramData/MySQL/MySQL Server 5.5/data/mysql/ndb_binlog_index.frm
Type	File System
Size	8778
File Name Allocation	Allocated
Metadata Allocation	Allocated
Modified	2016-09-08 08:12:40 EAT
Accessed	2016-09-08 10:13:35 EAT
Created	2016-09-08 08:12:40 EAT
Changed	2016-09-08 10:21:14 EAT
MD5	7d389b674f706c1883bd40fb7b0e72be2

Observation: The created time, modified time and file size has remained the same in the image before UPDATE and after UPDATE. However the access time, change time and MD5 have changed after UPDATE. This shows that the file was changed during the update operation since the value of the change time corresponds to the time the UPDATE operation was executed. The difference in MD5 hash value shows that the current state of the file is different from the previous state.

ii. Re-Do log.

This is physically present in the disk by default as a set of files going by the name; ib_logfile0 and ib_logfile1. They are used during crash recovery to correct data written by a transaction that did not complete executing. During normal operations, the redo log encodes requests to change InnoDB table data that result from sql statements. The two data structures are as bellow.

Ib_logfile0 before update

Name	/img_innodb1.E01/ProgramData/MySQL/MySQL Server 5.5/data/ib_logfile0
Type	File System
Size	10485760
File Name Allocation	Allocated
Metadata Allocation	Allocated
Modified	2016-09-08 08:12:40 EAT
Accessed	2016-09-08 09:31:35 EAT
Created	2016-09-08 08:12:40 EAT
Changed	2016-09-08 09:31:30 EAT
MD5	7c6fbab3c474a8a07c7c7031bc58e1cb

Ib_logfile0 after update

Name	/img_InnoDB After Update.E01/ProgramData/MySQL/MySQL Server 5.5/data/ib_logfile0
Type	File System
Size	10485760

File Name Allocation	Allocated
Metadata Allocation	Allocated
Modified	2016-09-08 08:12:40 EAT
Accessed	2016-09-08 09:31:35 EAT
Created	2016-09-08 08:12:40 EAT
Changed	2016-09-08 09:31:38 EAT
MD5	e2adb6b0586e713ae74292c336d5aeee

Observation: All parameters have remained the same apart from the MD5 value of the file. This shows that there was a change made to Ib_logfile0 before file during update operation.

Ib_logfile1 before UPDATE

Name	/img_innodb1.E01/ProgramData/MySQL/MySQL Server 5.5/data/ib_logfile1
Type	File System
Size	10485760
File Name Allocation	Allocated
Metadata Allocation	Allocated
Modified	2016-09-08 08:12:40 EAT
Accessed	2016-09-08 09:31:35 EAT
Created	2016-09-08 08:12:40 EAT
Changed	2016-09-08 09:31:37 EAT
MD5	f1c9645dbc14efddc7d8a322685f26e9

Ib_logfile1 after update

Name	/img_InnoDB After Update.E01/ProgramData/MySQL/MySQL Server 5.5/data/ib_logfile1
Type	File System
Size	10485760
File Name Allocation	Allocated
Metadata Allocation	Allocated
Modified	2016-09-08 08:12:40 EAT
Accessed	2016-09-08 10:13:35 EAT
Created	2016-09-08 08:12:40 EAT

Changed	2016-09-08 10:13:37 EAT
MD5	f1c9645dbc14efddc7d8a322685f26eb
Hash Lookup Results	UNKNOWN
Internal ID	182038

Observation: The file sizes of Ib_logfile1have remained the same while the MD5 values are different showing that there was a modification to the file.

iv. General Query log.

