A FORENSIC EXAMINATION OF DATABASE SLACK

by

Joseph Balazs

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the Degree of

Master of Science

Department of Computer and Information Technology

West Lafayette, Indiana

August 2021

THE PURDUE UNIVERSITY GRADUATE SCHOOL

STATEMENT OF COMMITTEE APPROVAL

Dr. Marcus K. Rogers, Chair

Department of Computer and Information Technology

Dr. John A. Springer

Department of Computer and Information Technology

Dr. Dawn D. Laux

Department of Computer and Information Technology

Approved by:

Dr. John A. Springer

Chair of the Graduate Education Committee

For Mom and Dad.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CentOS	community enterprise operating system
DBMS	database management system
DFRWS	Digital Forensic Research Workshop
IBM	International Business Machines
IC3	Internet Crime Complaint Center
NIST	National Institute of Standards and Technology
RAM	random access memory
SQL	structured query language
SSD	solid state drive
SWGDE	Scientific Working Group on Digital Evidence
TPC	Transaction Processing Performance Council

GLOSSARY

Database: "...a collection of related tables and other structures," (Kroenke & Auer, 2010, p. 13)

Database Management System: "...a computer program used to create, process, and administer the database," (Kroenke & Auer, 2010, p. 13)

Database Slack: "...the data is located in a file in use by the database system," (Stahlberg, Miklau, & Levine, 2007, p. 93)

Digital Evidence: "Information of probative value that is stored or transmitted in binary form," (Scientific Working Group on Digital Evidence, 2016, p. 7)

Digital Forensic Science: "The use of scientifically derived and proven methods toward the preservation, collection, validation, identification, analysis, interpretation, documentation and presentation of digital evidence derived from digital sources for the purpose of facilitating or furthering the reconstruction of events found to be criminal, or helping to anticipate unauthorized actions shown to be disruptive to planned operations," (Palmer, 2001, p. 16)

File Slack: "The data between the logical end of a file and the end of the last storage unit for that file," (Scientific Working Group on Digital Evidence, 2016, p. 8)

Forensic: "The use or application of scientific knowledge to a point of law, especially as it applies to the investigation of crime," (Scientific Working Group on Digital Evidence, 2016, p. 8)

Free Space: "Data storage areas available for use by the computer. The area may already contain previously stored information. Also referred to as Unallocated Space," (Scientific Working Group on Digital Evidence, 2016, p. 9)

Relational Model: "The relational model uses a collection of tables to represent both data and the relationships among those data...The relational model is a combination of three components, such as Structural, Integrity, and Manipulative parts," (Sumathi & Esakkirajan, 2007, p. 67)

Validation: "The process of performing a set of experiments, which establishes the efficacy and reliability of a tool, technique or procedure or modification thereof," (Scientific Working Group on Digital Evidence, 2016, p. 18)

Validation Testing: “An evaluation to determine if a tool, technique or procedure functions correctly and as intended,” (Scientific Working Group on Digital Evidence, 2016, p. 18)

Verification: “Confirmation that a tool, technique or procedure performs as expected,” (Scientific Working Group on Digital Evidence, 2016, p. 18)

Write Block/Write Protect: “Hardware and/or software methods of preventing modification of media content,” (Scientific Working Group on Digital Evidence, 2016, p. 19)

ABSTRACT

This research includes an examination and analysis of the phenomenon of database slack. Database forensics is an underexplored subfield of Digital Forensics, and the lack of research is becoming more important with every breach and theft of data. A small amount of research exists in the literature regarding database slack. This exploratory work examined what partial records of forensic significance can be found in database slack. A series of experiments performed update and delete transactions upon data in a PostgreSQL database, which created database slack. Patterns of hexadecimal indicators for database slack in the file system were found and analyzed. Despite limitations in the experiments, the results indicated that partial records of forensic significance are found in database slack. Significantly, partial records found in database slack may aid a forensic investigation of a database breach. The details of the hexadecimal patterns of the database slack fill in gaps in the literature, the impact of log findings on an investigation was shown, and complexity aspects back up existing parts of database forensics research. This research helped to lessen the dearth of work in the area of database forensics as well as database slack.

CHAPTER 1. INTRODUCTION

This chapter gives an overview of the research. The scope and significance of the research question will be introduced. Bounds of the study will also be covered. Definitions of key terms are provided for context and clarity.

1.1 Background

Digital media is pervasive. The business, education, government, and consumer realms all store data through digital means. As data continues to accumulate daily, a criminal element has emerged to take advantage of digital media's ease of access and the ability to make a profit. According to the Internet Crime Complaint Center (IC3), in 2015, nearly 300,000 victims lost a grand total of more than one billion dollars (Federal Bureau of Investigation, 2015). Businesses have been hit even harder. Recent research by IBM puts the average cost of a breach around four million dollars (Hackett, 2016). Others forms of loss include identity theft, loss of intellectual property, and network accounts and credentials. Breaches were a massive problem in 2016. According to a report from IBM, more than 4 billion records were leaked in 2016 (IBM Security, 2017; Seals, 2017).

Databases are not a new concept; they have been around since the 1960s (Fowler, 2008; Matthew & Stones, 2005). One of the most important and prevalent forms is that of the relational database model, which is based upon relational algebra and discrete mathematics. The concept was established by a researcher at IBM, E. F. Codd (Codd, 1970). The relational model is still in use today, and it is quite prevalent for data storage. As data grows daily, the importance of databases increases equally. This importance is reinforced with every breach and theft of consumer data.

Digital Forensics is an emerging field of Forensic Science. As the overall field reaches an increasing level of maturity, many of its subfields have not. Database forensics is one such subfield. With the entrenched position of the relational model in the real world, it is imperative to further explore the forensics of databases. Crime involving databases will only increase as time moves forward.

1.2 Scope

Some concepts and techniques are well established in the domain of Digital Forensics. One such idea is file slack. This represents the empty space from the physical end of a file to the logical end within a data block (Carrier, 2005). In this allocated space, data can be hidden.

The notion of file slack can also be applied to database forensics. It is known as database slack. Database slack refers to any object or artifact in use by a database management system that has not been released to the file system (Stahlberg et al., 2007). It is similar to file slack; areas within the file system that are allocated to a database management system can have items no longer in use yet still remain in a persistent state (Stahlberg et al., 2007). This may include transactions, logs, and data.

More research needs to be done regarding the items of forensic interest residing in database slack. To date, only one set of authors, Stahlberg et al., has done research involving the topic (2007).

1.3 Research Question and Hypothesis

The research question is what items, known as partial records, of forensic significance can be found in database slack?

Hypotheses consist of the following:

H_0: No items of forensic significance will be found in database slack.

H_α: Partial records of forensic significance will be found in database slack.

In terms of the research question, partial records can refer to an array of objects, including but not limited to tables, columns, rows, fields, data, transactions, logs, file fragments, and artifacts. Forensic significance can be used in terms of any piece of digital evidence that would be useful to an investigation.

1.4 Significance

One reason this research project is important is that database forensics is an area of Digital Forensics that has not been thoroughly researched. Several voices in the field have pronounced the need for more research on the topic. The Digital Forensic Research Workshop (DFRWS) has had calls for papers on this topic for the last ten years (2010; 2011; 2012; 2013; 2014; 2015; 2016; 2017; 2018; 2019; 2020). One of the main authors of the field, Martin Olivier, has decried the lack of research as well (Olivier, 2009). In an article on the state of database forensics, Lawrence Suffern succinctly offered this thought, "Much of the research discusses the need for more research" (Suffern, 2010, p. 70).

Another justification for the value of this study is its relevance to current events in the field. There have been numerous breaches in the news, and the target of the breaches is the stored private data of individuals. The breach of Anthem, one of the largest health insurers, is one example. More than 80 million information files were stolen (Khosla, 2015). The Office of Personnel Management breach is another case. The records of more than 21 million people were pilfered (OPM, n.d.). On February 4, 2016, the University of Central Florida was breached in yet another illustration. The Social Security Numbers of 63,000 students were obtained (Barth, 2016). Recently, two large breaches have victimized Yahoo!, an online technology enterprise. The second breach involved a billion accounts, making it one of the largest breaches in history (Perlroth, 2016). Breaches cost users private protected information, which can lead to identity theft, ruined credit, and loss of finances. Businesses feel the cost through financial, security, and reputation losses.