The General Query log records what mysqld is doing. The server writes information here when a client connects or disconnects and logs each sql statement received from each client. This file's location is in the path: programdata/mysql/mysql server 5.5/data/uon-pc.log

General query log before update

Name	/img_innodb1.E01/ProgramData/MySQL/MySQL Server 5.5/data/uon-PC.log
Type	File System
Size	1096
File Name Allocation	Allocated
Metadata Allocation	Allocated
Modified	2016-09-08 08:20:42 EAT
Accessed	2016-09-08 08:20:42 EAT
Created	2016-09-08 08:20:42:EAT
Changed	2016-09-08 13:54:42 EAT
MD5	5bf18f35d316d2938a7b00f3617b44dd

```
From The Sleuth Kit istat Tool:

MFT Entry Header Values:
Entry: 70034        Sequence: 77
$LogFile Sequence Number: 780114566
Allocated File
Links: 1

$STANDARD_INFORMATION Attribute Values:
Flags: Archive, Not Content Indexed
Owner ID: 0
Security ID: 982  (S-1-5-32-544)
Last User Journal Update Sequence Number: 239710320
Created:    2016-09-08 08:20:42.198701100 (E. Africa Standard Time)
File Modified:    2016-09-08 08:20:42.108560100 (E. Africa Standard
Time)
```

```
MFT Modified:     2016-09-08 08:20:42.108560100 (E. Africa Standard
Time)
Accessed:   2016-09-08 08:20:42.198701100 (E. Africa Standard Time)

$FILE_NAME Attribute Values:
Flags: Archive, Not Content Indexed
Name: uon-PC.log
Parent MFT Entry: 57826       Sequence: 3
Allocated Size: 0       Actual Size: 0
Created:    2016-09-08 08:20:42.198701100 (E. Africa Standard Time)
File Modified:    2016-09-08 08:20:42.108560100 (E. Africa Standard
Time)
MFT Modified:     2016-09-08 08:20:42.108560100 (E. Africa Standard
Time)
Accessed:   2016-09-08 08:20:42.198701100 (E. Africa Standard Time)

Attributes:
Type: $STANDARD_INFORMATION (16-0)   Name: N/A   Resident   size: 72
Type: $FILE_NAME (48-2)   Name: N/A   Resident   size: 86
Type: $DATA (128-3)   Name: N/A   Non-Resident   size: 1096  init_size:
1096
1513850
```

General Quer log after UPDATE

Name	/img_InnoDB After Update.E01/ProgramData/MySQL/MySQL Server 5.5/data/uon-PC.log
Type	File System
Size	2885
File Name Allocation	Allocated
Metadata Allocation	Allocated
Modified	2016-09-09 08:20:42 EAT
Accessed	2016-09-08 10:13:39 EAT
Created	2016-09-08 08:20:42 EAT
Changed	2016-09-09 10:13:39 EAT
MD5	418fa732cf5570ac08e82c6e89227a7b
Hash Lookup Results	UNKNOWN
Internal ID	182142

……

```
From The Sleuth Kit istat Tool:

MFT Entry Header Values:
Entry: 70034        Sequence: 77
$LogFile Sequence Number: 822891138
Allocated File
Links: 1
```

```
$STANDARD_INFORMATION Attribute Values:
Flags: Archive, Not Content Indexed
Owner ID: 0
Security ID: 982  (S-1-5-32-544)
Last User Journal Update Sequence Number: 246616720
Created:       2016-09-08 08:20:42.198701100 (E. Africa Standard Time)
File Modified: 2016-09-09 10:13:43.105062600 (E. Africa Standard Time)
MFT Modified:  2016-09-09 08:20:42.105062600 (E. Africa Standard Time)
Accessed:      2016-09-08 08:20:42.198701100 (E. Africa Standard Time)

Attributes:
Type: $STANDARD_INFORMATION (16-0)   Name: N/A   Resident   size: 72
Type: $FILE_NAME (48-2)   Name: N/A   Resident   size: 86
Type: $DATA (128-3)   Name: N/A   Non-Resident   size: 2885  init_size:
2885
1513850
```

Observation: The content of the file in both cases match the time that the queries were run. Line 1 in the metadata in the image before UPBATE shows; '`Created: 2016-09-08 08:20:42.198701100 (E. Africa Standard Time)`' which is the time at which the database was created while the attribute 'File Modified:2016-09-09 10:13:43.105062600 (E. Africa Standard Time)' in the image after UPDATE show when UDATE was run. I t can be seen that this file captures the activities with their timeline however it doesn't show the exact sql statement. The file was affected by UPDATE operation.