1.5 Assumptions

This research involves the use of a computer running specific software. These assumptions have been made for this study:

- There will be no hardware or software failures on the testing machine during this research.
- Hardware write blockers function properly.

- There will be no power outages or natural disasters while the research is taking place.

1.6 Limitations

The limitations of the research are:

- Database slack research will be limited to the PostgreSQL database software management system.
- The database will be limited to one machine, which does not approximate the client/server model of a real life scenario.
- The research will be performed on a system running the Linux distribution CentOS.
- Beyond any available licensed software in the Purdue Cyber Forensics Lab, cost will limit analysis tools to free and/or open source implementations.
- Time will be a limitation; the research will not go on ad infinitum.
- File system will be limited to LVM2 and Ext4.
- RAM will be limited to 16 GB.
- Hard drives will be limited to magnetic hard drives.

1.7 Delimitations

The delimitations of this study include:

- Testing will only be performed on a Linux system; Windows, Macintosh, and other operating systems will not be considered.
- One relational database management system will be used; other database management systems such as MySQL, Oracle, SQL Server, InnoDB, etc., will not be considered.
- Live analysis will not be performed during the course of this research.

- RAM configurations other than 16GB will not be used.

1.8 Summary

This chapter has provided an overview of the research. It has covered the scope and significance of the research question while also supplying bounds for the study. Definitions of key terms were also offered.

CHAPTER 2. REVIEW OF LITERATURE

This chapter reviews literature relevant to database forensics, database slack, and file slack.

2.1 Database Forensics Background and Prior Work

Database forensics is a newer branch of the field of Digital Forensics. A portion of the existing work decries the lack of research on the topic and pleads for more examination to be done on database forensics. Two early overview of the field pieces did this: Olivier's *On metadata context in database forensics* paper and Suffern's piece *A study of current trends in database forensics* (Olivier, 2009; Suffern, 2010). Both of those articles reviewed the existing literature at the time, enumerated the accomplishments, and concluded that much more work needs to be done (Olivier, 2009; Suffern, 2010). Olivier revisited the topic recently in a follow up piece, *The state of database forensic research*, that tries to pinpoint reasons for the lack of research (W. K. Hauger & Olivier, 2015). While the overall amount of research has increased since the 2009 piece, the authors were unable to find a definitive reason or explanation for the dearth of work on the subject (W. K. Hauger & Olivier, 2015).

There have been calls for action and requests for more work in the subfield of database forensics from the larger domain of Digital Forensics. The Digital Forensics Research Workshop (DFRWS) has made calls for papers on database forensics every year since 2010 (2010; 2011; 2012; 2013; 2014; 2015; 2016; 2017; 2018; 2019; 2020). They finally received and accepted pieces in 2015, 2016, 2017, and 2019 (*DFRWS USA 2015 Call for Papers*, 2015; *DFRWS USA 2016 Call for Papers*, 2016; *DFRWS USA 2017*, 2017; *DFRWS USA 2019*, 2019; Kim, Park, & Lee, 2016; Meng & Baier, 2019; Wagner et al., 2017; Wagner, Rasin, & Grier, 2015, 2016; Wagner, Rasin, Heart, Jacob, & Grier, 2019). Beebe identified database forensics research as a significant area of need in her state of the field of Digital Forensics paper titled, *Digital forensic research: The good, the bad and the unaddressed* (Beebe, 2009). Casey classified database forensics as a new area of Digital Forensics and also as an example of a need to adapt processes and practices (Casey, 2006).

Olivier has published the most literature on the topic of database forensics. Olivier's work is mostly expository and theoretical. The topics have included data hiding methods, log settings, schema, and mathematical proofs (Adedayo & Olivier, 2014, 2013, 2015; Beyers, Olivier, & Hancke, 2011; Fasan & Olivier, 2012; W. Hauger & Olivier, 2015; Pieterse & Olivier, 2012). While Olivier has published a great deal of work, little can be used for this research.

Frühwirt, Huber, et al. published a piece entitled, *InnoDB database forensics* (2010). Its focus is a MySQL database with an InnoDB storage engine (Frühwirt et al., 2010). After an audit of MySQL source code, the authors have decoded the hex code representations for key locations of important data for forensic reconstruction of a database incident (Frühwirt et al., 2010). This is the kind of piece that is missing from the literature for PostgreSQL; something detailing the hex code for the DBMS. A later work by this set of authors, *InnoDB database forensics: Enhanced reconstruction of data manipulation queries from redo logs*, focuses on the redo log and the forensic reconstruction work that can be done from it (Frühwirt, Kieseberg, Schrittwieser, Huber, & Weippl, 2013).

Kieseberg, Schrittwieser, et al. have written a piece detailing a new approach for use in database forensic analysis (Kieseberg, Schrittwieser, Mulazzani, Huber, & Weippl, 2011). It focuses on the analysis of binary trees as an underlying data structure of the code used in databases (Kieseberg et al., 2011). While this presents a different method, it spends much of its time in mathematical proofs and theoretical concepts which won't be applicable to this research (Kieseberg et al., 2011).

Several authors have produced work relating to the DBMS Oracle; these are the specific type of pieces that are missing from the literature for PostgreSQL that would be useful for this research. David Litchfield has published several books on database security. He was also set to publish a book on Oracle forensics, but did not finish it due to legal concerns. Litchfield has also published a blog and has written a series of articles pertaining to Oracle forensics (Olivier, 2009). Paul Wright has written a book on Oracle forensics, but this is more of a guidebook for system administrators than an in depth look at forensic processes and methodologies (Olivier, 2009; Wright, 2008). Wright has also provided some independent work on the topic through his blog (Olivier, 2009; Wright, 2007).

Kevvie Fowler has published a book, a presentation at Black Hat, and a real world type scenario of an investigation for Microsoft's line of database management software, SQL Server (Fowler, 2007a, 2007b, 2008). His book also features a custom set of tools he uses for investigations (Fowler, 2008). He has also published several pieces on his blog as well. His work is the closest to any type of investigation model for database forensics (Suffern, 2010). These kinds of pieces are needed in the literature for PostrgreSQL.

2.2 Research on Database Slack

Stahlberg, Miklau, et al. published their first research in 2007; this piece forms the basis of this thesis (2007). *Threats to privacy in the forensic analysis of database systems* has several important attributes and concepts. The most important idea is the notion of database slack (Stahlberg et al., 2007). Similar to file slack or slack space, this concept refers to pieces of data held by a database management system, but not yet released back to the file system (Stahlberg et al., 2007). The importance is that valuable pieces or fragments of data may still reside in the system, even though they may have been marked as expired and altered or removed from a table, log, or index (Stahlberg et al., 2007). Expired data will remain in this state until a database management system process alters it, usually in the form of a "vacuum" command as defined by the authors (Stahlberg et al., 2007).

To explore the concept of expired data retention in database slack, the authors designed and executed a series of experiments (Stahlberg et al., 2007). For five different database management systems (PostgreSQL, MySQL with MyISAM, MySQL with InnoDB, SQLite, and DB2), sets of records were loaded into them (Stahlberg et al., 2007). Randomized operations were run, and after 100 operations, items in database slack were measured (Stahlberg et al., 2007). This was repeated until 50,000 operations occurred, and then database slack was measured again at the end of the experiment (Stahlberg et al., 2007). Results varied by database management system type; PostgreSQL puts all records into database slack, MySQL with MyISAM does not put any records in database slack, and the remaining systems had results in between the two extremes (Stahlberg et al., 2007).

At a different point in their paper, the authors enumerate a set of criteria to make sensitive data retention as small as possible (Stahlberg et al., 2007). They subsequently alter the configuration of MySQL with InnoDB to meet their criteria and rerun the experiment (Stahlberg et al., 2007). This reduced the amount of data in database slack to almost zero (Stahlberg et al., 2007).

A follow-up paper, *Securing history: Privacy and accountability in database systems*, expands some of the analysis of the previous paper, but the focus is on privacy and accountability (Miklau, Levine, & Stahlberg, 2007). Some useful information is presented on the duration of data in database slack and the process of secure deletion (Miklau et al., 2007).

2.3 How This Research is Different

The purpose of this thesis is to build upon the work of Stahlberg, Miklau, et al (2007). The goal is to be observing specific items in database slack, enumerating them, and analyzing the results. This is important because there could be potential inculpatory and exculpatory evidence contained in the database slack.