(v). Error log.

This contains the information of when the server was started and stopped.

Error log before update

Name	/img_innodb1.E01/ProgramData/MySQL/MySQL Server 5.5/data/uon-PC.err
Type	File System
Size	6562
File Name Allocation	Allocated
Metadata Allocation	Allocated
Modified	2016-09-08 13:54:43 EAT
Accessed	2016-09-08 09:37:13 EAT
Created	2016-09-08 09:37:13 EAT

Changed	2016-09-08 10:54:43 EAT
MD5	234883c6ef0f62999513388cb3bab8ca
Hash Lookup Results	UNKNOWN
Internal ID	176109

Error log after UPDATE

Name	/img_InnoDB After Update.E01/ProgramData/MySQL/MySQL Server 5.5/data/uon-PC.err
Type	File System
Size	8328
File Name Allocation	Allocated
Metadata Allocation	Allocated
Modified	2016-09-08 10:54:55 EAT
Accessed	2016-09-08 09:37:13 EAT
Created	2016-09-08 09:37:13 EAT
Changed	2016-09-00 10:14:55 EAT
MD5	00fd2b14b8c5be23501887375cc31daa

Observation: it can be seen that the size of the file has increased from 6562 to 8328. This is reflective of the behaviour of error log in that its content increases with continued operation of the database. The specific information can be mined from the log based on the timeline of the activity.

vi. Undo logs

This is the storage area that holds versions of data modified by a currently active transaction. It helps in ensuring transactional consistence and accuracy. It helps in ensuring that another transaction sees the original version of data as a part of consistent read operation. This area is physically part of system table space.

This concept will not be of importance in this research because the product of a rollback segment is .ib data file which has the same content present in the general query log and redo log.

vii. The .frm file:Account Balance.frm

This is the representation of the table on the disk that describes the table format. It bares the same name as the table.

.frm before update

This is a representation of the structure of the table on the disk.

Name	/img_innodb1.E01/ProgramData/MySQL/MySQL Server 5.5/data/bank_details/account_balances.frm
Type	File System
Size	8716
File Name Allocation	Allocated
Metadata Allocation	Allocated
Modified	2016-09-08 08:29:44 EAT
Accessed	2016-09-08 08:29:44 EAT
Created	2016-09-08 08:23:39 EAT
Changed	2016-09-08 08:28:14 EAT
MD5	2e5e8bc2c1f5029c640a230aaee07f8b
Hash Lookup Results	UNKNOWN
Internal ID	176002

.frm after update

Name	/img_InnoDB After Update.E01/ProgramData/MySQL/MySQL Server 5.5/data/bank_details/account_balances.frm
Type	File System
Size	8716
File Name Allocation	Allocated
Metadata Allocation	Allocated
Modified	2016-09-08 08:29:44 EAT
Accessed	2016-09-08 08:29:44 EAT
Created	2016-09-08 08:23:39 EAT
Changed	2016-09-08 08:28:14 EAT
MD5	2e5e8bc2c1f5029c640a230aaee07f8b
Hash Lookup Results	UNKNOWN
Internal ID	182034

Observation: It can be seen that there were no changes in the metadata of the .frm in the two images. This is theoretically correct because the .frm file does not change if there is no change in the structure of the table.

4.2.2. **MyISAM Image Results**

i. Transaction log

Transaction log keeps log of all queries that have changed something in the database. It is the equivalence of binary log in mysql 5.5(MySQL 5.6 Reference manual) it is located in C:\program files/program data/mysql/mysql server5.5/data/mysql/ndb_binlog. The logical paths to these files in the result may look different from the norm because all are based on the image data is the base drive.