One area that needed further detail from the original author's work was the specifics involving database slack. How were they able to determine where pieces of database slack were? What were the specifics of what constitutes database slack, i.e. the markers within the hex? The authors indicate they made a custom tool to accomplish this task; how did that perform its goal? The lack of information and details was absent from the rest of the literature as well; there was no existing documentation anywhere on how to determine the hex location of a record, let alone a deleted or updated item within a database.

Since years have passed since their experiments occurred, it is useful to determine if current versions of database management systems have improved, worsened, or remained the same in regards to the amount of database slack produced and retained. Another important consideration is that none of the subsequent research by any author in database forensics has addressed or contained anything pertaining to database slack.

2.4 File Slack/Slack Space Background

In computers, the smallest unit of data is a bit (Elrick, 2014). Eight bits compromise a byte (Elrick, 2014). Hard disk drives store bytes in groups called sectors (Elrick, 2014). Files are stored within the allocated space of a file system, which is in turn managed by an operating system. Only one file can be stored per allocation unit. Because a file might not take up an entire unit of allocation, a phenomenon known as file slack or slack space occurs. Slack space is the padding at the end of the file to the end of the allocated data block (UNIX based systems) or cluster (Windows based systems) (Kruse, 2002; Prosise, Mandia, & Pepe, 2003). The "garbage data" that makes up file slack comes from several different places: unallocated space, overwritten files, items in RAM (also known as RAM slack), and deleted files. An operating system will not read the slack data from the file system; special tools are needed to view file slack (Kruse, 2002).

Many sources address file slack and slack space, and they go in a reverse chronological order: Carrier's book from 2005; Mandia, Prosise, and Pepe's book and Rogers' article, both from 2003; Kruse's book from 2002; and lastly, Casey's book from 2000 (Carrier, 2005; Casey, 2000; Kruse, 2002; Prosise et al., 2003; Rogers, 2003). File slack and slack space seems to have reached a point of "general acceptance" at some point over the years. All of the aforementioned sources do not cite a source when discussing slack space; all of the pieces were written by credible, established names within the field of Digital Forensics. Efforts to find a history of the subject of file slack/slack space or a person credited with the creation or discovery of the concept proved fruitless. An article titled *Computer forensics today*, an in-depth overview of the field of Digital Forensics, was published in the year 2000 (Kuchta, 2000). It mentions slack space as well, but does not cite a source for it, either (Kuchta, 2000). It seems the term had found an acceptance level as early as the year 2000.

Slack data could be potential evidence. Critical pieces of data or information that could be inculpatory or exculpatory in a legal case may reside in slack space. Slack space data may have been put there by system processes, or it could be hidden there by a malicious user. While special tools have been used to extract and analyze file slack and slack space for Digital Forensics, it is crucial to thoroughly research this area as it relates to database forensics.

2.5 Summary

This chapter presented a review of the literature relevant to database forensics and slack space. The next chapter covers the methodology used for the study.

CHAPTER 3. METHODOLOGY

3.1 Study Design

An existing methodology for research in Digital Forensics, "General Test Methodology for Computer Forensic Tools," provided by the National Institute of Standards and Technology (NIST) was used in this research (National Institute of Standards and Technology, 2001). It was modified by Leshney for exploratory research for his investigation (Leshney, 2008). Leshney's modified methodology consists of five parts: "establish categories of forensic requirements, identify test assertions/variables, develop test cases, develop testing procedures and methods, and report results" (Leshney, 2008, p. 33). This research followed Leshney's methodology.

The first part of the methodology, establish categories of forensic requirements, was a novel approach. Leshney based his research on three categories defined by Burchett (Leshney, 2008). These were not ideal fits for this research, as they are file and directory related. This research pertains to databases; they consist of more than files, directories, and logs. The log category was of use, however. An alternative was to use the leeway given by SWGDE to come up with a unique category of needed research. Based on that freedom from SWGDE, several new database related categories were created. They consisted of a row, a column, and a transaction. A row results in a record, or a full entry in a database. A column represents a single value or piece of data from a database. A transaction results from a series of SQL statements that perform actions and modify the database.

The second part of the methodology was to identify test assertions/variables. This would be defining an action or state that a user could accomplish on a database. This research was defined by three states based on the possible actions on a database: create/insert, update, delete. Basically, those states are the write functions available within a database.

To develop test cases, a set of data was created that populated a database. The TPC Benchmark guidelines have been consulted as a model, and nine tables of varying complexity were constructed to make the database (Transaction Processing Performance Council (TPC), n.d.). The nine tables that were created were Customer, District, History, Item, NewOrder,

OrderDetails, OrderLine, Stock, and Warehouse (Transaction Processing Performance Council (TPC), n.d.). Minor modifications were made to the schema due to the reserved keyword ORDER; thus, Order was changed to OrderDetails (Transaction Processing Performance Council (TPC), n.d.). The schema is contained in Appendix A, and an overview of the foreign keys is featured in Appendix B. Test data for the database was suggested to be 10,000 records per table, which seemed like overkill (Transaction Processing Performance Council (TPC), n.d.). Half of that suggestion, 5,000 records per table, was adequate. The website Mockaroo was used to create the test data (*Mockaroo - Random Data Generator and API Mocking Tool | JSON / CSV / SQL / Excel*, n.d.).

Testing procedures and methods followed the best practices of Digital Forensics. They included the use of write blockers, validation software, and forensic analysis software. A series of transactions took place on the database. At a pre-determined interval, 500 transactions, items in database slack were measured. Measurements were done by forensic software and a hex editor. The process was then repeated.

The final part of the methodology was to report results. After all measurements had been done, an analysis of the results took place. At that point, the analysis was reported.

3.2 Conditions/Environment

Testing took place in a controlled environment. The desktop machine was in a lab, and it was isolated. To replicate a real life scenario of a typical database workload, a series of transactions (create/insert, update, delete) on the database took place. The transactions were accomplished through a programming script. After 1,500 transactions took place, an image was taken of the hard drive. This process was repeated four times.

3.3 Research Question and Hypotheses

The research question is what items, known as partial records, of forensic significance can be found in database slack? Hypotheses consist of the following:

H_0: No items of forensic significance will be found in database slack.

H_α: Partial records of forensic significance will be found in database slack.

3.4 Procedures

A desktop machine was used to host a database management system. The hard drive was wiped to ensure any previous remnants of data did not interfere with the experiments. The Linux distribution CentOS was used as the operating system for the machine, and PostgreSQL was the database management system (*The CentOS Project*, n.d.; *PostgreSQL*, 2021). CentOS was chosen as the operating system as it is open source, free of cost, reliable, stable, and the author has familiarity and experience with it (*The CentOS Project*, n.d.). PostgreSQL was chosen for the database management system for several reasons (*PostgreSQL*, 2021). The first was that it performed the worst of the database management systems in Stahlberg et al.'s work; it placed everything into database slack (Stahlberg et al., 2007). The second reason was that it is open source and available free of charge. The open source characteristic was especially important as forensic tools may not recognize the database files, and analysis may have to be done by reviewing hex code. Reviewing the source code for PostgreSQL may be the only possible way to determine how pieces of data are stored and where they will be located within a system's layout. A diagram of the basic process is contained in 3.1.

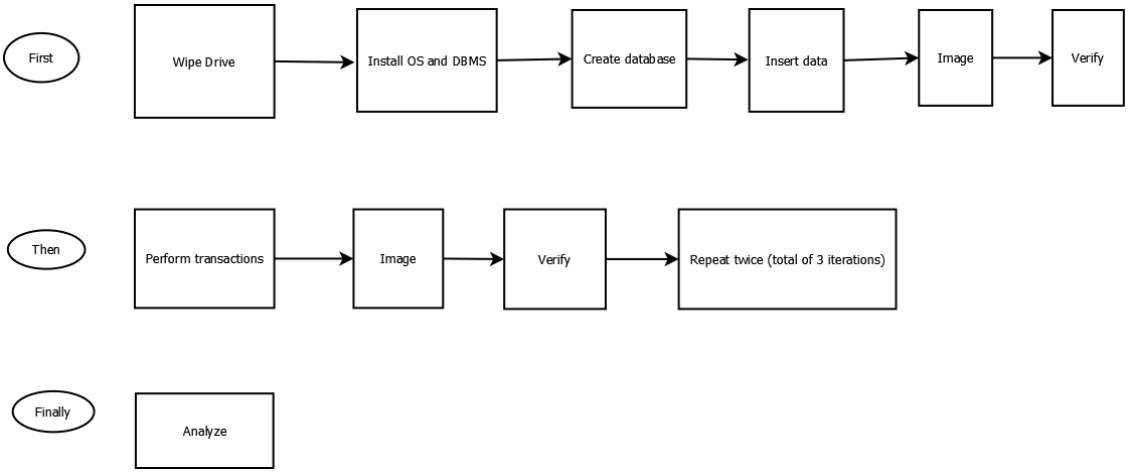

Figure 3.1. Flow Chart of Process

Forensic acquisition tools and best practices were used to extract and analyze the database. The hard drive of the desktop machine was imaged using hardware write blockers and the forensic imaging software FTK Imager (*FTK Imager*, n.d.). The acquired image was saved to a storage drive, and analysis occurred from this forensic copy. Analysis was performed using the hex editor contained in FTK Imager (*FTK Imager*, n.d.).