Transaction log before update

Name	/img_myisam1/ProgramData/MySQL/MySQL Server 5.5/data/mysql/ndb_binlog_index.frm
Type	File System
Size	8778
File Name Allocation	Allocated
Metadata Allocation	Allocated
Modified	2016-09-08 13:42:33 EAT
Accessed	2016-09-08 13:43:33 EAT
Created	2016-09-08 13:42:33 EAT
Changed	2016-09-08 13:4:33 EAT
MD5	7c38a874f706c1883bd40fb7b0e72be2

Transaction log after UPDATE

Name	/img_myisam2.E01/ProgramData/MySQL/MySQL Server 5.5/data/mysql/ndb_binlog_index.frm
Type	File System
Size	8778
File Name Allocation	Allocated
Metadata Allocation	Allocated
Modified	2016-09-08 13:42:33 EAT
Accessed	2016-09-08 16:40:18 EAT
Created	2016-09-08 13:42:33 EAT
Changed	2016-09-08 16:40:18 EAT
MD5	8d38a874f706c1883bd40fb7b0e72be2

Observation: The created time and modified time has remained the same aster UPDATE while change time access time and MD5 hash have changed after update. This shows that the operation had affected the transaction log file.

ii. Re-Do log.

This is physically present in the disk by default as a set of files with reference names of ib_logfile0 and ib_logfile1. They form a data structure is used during crash recovery to correct data written by a transaction that did not complete its execution. Normally operating, the redo log encodes requests forwarded to change InnoDB table data that result from sql statements. The two data structures are as bellow.

Ib_logfile0 before update

Name	/img_myisam1/ProgramData/MySQL/MySQL Server 5.5/data/ib_logfile0
Type	File System
Size	14487760
File Name Allocation	Allocated
Metadata Allocation	Allocated
Modified	2016-09-08 13:42:33 EAT
Accessed	2016-09-08 13:43:33 EAT
Created	2016-09-08 13:42:33 EAT
Changed	2016-09-08 13:4:33 EAT
MD5	5c7fbab3c474a8a07c7c7031bc58e1cb

Ib_logfile0 after update

Name	/img_myisam2/ProgramData/MySQL/MySQL Server 5.5/data/ib_logfile0
Type	File System
Size	14487760
File Name Allocation	Allocated
Metadata Allocation	Allocated
Modified	2016-09-08 13:42:33 EAT
Accessed	2016-09-08 16:40:18 EAT

Created	2016-09-08 13:42:33 EAT
Changed	2016-09-08 16:40:19 EAT
MD5	e2adb6b0586e713ae74292c336d5aeee

Observation: There is evidence of the file being affected by the UPDATE operation based on the transformation of the metadata. The MD5, the changed time accessed times are different in the two images.

Ib_logfile1 before UPDATE

Name	/img_myisam1.E01/ProgramData/MySQL/MySQL Server 5.5/data/ib_logfile1
Type	File System
Size	10485760
File Name Allocation	Allocated
Metadata Allocation	Allocated
Modified	2016-09-08 13:42:33 EAT
Accessed	2016-09-08 13:43:33 EAT
Created	2016-09-08 13:42:33 EAT
Changed	2016-09-08 13:43:33 EAT
MD5	c1c9645dbc14efddc7d8a322d85f26e9

Ib_logfile1 after update

Name	/img_myisam2.E01/ProgramData/MySQL/MySQL Server 5.5/data/ib_logfile1
Type	File System
Size	10485760
File Name Allocation	Allocated
Metadata Allocation	Allocated
Modified	2016-09-08 13:42:33 EAT
Accessed	2016-09-08 13:43:33 EAT
Created	2016-09-08 13:42:33 EAT
Changed	2016-09-08 13:43:33 EAT
MD5	c1c9645dbc14efddc7d8a322d85f26e9
Internal ID	182038

Observation: all the values of metadata have remained the same therefore showing that this file was not affected by the UPDATE operation

iii. General Query log

The General Query log records what Mysqld is doing. The server writes information here when a client connects or disconnects and logs each sql statement received from each client. This file is located in program data/mysql/mysql server 5.5/data/uon-pc.log

General query log before update

Name	/img_myisam.E01/ProgramData/MySQL/MySQL Server 5.5/data/uon-PC.log
Type	File System
Size	144096
File Name Allocation	Allocated
Metadata Allocation	Allocated
Modified	2016-09-08 13:46:42 EAT
Accessed	2016-09-08 13:46:47 EAT
Created	2016-09-08 13:46:41:EAT
Changed	2016-09-08 13:46:47 EAT
MD5	7bf18f35d316d2938a7b00f3617b44dd