After wiping a hard drive and verifying the contents were empty, a machine was loaded with the CentOS operating system (*The CentOS Project*, n.d.). PostgreSQL was installed and configured as the database management system (*PostgreSQL*, 2021). A database was created and filled with sample data. At this point, the database had transactions performed on it by a script. After the script was finished, the hard drive was pulled and imaged with hardware write blockers to a destination storage drive for the forensic image. From the forensic image, analysis took place with forensic software, specifically a built-in hex editor, to determine the amount and type of objects placed into database slack. The results were measured and analyzed. The entire process was to be repeated for four more iterations, giving a total of five sets of experimental data to analyze.

3.5 Data Sources

A database was created and populated with data. This provided a source of data to test as items go into database slack. A database of nine tables was created. Test data, 5,000 records per table, were created. Test data was populated via the website Mockaroo (*Mockaroo - Random Data Generator and API Mocking Tool | JSON / CSV / SQL / Excel*, n.d.). Data sets are located in 'Dry Run Testing.zip.'

3.6 Measure for Success

The primary measure of success for this research was locating and identifying items contained in database slack. Due to the exploratory nature of this research, this primary measurement of success is binary in nature; pieces of evidence will either be located or they will fail to be located.

3.7 Threats to Validity

The primary threats to validity in this research are internal threats. Failure to follow procedures will result in a loss of validity. As this research is a forensic analysis, digital forensic practices will be followed; forensic software, forensic imaging, verification, and analysis from forensic copies were all used in the research. If they were not adhered to, this would also pose a threat to validity. For example, if the imaging process in any way results in a change to the original data, then the research will be considered forensically invalid.

3.8 Summary

This chapter discussed the methodology to be followed for this research. It also examined the procedures, conditions, data sources, measures of success, and threats to validity.

CHAPTER 4. RESULTS

This chapter contains the results of the research. Three phases of testing (Phase 1, Phase 2, and a Dry Run of Full Scale Experiment) will give details of the work. Analysis will also be expanded.

4.1 Method to Locate Database Slack

An approach based on carving was considered. Carving uses file signatures to locate deleted files. After consulting literature for file signatures, a signature for Postgres files could not be located. Additionally, carving would not work to locate a specific value within a file anyways.

After some brainstorming, a testing method using trial and error to locate the hex for the database slack was decided upon. This would use comparison between original data and subsequently updated and deleted data. As this was the simplest solution to the problem, an approach based around Occam's Razor would work best due to the complete lack of details of what specifically constituted database slack.

PostgreSQL 10 was used (*PostgreSQL*, 2021). All database work was done via the command line; no front end GUI was ever used. Code was run through the use of scripts passed via the \I command to the file. FTK Imager version 4.1.1.1 was used for imaging for Phase 1 Testing; version 4.2.0.13 was used for imaging for all other phases (*FTK Imager*, n.d.). FTK Imager version 4.2.0.13 was used for analysis (*FTK Imager*, n.d.).

4.2 Testing Phase 1

Small scale testing was done. A test database consisting of a single table with three columns, idNumber, FirstName, and LastName, was created. Five rows of data were inserted into the table. After the successful creation of the database and insertion of data, a forensic image was made as a control for comparison. The database and data were successfully located within the file system structure. The file was located at /var/lib/pgsql/10/data/base/16384/16399. The 16384 was

the directory that held the database. The 16399 file contained the data for the table. The location within the file system, drilled down from root, is illustrated in 4.1.

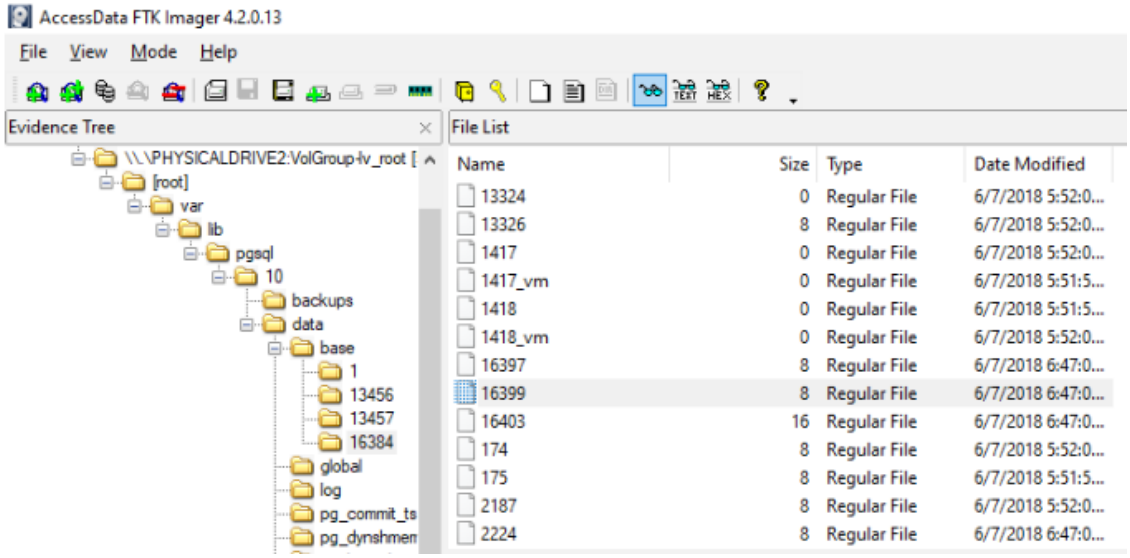

Figure 4.1. File Location in File System

All data was located in the data file. It begins at the very end of the file, and works it way up from the bottom. All five sets of first names and last names appeared in order from the end of the file. This is shown in figure 4.2.

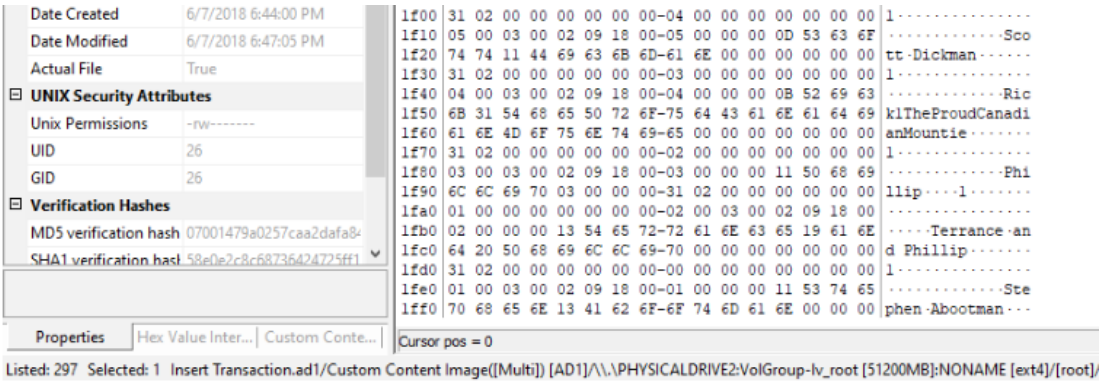

Figure 4.2. Data Location in Data File

Next, an update transaction was performed that changed two pieces of data, and a forensic image was then taken of the hard drive. Analysis was performed using the hex interpreter in FTK

Table 4.1. *Hexadecimal Indicators for Phase 1*

Record Status	Hex String
Live	xx 03 00 02 09 18 00
Updated Modified	xx 03 80 02 28 18 00
Updated Expired	xx 03 40 02 01 18 00
Deleted	xx 03 20 02 01 18 00

Imager 4.2.0.13, and the database slack was located by comparing the hex values of the new pieces of data versus the old pieces of data (*FTK Imager*, n.d.).

Updates resulted in two separate yet related locations for data; an inactive copy of the original data and an active copy of the new data. This also resulted in a new ID number for the row after modification. Records that did not receive an update were symbolized by a hexadecimal string consisting of xx 03 00 02 09 18 00, where the xx was variable to each individual record. By comparing those original records to the ones that received an update, there were differences in the hex strings. A modified record was indicated by a hex string of xx 03 80 02 28 18 00. The copy of the inactive original record was now specified by a string of xx 03 40 02 01 18 00. The xx's were variable to the record and the ID number. Hence, the database slack was present after an update.

Finally, a delete transaction was performed, deleting one piece of data, and another forensic image was created. Comparisons were made between the three images to try and determine the location and indication of updated and deleted items within the hexadecimal code of the file system. The live records, the updated modified records and updated expired records, and the deleted record were all present. The deleted record was now symbolized by a hex string of xx 03 20 02 01 18 00. Thus, the deleted record was also present in database slack.

A summary of the different hex values is illustrated in table 4.1. The live records and updated modified records are accessible in the database. The updated expired records and deleted record are not present in the database, yet they remain in the file system and database slack.

4.3 Testing Phase 2

The next phase of testing attempted to solve some problems and achieve several goals: randomization of numbers and text within the data and ramping up to a larger amount of data. Testing phase two incorporated a database containing a single table with columns similar to the phase one table: NameID, FirstName, and LastName. A larger amount of records, fifty rows, was used this time. Fifty first and last names were created with the online data tool, Mockaroo (*Mockaroo - Random Data Generator and API Mocking Tool | JSON / CSV / SQL / Excel*, n.d.). Data was inserted via a script containing an insert transaction. A similar process as Phase 1 was followed with imaging and analysis. The data file was located at /var/lib/pgsql/10/data/base/16405/16458. All data was present, and the original contents of the table are illustrated in Figure 4.3.

Code was written in PLPGSQL, a built-in high level language for PostgreSQL, that would randomize the rows selected for transactions as well as writing random text on the fly for changing names in the data (Lipiński, n.d., 2011; *PL/pgSQL - SQL Procedural Language*, 2021). The functions are contained in 'Phase Two Testing.zip.' Fifty transactions were then run via script to perform updates followed by a requisite forensic image to perform analysis. Updated data was present in the table, and the new contents of the table after the modifications are shown in Figure 4.4.

Examination via FTK Imager's hex editor revealed similar patterns in the hex as a result of the updates, but the exact values in the hex were different (*FTK Imager*, n.d.). Live records were still represented by the hex string xx 03 00 02 09 18 00. However, the updated records had different indicators within the hex string. While updated expired records previously had a string of xx 03 40 02 01 18 00, they were now symbolized by the string of xx 03 40 02 05 18 00. Similarly, updated modified records were earlier represented by the hex string of xx 03 80 02 28 18 00, and here they were indicated by the values xx 03 80 02 29 18 00. A summary of the phase two hex values is contained in table 4.2.

Phase two testing again produced database slack. However, the patterns of the hex string had changed. To this point, it had been relatively easy to locate the data and database slack

nameid	firstname	lastname
1	Charles	Prawle
2	Murray	Erbain
3	Rafaello	Baltzar
4	Vinnie	Sanderson
5	Lisbeth	Fellon
6	Tiler	Clinton
7	Anatole	Catling
8	Lorant	Boothroyd
9	Malissia	Gerardin
10	Cherida	Morrott
11	Breena	Bernucci
12	Mureil	Greated
13	Foss	Elliot
14	Erinn	Dalziel
15	Barny	Van den Velde
16	Werner	Doulton
17	Milly	Woodyer
18	Araldo	Lammert
19	Alva	Faucherand
20	Elvyn	Harmond
21	Martyn	Gillyett
22	Meredith	Longo
23	Carmel	Oman
24	Kimmy	Hulatt
25	Silvia	Marjanovic
26	Demetria	Saylor
27	Blanche	Catterill
28	Milli	Woolfitt
29	Johny	Treuge
30	Gareth	Demko
31	Arly	Spray
32	Joellyn	Neward
33	Alberik	Clothier
34	Lucie	Cajkler
35	Channa	Polding
36	Lesli	Elsley
37	Aldous	Wallington
38	Onfroi	Hazley
39	Roseann	Laughrey
40	Abe	McAndie
41	Florella	Guymer
42	Cami	Aleksidze
43	Karon	Troake
44	Zita	Jaume
45	Osbert	Treharne
46	Harriott	Regitz
47	Merle	Tatersale
48	Stanislaus	Huxstep
49	Phebe	Hazel
50	Wallache	Darrigrand

(50 rows)

Figure 4.3. Database Before Transaction

```
--------+---------------------------+-------------------------
      1 | Charles                   | Prawle
      6 | Tiler                     | Clinton
      8 | Lorant                    | Boothroyd
     11 | Breena                    | Bernucci
     13 | Foss                      | Elliot
     15 | Barny                     | Van den Velde
     16 | Werner                    | Doulton
     18 | Araldo                    | Lammert
     19 | Alva                      | Faucherand
     20 | Elvyn                     | Harmond
     21 | Martyn                    | Gillyett
     23 | Carmel                    | Oman
     25 | Silvia                    | Marjanovic
     27 | Blanche                   | Catterill
     28 | Milli                     | Woolfitt
     29 | Johny                     | Treuge
     31 | Arly                      | Spray
     33 | Alberik                   | Clothier
     34 | Lucie                     | Cajkler
     41 | Florella                  | Guymer
     43 | Karon                     | Troake
     44 | Zita                      | Jaume
     46 | Harriott                  | Regitz
     50 | Wallache                  | Darrigrand
     12 | lbeqxmvg                  | Greated
     32 | zxqab                     | rhnpmfvqdtrexfiavu
     17 | adropuctyocvo             | Woodyer
     39 | tmcv                      | Laughrey
      7 | Anatole                   | rggdjbsaffoqs
     30 | Gareth                    | zskfs
     45 | Osbert                    | om
      3 | Rafaello                  | fctqhkwbesulanuikrdvsud
     42 | Cami                      | bfcwdblc
     24 | pgzwbgalwktyjhlyzlqxu     | qviabnygfc
     40 | hxhzvntzvj                | immhroqizphynoduhpm
      4 | Vinnie                    | govkfoeqkojonaqfiwevbh
      5 | Lisbeth                   | xxvvdwrkln
     37 | t                         | Wallington
      9 | qkggxemshju               | Gerardin
     49 | Phebe                     | ljyxfvthymnoybnein
     10 | jsdsfycmsafcgfhsefpnbkct  | Morrott
     35 | Channa                    | zudreqmznlxvnilxuj
     22 | Meredith                  | d
     48 | Stanislaus                | hxftibxjxlxqwy
     14 | sihxxvlb                  | nwefzpsrktjmlbswhon
      2 | ugnxsxnof                 | hxwam
     36 | ewxtgiflteqdmanaxkgu      | Elsley
     38 | Onfroi                    | axjlgqjetlhn
     47 | dfrz                      | Tatersale
     26 | cxdpsn                    | votyyrvunzqe
(50 rows)
```

Figure 4.4. Database After Transaction

Table 4.2. *Hexadecimal Indicators for Phase 2*

Record Status	Hex String
Live	xx 03 00 02 09 18 00
Updated Modified	xx 03 80 02 29 18 00
Updated Expired	xx 03 40 02 05 18 00

because of the small scale of the tables and data. The next phase was to try and scale up to a full size database.

4.4 Dry Run of Full Scale Experiment

The next phase of testing was to do a sample run of the proposed experiment. This required making the complete database from TPC guidelines (Transaction Processing Performance Council (TPC), n.d.). Due to the updates and deletes, the referential integrity of the database would need to be addressed with code specific implementations.

A set of functions were created with the built-in language PLPGSQL (*PL/pgSQL - SQL Procedural Language*, 2021). These functions would randomize the rows to be selected, randomly created numbers on the fly for updates, and randomly created strings on the fly for updates. Several blog posts were helpful with algorithms for the code (Lipiński, n.d., 2011). One focused on randomization, and the other dealt with creating a random string.