General Quer log after UPDATE

Name	/img_myisam2.E01/ProgramData/MySQL/MySQL Server 5.5/data/uon-PC.log
Type	File System
Size	147096
File Name Allocation	Allocated
Metadata Allocation	Allocated
Modified	2016-09-09 13:46:42 EAT
Accessed	2016-09-08 16:40:39 EAT
Created	2016-09-08 13:46:41 EAT
Changed	2016-09-09 16:40:39 EAT
MD5	f78fa732cf5570ad06e82c6e89727a7b

Observation: the MD5, change date as well as access date and size have been changed in the image after the UPDATE. This shows that the UPDATE operation has affected general query log. The change in the size of the file is also theoretically correct because general query log is continuously populated as more and more sql statements are executed.

iv. Error log.

This contains the information of when the server was started and stopped.

Error log before update

Name	/img_myisam1.E01/ProgramData/MySQL/MySQL Server 5.5/data/uon-PC.err
Type	File System
Size	7564
File Name Allocation	Allocated
Metadata Allocation	Allocated
Modified	2016-09-08 13:42:43 EAT
Accessed	2016-09-08 13:42:43 EAT
Created	2016-09-08 13:42:43 EAT
Changed	2016-09-08 13:42:43 EAT
MD5	2d4883c6ef0f62999513388cb3bab8ca
Hash Lookup Results	UNKNOWN
Internal ID	176109

Error log after UPDATE

Name	/img_myisam2.E01/ProgramData/MySQL/MySQL Server 5.5/data/uon-PC.err
Type	File System
Size	10134
File Name Allocation	Allocated
Metadata Allocation	Allocated
Modified	2016-09-08 13:42:43 EAT
Accessed	2016-09-08 16:40:39 EAT
Created	2016-09-08 13:42:43 EAT
Changed	2016-09-00 16:40:39 EAT

MD5	06fd2b14b8c5be23501887375cc31daf

Observation: It can be seen that the size of the file has increased from 7564 to 10134. This is reflective of the behaviour of error log in that its content increases with continued operation of the database hence the file has been affected by UPDATE operation.

v. Undo logs

This is the storage area which holds copies of data modified by a currently running transaction. It helps in ensuring transactional consistence and accuracy. It helps in ensuring that another transaction sees the original data while the current transaction is still underway as a part of consistent read operation. This area is physically part of system table space.

This concept will not be of importance in this research because the product of a rollback segment is .ib data file which has the same content present in the general query log and redo log.

vi. The .frm file:Account Balance.frm

This is the representation of the table on the disk that describes the table format. It bares the same name as the table.

.fr. before update

This is a representation of the structure of the table on the disk.

Name	/img_myisam1.E01/ProgramData/MySQL/MySQL Server 5.5/data/bank_details/account_balances.frm
Type	File System
Size	176002
File Name Allocation	Allocated
Metadata Allocation	Allocated
Modified	2016-09-08 13:46:38 EAT
Accessed	2016-09-08 13:46:38 EAT
Created	2016-09-08 13:46:38:EAT
Changed	2016-09-08 13:46:40 EAT
MD5	fe5e8bc2c1f5029c640a4c0aaee07f8b

.frm after update

Name	/img_myisam2.E01/ProgramData/MySQL/MySQL Server

	5.5/data/bank_details/account_balances.frm
Type	File System
Size	7210
File Name Allocation	Allocated
Metadata Allocation	Allocated
Modified	2016-09-08 13:46:38 EAT
Accessed	2016-09-08 13:46:38 EAT
Created	2016-09-08 13:46:38:EAT
Changed	2016-09-08 13:46:40 EAT
MD5	fe5e8bc2c1f5029c640a4c0aaee07f8b
Hash Lookup Results	UNKNOWN
Internal ID	182034

Observation: Just as with InnoDB image, it can be seen that there were no changes in the metadata of the .frm in the two images. This is theoretically correct because the .frm file does not change if there is no change in the structure of the table.