To deal with the referential integrity, foreign keys were set up with cascades to handle the updates and deletes of the references. A set of triggers was also made to account for updates and deletes of an original record (going back the other way). Full code is contained in 'Dry Run Testing.zip'; a brief overview is contained in Appendix B.

This dry run of the full experiment tried to follow all procedures and models for the proposed experiment as closely as possible. The process began with a DBAN of the hard drive (*Darik's Boot and Nuke*, n.d.). It was wiped clean and verified. Next, CentOS 6.9 was installed and patched (*The CentOS Project*, n.d.). PostgreSQL 8.4 (installed by OS default) was removed, and PostgreSQL 10 was then installed (*PostgreSQL*, 2021).

The database was created via scripts. It followed the TPC model relatively closely, with a couple of modifications (Transaction Processing Performance Council (TPC), n.d.). Some table names were revised due to reserved keywords. A script created the tables, and another script implemented foreign keys as well as cascades. Seven scripts created triggers for applicable tables (Customer, District, History, NewOrder, OrderDetails, OrderLine, and Stock). All scripts were run from the PSQL command line using the \I command to execute from a file. The code for the scripts are contained in 'Dry Run Testing.zip.'

Test data was created with Mockaroo (*Mockaroo - Random Data Generator and API Mocking Tool | JSON / CSV / SQL / Excel*, n.d.). Due to limitations imposed by the Mockaroo website, 1000 records was the limit for a file (*Mockaroo - Random Data Generator and API Mocking Tool | JSON / CSV / SQL / Excel*, n.d.). Thus, five different files would need to be made for a table to get to the desired total of 5000 records. After test cases of data were created with Mockaroo, they were inserted via the PSQL command line using the \i command (*Mockaroo - Random Data Generator and API Mocking Tool | JSON / CSV / SQL / Excel*, n.d.). The code containing all of the data and scripts is contained within 'Dry Run Testing.zip.'

The file locations were found for all nine tables of the database. They were located relative to the path of /var/lib/pgsql/10/data/base/16824. To locate the file for each table, the following command was used by replacing tablename with the name of each individual table:

```
SELECT pg_relation_filepath('tablename');
```

Listing 4.1: Locating table command

The file locations for the nine tables are featured in table 4.3.

Next, a forensic image was taken and verified by comparing hash values. Data was verified by locating it in the file and comparing it to the SQL files created by Mockaroo (*Mockaroo - Random Data Generator and API Mocking Tool | JSON / CSV / SQL / Excel*, n.d.). Once again, different hex values were present compared to prior testing. Live records were now represented by a hex string of 09 00 92 01 18 00. The hex string is illustrated in 4.5.

Table 4.3. *File Locations for Tables*

Table	File
Customer	16931
District	16917
History	16866
Item	16849
NewOrder	16880
OrderDetails	16952
OrderLine	16900
Stock	16838
Warehouse	16828

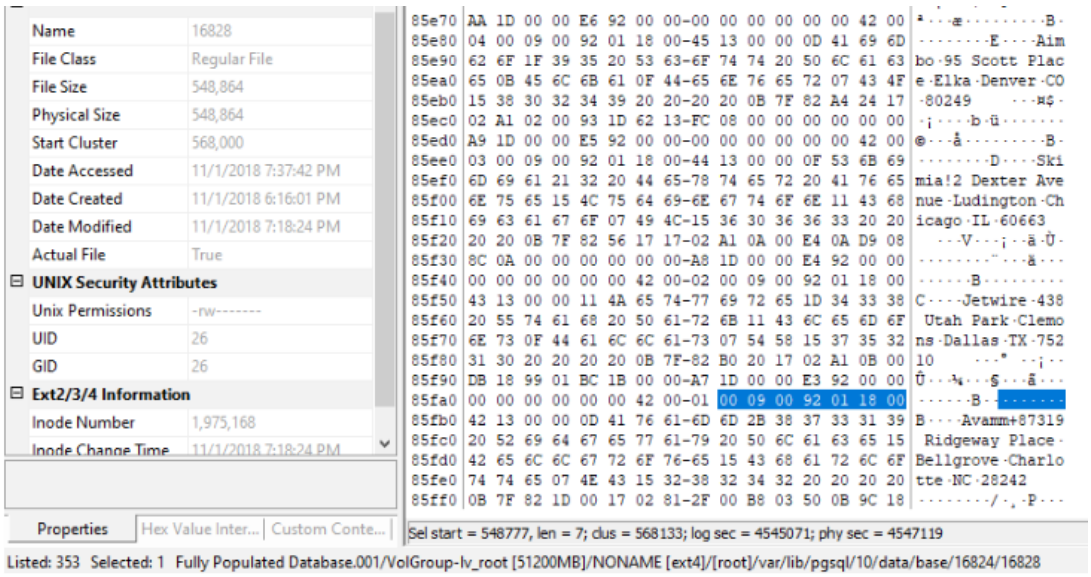

Figure 4.5. Hex Indicators for Live Records

Next up was a run of transactions to modify the data and create database slack. A set of 500 transactions to perform random updates and deletions throughout the tables of the database was run from the PSQL command line via the \I command. After the transactions completed successfully, a forensic image was made of the hard drive.

A simple verification using comparison to the previous image located database slack in the Warehouse table. By running the following command, it was possible to see tuples that had been modified and determine any missing ID numbers to indicate a deleted record:

```
SELECT * FROM Warehouse;
```

Listing 4.2: Displaying Warehouse contents command

A live record, a modified record, and a deleted record were all found in the hex. Hex indication for the live record had changed. And again, the values were different from testing. Live records were now indicated by 09 00 02 29 18 00. Updates were represented by a series of values. For some unknown reason (a rolled back transaction, perhaps?), there were now four records for an update. There were two copies of the updated live record and two copies of the updated expired record. Both copies of the updated live record had the string 09 00 02 28 18 00. The first copy of the expired updated record was indicated by 09 80 22 20 18 00, and the second was symbolized by 09 40 02 01 18 00. Deletions were indicated by 09 00 02 0A 18 00.

Two more sets of 500 transactions were successfully run using the \I command. Another image and verification process resulted after a complete set of 1500 transactions had taken place. All three types of database slack were again found in the Warehouse table. Once again, different hex indicators were present. Live records were now indicated by a hex string of 09 40 02 25 18 00. An updated live record was represented by 09 00 02 29 18 00, and an updated expired record was symbolized by 09 80 22 25 18 00. A deleted record was symbolized by 00 09 00 02 0A 18 00.

4.4.1 Problems

After everything was set up and a set of 500 transactions successfully ran, a problem was encountered. It was difficult to tell what had happened where inside of the full database. Other than doing a SELECT * FROM each table and scrolling through the entire table to find a modified result, no immediate solution came to mind.

To combat this problem, all of the logs were set to their highest level of verbosity. Although the logs now captured an extreme amount of detail, this solution did not solve the problem. The results of the functions were not captured in the logs, and neither were the cascades of update/delete nor triggers firing. As an update example from the transactions,

```
UPDATE Customer SET C_STREET_1 = string_creation(15)
```

```
4 WHERE (C_ID) = random_number_generator(1,5000);
```

Listing 4.3: Example from Transaction

the logs did not display specifics of the newly created string, which row would get the updated string, and any resulting cascades or triggers.

Since the logs were not helpful in tracking the changes to the database, a solution was needed to capture where the changes were being. A successful resolution to this problem was never found. This was one of the factors that contributed to experimentation not reaching the goal of five iterations.

Another problem that arose was the database slack itself. As observed in Phase Two Testing, the exact values of the patterns of database slack were not remaining consistent. As the experiment scaled to a database this large and complex, the patterns were changing with each iteration of transactions. The existing solution of comparison from testing was not going to apply to data of this scale.

4.5 Data Analysis and its Problems

The original plan was to use a python script designed for string searching, modify it to target the hex values gleaned from testing, and use that to locate all of the instances of database slack. The script was contained in Chapter Four of Hosmer (2014). However, once it was discovered that the hex values were changing, this method was no longer deemed feasible.

The next try to find a means of analysis was an examination of the source code for PostgreSQL. The goal was to try to find what flags indicated updated and deleted records. An initial dive into the source code did not yield positive results.

After discovering and consulting several blog posts, the necessary information was finally found (Hoogland, 2017a, 2017b, 2017c; Peschka, 2011a, 2011b, 2011c). Unfortunately, there are no definitive flags for updates and deletions; they are a conglomeration of different flags (Hoogland, 2017a, 2017b, 2017c; Peschka, 2011a, 2011b, 2011c). Several issues arose that would not make an approach based on the flags feasible. One, it required knowing which rows had been modified, which was already a separate problem. Two, it required being on the console of the machine running the database; it wouldn't be possible to do this from a forensic image.

Three, writing a piece of software this complex and to this scale is beyond the scope of this research and the author's experience.

One final idea was a recently released set of software tools by some authors affiliated with DePaul University. At DFRWS 2019 USA, the authors presented a new tool for database forensics (*DF-Toolkit*, n.d.; Wagner et al., 2019). This tool proposes to be a FTK for databases (*DF-Toolkit*, n.d.; Wagner et al., 2019). It is freely available and can be dowloaded via the link location (*DF-Toolkit*, n.d.; Wagner et al., 2019).

Unfortunately, it requires another tool by the authors, dbCarver (*DBCarver*, n.d.). That is not freely available, but it is available upon request per its website (*DBCarver*, n.d.). Four different emails sent to the contact and another member of the research team all resulted in non-response. There is no way to use one tool without the other. Thus, this solution was also abandoned. A lack of feasible options for analysis was another contributor to experimentation not continuing.

4.6 Summary

In this chapter, the phases of testing were described. Analysis and problems were also detailed.

CHAPTER 5. DISCUSSION

This chapter will discuss various aspects of the research, its significance to the area of research, and future considerations.

5.1 Research Question, Hypothesis and Results

The purpose of this research was to answer the research question, what items, known as partial records, of forensic significance can be found in database slack? All phases of testing produced partial records in database slack. Updates and deletes were contained in database slack in all phases of testing. The results support the research question.

The results are significant in that they can be applied to a real world breach or forensic investigation. By analyzing any partial records found in the hexadecimal values of a database, an investigator can learn more about the state of a database after an attack or breach. Conversely, an attacker may leverage items found in database slack to their advantage. With the popularity of PostgreSQL in the wild as a DBMS, more results like this research aid law enforcement and forensic analysts as they continue to work database breaches.

Other brands of DBMS's had technical break downs of key items like page values, hex indicators, etc. But these type of specific documents were lacking for PostgreSQL in the literature. Database slack produced noticeable, recognizable patterns in the hex of the database. Hex values changed with each forensic image in each phase of testing. And in the case of the dry run, they were changing each time modifications were made to the database. These items help to provide more information about the details and behavior of database slack than were present in the literature, and this narrows some of the gaps that were present regarding PostgreSQL in the literature.

Stahlberg found that PostgreSQL produced 100% database slack (2007). Because of the amount of time that has passed since that piece was published, it was important to see if PostgreSQL continued to exhibit the same behavior. Results indicated that PostgreSQL was producing 100% database slack, which is the same rate as Stahlberg's work (2007). Minimal

changes have occurred over the years in terms of PostgreSQL and how it processes items in database slack.

5.2 Other Considerations for Database Forensics

One consideration is how this research backs up established ideas in the area of database forensics. The complexity present in experimentation reaffirms Olivier's theory that complexity is limiting the amount of research done regarding database forensics. cite So many variables affected the research: triggers, cascades, logs, buggy code, lack of information, scale, time, etc. Complexity had an impact on the outcome of the experiments for this research.

Another notion is the impact of logs on a forensic investigation. PostgreSQL logs are set to a default level (Adedayo & Olivier, 2015). Other types of database management systems don't have them set up by default (Adedayo & Olivier, 2015). Given the wide variance of default log settings, this research provided an opportunity to have the logs set to their highest level of detail to aid in debugging and solve problems, creating an ideal situation not often seen in production environments. Even set to the most verbose setting, it is difficult to ascertain what is going on "under the hood." Just about every detail is logged, except what was really necessary for the outcome of the experiments. Only the code of the transactions was shown, but not the results of the functions within the transactions. For example, a random write to a random row would not show what was written to what row in a particular table. There was also no trace of delete/update cascades or any triggers firing as a result of the transactions. Even in this ideal setting, verbose logs did not have a substantial impact on the outcome of the research. If this is applied to a real world scenario of an investigation, it is questionable how much useful information and aid they can provide.

Scale is another issue for database forensics. It was easy to ascertain what was going on in a small table. In an elaborate database with many tables, columns, rows, and relations, it was extremely difficult to locate what was going on where in the database. The sheer volume of data was another concern.

5.3 Limitations

Several limitations came into play regarding this research. One was scale. Once the database became large, complex, and full of data, it was difficult to track what was changing where. Logs provided little help, and a feasible solution to keep track of the random changes to rows and data was never found.

Another limitation was a way to find the database slack in an efficient manner. The fact the hex indicators were changing every time made it unwieldy to locate them in a full size database. This changing aspect also made it difficult to implement an automated solution to quickly find the database slack.

Full scale experiments were limited in that they did not get past the Dry Run phase. The lack of a tracking mechanism for changes and inability to find an automated solution to locate database slack hampered completing full scale experiments. Despite the inability to perform all full scale experiments to completion, proof of concept was attained.

5.4 Uses and Implications for Privacy and Forensics

Database slack poses a threat to privacy. Stahlberg et al. focused on this aspect of database slack in both of their works (2007; 2007). It is unknown how the other databases like Oracle, SQL Server, etc. currently handle database slack. But PostgreSQL continues to produce 100% database slack. This is a major privacy issue as it pertains to users.

But database slack does have a usefulness when it relates to Digital Forensics. As in the case of this research, PostgreSQL produced database slack. The 100% production rate does it make it easier for a forensic investigation to determine specific changes that have occurred within a database. Digital Forensics has enough challenges, and anything that will make an investigation easier should be considered a positive.

There is a trade-off between privacy interests and forensic interests. Several forensic tools, FTK and Autopsy, both use PostgreSQL as a back-end for data storage (*Autopsy*, n.d.; *FTK 6.0 User Guide Attachments*, n.d.). A forensic analyst working on a case would have privacy concerns as it pertains to PostgreSQL.

5.5 Lessons Learned

The author did not accomplish everything they had planned to with this research. Even during periods where stuck, the researcher continued to think a solution to all problems would be discovered. The big lesson there is, "You can't fix everything all of the time, and sometimes, you end up failing." Expectations need to be constantly adjusted when approaching a project this large with so many unknowns.

The author learned quite a bit about coding and implementing databases of this scale. For the first time, the researcher successfully implemented a script. The author was able to learn Python in a few days to accomplish that script.

5.6 Future Work

Clearly, more work needs to be done in the area of database slack. Finding a way to finish the experiments contained herein to completion would be one step. But this research has been limited to PostgreSQL.

Further research needs to be done on the other types of databases as well, including Oracle, MySQL, SQL Server, etc. What level the other DBMS's produce database slack at this time is an unknown and needs to be illuminated. This unknown creates another gap in the literature that needs to be filled.

A related area would be testing and using the tools from the Depaul authors to see how much help they could offer. The lack of tools for database forensics has been an issue for the subfield, and any available tools need to be put to work for research as well as law enforcement.

The update producing four copies in the first set of transactions in the Dry Run Phase of testing provides a need for follow up investigation. Why did it produce this result? Was this from a failed transaction? Was this because of problems in the code? Would a transaction that rolls back create additional database slack? These questions need to be answered.

5.7 Conclusions

In this chapter, the significance of the experiments was discussed. The outcomes were also related to other areas of database forensics, and future work was suggested.

REFERENCES

Adedayo, O. M., & Olivier, M. (2014). Schema reconstruction in database forensics. In G. Peterson & S. Shenoi (Eds.), *Advances in Digital Forensics X* (Vol. 433, pp. 101–116). Berlin, Heidelberg: Springer Berlin Heidelberg. Retrieved 2016-03-03, from http://link.springer.com/10.1007/978-3-662-44952-3_8

Adedayo, O. M., & Olivier, M. S. (2013). On the completeness of reconstructed data for database forensics. In O. Akan et al. (Eds.), *Digital Forensics and Cyber Crime* (Vol. 114, pp. 220–238). Berlin, Heidelberg: Springer Berlin Heidelberg. Retrieved 2016-03-03, from http://link.springer.com/10.1007/978-3-642-39891-9_14

Adedayo, O. M., & Olivier, M. S. (2015, March). Ideal log setting for database forensics reconstruction. *Digital Investigation*, *12*, 27–40. Retrieved 2016-02-14, from http://linkinghub.elsevier.com/retrieve/pii/S1742287614001200 doi: 10.1016/j.diin.2014.12.002

Autopsy. (n.d.). Retrieved 2021-07-05, from http://www.sleuthkit.org/autopsy/

Barth, B. (2016, February). *Student SSNs exposed in University of Central Florida breach.* Retrieved 2016-04-16, from http://www.scmagazine.com/news/student-ssns-exposed-in-university-of-central-florida-breach/article/471439/

Beebe, N. (2009). Digital forensic research: The good, the bad and the unaddressed. In G. Peterson & S. Shenoi (Eds.), *Advances in Digital Forensics V* (Vol. 306, pp. 17–36). Berlin, Heidelberg: Springer Berlin Heidelberg.