4.3 Discussion and Interpretation

The aim of the research was to show which files from the list of target files is affected by UPDATE operation in the two installations. The means of telling which file has been affected is by comparing the metadata of the file before the UPDATE operation and the metadata of the file after. The result section shows various metadata but the most important of them are the MAC (Modify, Access and Create) timeand the MD5 hash of the file.

The importance of MAC time in forensics analysis is stressed in Naiqi and Yujie (2008), in that it is crucial in event reconstruction. Whenever there is need for activity reconstruction, then activity timeline can be matched with the MAC time of the files to earmark files to be considered for forensic analysis. Filewhose MAC timeisclose to or matching the timeline of the activity under investigation can then be subjected to forensic analysis .The MD5 hash in the other hand shows the slightest change in data, a change as small as a single bit. So if any operation or application makes the slightest change to a file the MD5hash value will detect this.

By comparing these metadata, it is possible to tell which files were affected in the research.The table below is the summary of how UPDATE operation affected the target files in InnoDB and MyISAM storage engine implementations.

	TARGET FILES						
STORAGE ENGINES	Trans action log	Re-Do log(ib_0)	Re-Do log(ib_1)	Query log	Error log	Master Data File	.frm
InnoDB	✓	✓	✓	✓	✓	*	X
MyISAM	✓	✓	X	✓	✓	*	X

Table2: Summary of the result
Legend:

✓ -File affected

X -File not affected

* File non-existent because of implementation reasons.

From this table, it can be seen that transaction log, redo log(ib_0), redo log(ib_1), query log and Error log are affected in InnoDB implementation while transaction log, redo log(ib_0), query log and Error log are affected in MyISAM implementation. Re-do (ib_01) is however not affected in MyISAM implementation. The .frmfiles of the tables are not affected in both implementations. Master Data File (MDF) is however not analysed because the implementations were stand alone and could not implement the concept of MDF.

4.4. Proposed methodology

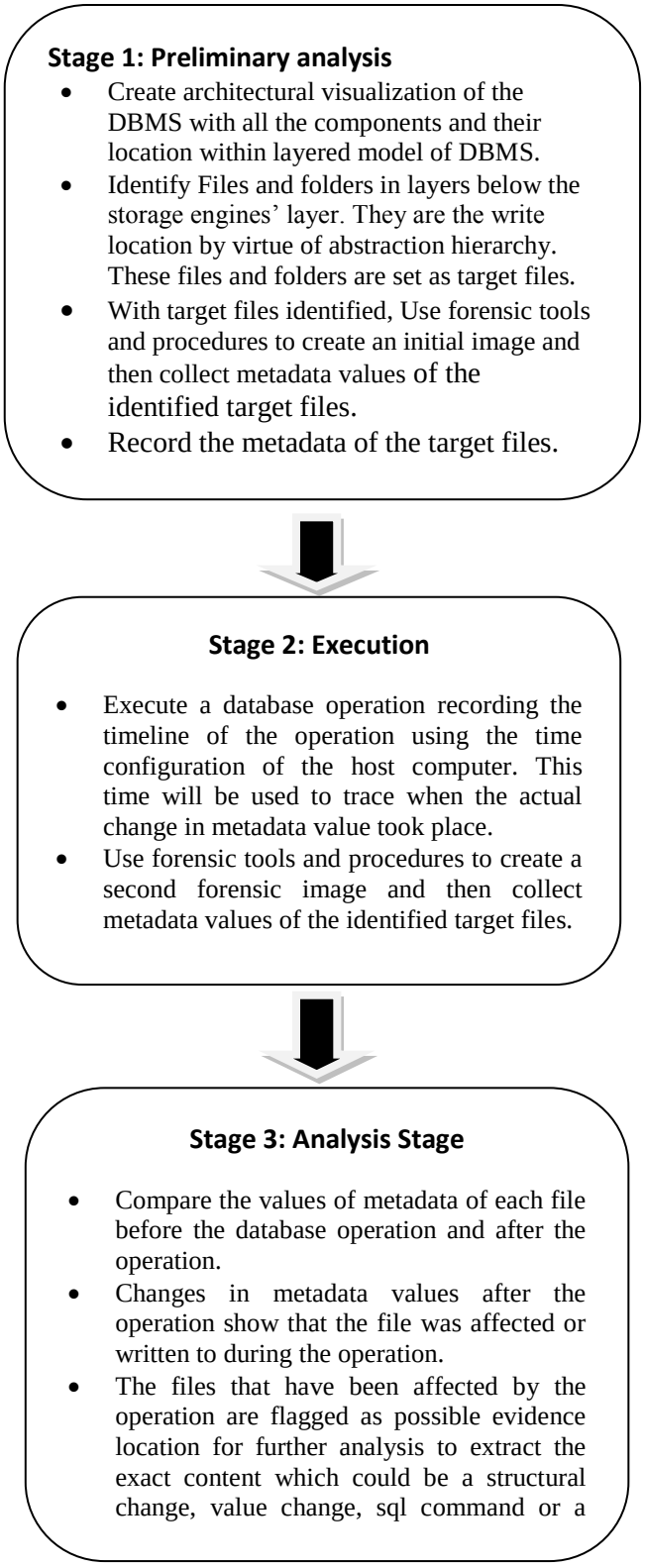
Based on the steps followed in this research, a methodology is outlined as a summary of the steps that were followed in undertaking the experiment. This methodology can be used to test the footprints of any storage engine on the internal files of a DBMS. This will help in flagging and listing files that have been affected by a particular database operation. These file can then be analysed to interpret the actual content to see the nature of change to determine the worth of the evidence.

It is important to remember that the metadata and file information used in this work were as follows:

- **Modified** -when the content of the file most recently changed
- **Accessed** -when the file was most recently opened for reading
- **Created** -the time of creation
- **Changed/Change**-When the file was first created or had the meta data changed
- **MD5** -the MD5 digest of the file
- **Size** -size of the file

In addition to Modified, Access, Created\Change time information of the files, the forensic tool used should give the hash values and size of the file both before and after the operation. In our case, Autopsy gave MD5 hash for the two file instances. The strength of MD5 hash is that it can detect a change in a file as small as a single bit. The size of the file was also observed to have changed for some files after the operation.

The proposed methodology for testing the forensic richness of a storage engine is illustrated in the following diagram.



Fig, 3.Proposed methodology.

CHAPTER FIVE: CONCLUSION AND RECOMENDATION

5.1 Introduction

This chapter evaluates how the objectives of the research were met, assesses the value of the work, gives limitation and finally gives conclusion and recommendation.

5.2 Evaluation of the research objectives

i. **Objectiveone;** *To investigate the forensic implications of using MyISAM or InnoDB storage engines in MySQL database.*

This objective has been met in that study has been carried out on key and comparable internal DBMS files that are written by storage engines during database operation. While only one operation (UPDATE) was used for the experiment, the same methodology can be used with any other operation and any set of files to see the effect.

ii. **Objective two**; *To study how MyISAM and InnoDB storage engines individually generate persistent forensic evidence data in the logs.*

This objective has been achieved by performing the experiment and showing the list of files that have been affected by the operation. These files can then be listed as evidence location and by analysing their content the actual evidence values can be shown.

iii. **Objective three**; *To perform an analysis to determine the number of occurrence of the persistent forensic data in each case and make a comparison.*

This objective has been met by performing analysis of the metadata values and showing the number of files affected in each installation.

5.3 Limitations

While the study has given a methodology of how to list potential forensic evidence locations in a file system, hence showing forensic richness, the study does not however scrutinize the individual evidence location to show its content as either operation

instruction or data values. While this work gives the methodology of how to list evidence location, more work is required is necessary to verify what type of evidence is present in each listed location.

5.4 Conclusions

The following are the conclusions from the study.

- It has been shown that more files are affected in InnoDB than MyISAM implementation hence InnoDB has a higher number of potential forensic evidence locations than MyISAM.
- It is possible to list files affected by an operation in a database system by using a forensic tool to perform a file system analysis
- The methodology shown in this research can be replicated in other file systems to show which files or set of files are affected by an operation
- Because the content of a files are not always the same in different systems, this methodology will help forensic examiners to sieve the files that have potential evidence and then subject them to further analysis therefore eliminating the problem of analysing all files, some of which may not contain any evidence.

REFERENCES

Alexander, K.B., (2014). Database Forensic Analysis. *International Journal*, *2*(3).Apr 13, 2014 - *Database Forensic Analysis Using Log Files*.Mr.Jitendra R Chavan, Prof.HarmeetKaurKhanuja. (Department of Computer Engineering.

Bassil, Y., 2012. A comparative study on the performance of the Top DBMS systems. *arXiv preprint arXiv:1205.2889*.

Bannon, R., Chin, A., Kassam, F., Roszko, A. and Holt, R., (2002).*Mysql conceptual architecture*.Technical report, University of Waterloo.

Bricki, N. and Green, J., (2007). A guide to using qualitative research methodology.

Flores, D.A., Angelopoulou, O. and Self, R.J., (2012).September.Combining Digital Forensic Practices and Database Analysis as an Anti-Money Laundering Strategy for Financial Institutions.In*2012 Third International Conference on Emerging Intelligent Data and Web Technologies* (pp. 218-224).IEEE.

Fruhwirt, P., Huber, M., Mulazzani, M. and Weippl, E.R., (2010).April.Innodb database forensics. In *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on* (pp. 1028-1036). IEEE.

Gilfillan, I., (2003). *Mastering MySQL 4*.Sybex.

Grebhahn, A., Schäler, M. and Köppen, V., (2013). Secure Deletion: Towards Tailor-Made Privacy in Database Systems. In *BTW Workshops* (pp. 99-113).

Hancock, B., Ockleford, E. and Windridge, K., (1998). *An introduction to qualitative research*. Nottingham: Trent focus group.

Holck, J., Mahnke, V. and Zicari, R., (2008).Winning through incremental innovation: The case of MySQL AB. IRIS.

Johnson, B., (2001). Toward a new classification of nonexperimental quantitative research.*Educational Researcher*, *30*(2), pp.3-13.

Khanuja, H.K. and Adane, D.D., (2011). Database security threats and challenges in database forensic: A survey. In *Proceedings of 2011 International Conference on Advancements in Information Technology (AIT 2011), available at http://www.ipcsit.com/vol20/33-ICAIT2011-A4072. pdf.*

Khanuja, H.K. and Adane, D.S., (2012). A framework for database forensic analysis.*Computer Science & Engineering*, *2*(3), p.27.

Martini, B., Do, Q. and Choo, K.K.R., 2015. Conceptual evidence collection and analysis methodology for Android devices.*arXiv preprint arXiv:1506.05527*.

Naiqi, L., Zhongshan, W. and Yujie, H., 2008, August. Computer Forensics Research and Implementation Based on NTFS File System. In *2008 ISECS InternationalColloquium on Computing, Communication, Control, and Management*(Vol. 1, pp. 519-523). IEEE.

Oracle, M., 5.6 Reference Manual, 2014.

Pozniak-Koszalka, I. and Helwich, M., (2005).Experimentation System for Evaluating MySQL Database Management System Efficiency. In *Databases and Applications* (pp. 94-98).

Robbins, R.J., (1994). Database Fundamentals.*Johns Hopkins University, rrobbins@ gdb.org*.

Rodr´ıguez A., (2004).Brief Introduction to Database Concepts.*Summer School - Castell´on.*

Stahlberg, P., Miklau, G. and Levine, B.N., (2007). June. Threats to privacy in the forensic analysis of database systems.In*Proceedings of the 2007 ACM SIGMOD international conference on Management of data* (pp. 91-102). ACM

Tocci, G., 2013. A Comparison of Leading Database Storage Engines in Support of Online Analytical Processing in an Open Source Environment.

Vaswani, V., (2003).*MySQL: The complete reference*. McGraw Hill Professional.

Weippl, E., (2010). Database Forensics.In*Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications (AINA)*.

http://db-engines.com/en/ranking