Beyers, H., Olivier, M., & Hancke, G. (2011). Assembling metadata for database forensics. In G. Peterson & S. Shenoi (Eds.), *Advances in Digital Forensics VII* (Vol. 361, pp. 89–99). Berlin, Heidelberg: Springer Berlin Heidelberg. Retrieved 2016-03-01, from http://link.springer.com/10.1007/978-3-642-24212-0_7

Carrier, B. (2005). *File system forensic analysis*. Upper Saddle River: Pearson Education, Inc.

Casey, E. (2000). *Digital evidence and computer crime : forensic science, computers and the Internet.* San Diego, Calif; London: Academic.

Casey, E. (2006, March). Moving forward in a changing landscape. *Digital Investigation*, *3*(1), 1–2. Retrieved 2016-03-01, from http://linkinghub.elsevier.com/retrieve/pii/S1742287606000107 doi: 10.1016/j.diin.2006.01.007

The CentOS Project. (n.d.). Retrieved 2021-07-05, from https://www.centos.org/

Codd, E. F. (1970, June). A relational model of data for large shared data banks. *Communications of the ACM*, *13*(6), 377–387. Retrieved 2016-07-06, from http://portal.acm.org/citation.cfm?doid=362384.362685 doi: 10.1145/362384.362685

Darik's Boot and Nuke. (n.d.). Retrieved 2021-07-05, from https://sourceforge.net/projects/dban/

DBCarver. (n.d.). Retrieved 2021-07-13, from http://dbgroup.cdm.depaul.edu/DBCarver.html

DFRWS 2010 Call for Papers. (2010). Retrieved 2016-03-02, from http://dfrws.org/2010/cfp.shtml

DFRWS 2011 Call for Papers. (2011). Retrieved 2016-03-02, from http://dfrws.org/2011/cfp.shtml

DFRWS 2012 Call for Papers. (2012). Retrieved 2016-03-02, from http://dfrws.org/2012/cfp.shtml

DFRWS 2013 Call for Papers. (2013). Retrieved 2016-03-02, from http://dfrws.org/2013/cfp.shtml

DFRWS USA 2014 Call for Papers. (2014). Retrieved 2016-03-02, from http://dfrws.org/2014/cfp.shtml

DFRWS USA 2015 Call for Papers. (2015). Retrieved 2016-03-02, from http://dfrws.org/2015/cfp.shtml

DFRWS USA 2016 Call for Papers. (2016). Retrieved 2016-02-14, from http://dfrws.org/2016/cfp.shtml

DFRWS USA 2017. (2017). Retrieved 2017-02-03, from http://dfrws.org/conferences/dfrws-usa-2017

DFRWS USA 2018. (2018). Retrieved 2019-02-14, from https://www.dfrws.org/conferences/dfrws-usa-2018

DFRWS USA 2019. (2019). Retrieved 2019-02-14, from http://dfrws.org/conferences/dfrws-usa-2019

DFRWS USA 2020 Call for Papers is Open. (2020). Retrieved 2020-07-07, from https://dfrws.org/dfrws-usa-2020-call-for-papers-is-open/ (Library Catalog: dfrws.org Section: USA)

DF-Toolkit. (n.d.). Retrieved 2019-07-26, from http://dbgroup.cdm.depaul.edu/DF-Toolkit.html

Elrick, D. (2014). *Forensic examination of Windows-supported file systems* (1st ed.; L. Bernhagen & P. Beckman, Eds.). USA: CreateSpace Independent Publishing Platform.

Fasan, O. M., & Olivier, M. S. (2012, November). Correctness proof for database reconstruction algorithm. *Digital Investigation*, *9*(2), 138–150. Retrieved 2016-03-03, from http://linkinghub.elsevier.com/retrieve/pii/S1742287612000631 doi: 10.1016/j.diin.2012.09.002

Federal Bureau of Investigation. (2015). *2015 Internet Crime Report* (Tech. Rep.). Retrieved 2016-06-26, from https://www.ic3.gov/media/annualreport/2015_IC3Report.pdf

Fowler, K. (2007a). A real world scenario of a SQL Server 2005 database forensics investigation. *Information security reading room paper, SANS Institute*. Retrieved 2016-03-01, from http://docs.huihoo.com/blackhat/usa-2007/bh-usa-07-fowler-sql-server-database-forensics-wp.pdf

Fowler, K. (2007b). SQL Server database forensics. In *Black Hat USA Conference*. Retrieved 2016-03-01, from https://blackhat.com/presentations/bh-usa-07/Fowler/Presentation/bh-usa-07-fowler.pdf

Fowler, K. (2008). *SQL Server forensic analysis*. Upper Saddle River: Addison-Wesley.

Frühwirt, P., Huber, M., Mulazzani, M., & Weippl, E. R. (2010). InnoDB database forensics. In (pp. 1028–1036). IEEE. Retrieved 2016-02-14, from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5474822 doi: 10.1109/AINA.2010.152

Frühwirt, P., Kieseberg, P., Schrittwieser, S., Huber, M., & Weippl, E. (2013, May). InnoDB database forensics: Enhanced reconstruction of data manipulation queries from redo logs. *Information Security Technical Report*, *17*(4), 227–238. Retrieved 2016-02-14, from http://linkinghub.elsevier.com/retrieve/pii/S1363412713000137 doi: 10.1016/j.istr.2013.02.003

FTK 6.0 User Guide Attachments. (n.d.). Retrieved 2021-07-05, from https://support.accessdata.com/hc/en-us/articles/204056525-FTK-6-0-User-Guide-Attachments

FTK Imager. (n.d.). Retrieved 2021-07-05, from https://www.exterro.com/ftk-imager

Hackett, R. (2016, June). *IBM: Data breaches now cost $4 million on average.* Retrieved 2016-06-27, from http://fortune.com/2016/06/15/data-breach-cost-study-ibm/?xid=nl_termsheet&utm_content=29996241&utm_medium=social&utm_source=twitter

Hauger, W., & Olivier, M. (2015). Determining trigger involvement during forensic attribution in databases. In G. Peterson & S. Shenoi (Eds.), *Advances in Digital Forensics XI* (Vol. 462, pp. 163–177). Cham: Springer International Publishing. Retrieved 2016-02-27, from http://link.springer.com/10.1007/978-3-319-24123-4_10

Hauger, W. K., & Olivier, M. S. (2015, August). The state of database forensic research. In (pp. 1–8). IEEE. Retrieved 2016-02-14, from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7335071 doi: 10.1109/ISSA.2015.7335071

Hoogland, F. (2017a, July). *Postgresql block internals.* Retrieved 2021-07-13, from https://fritshoogland.wordpress.com/2017/07/01/postgresql-block-internals/

Hoogland, F. (2017b, July). *Postgresql block internals, part 2.* Retrieved 2019-08-11, from https://fritshoogland.wordpress.com/2017/07/04/postgresql-block-internals-part-2/

Hoogland, F. (2017c, July). *Postgresql block internals, part 3.* Retrieved 2021-07-13, from https://fritshoogland.wordpress.com/2017/07/07/postgresql-block-internals-part-3/

Hosmer, C. (2014). *Python Forensics: A workbench for inventing and sharing digital forensic technology*. Burlington: Elsevier Science. doi: 10.1016/C2013-0-09975-6

IBM Security. (2017, March). *IBM X-Force Threat Intelligence Index 2017.* Retrieved 2017-04-10, from https://www-01.ibm.com/common/ssi/cgi-bin/ssialias?htmlfid=WGL03140USEN&

Khosla, V. (2015, February). *Behavioral analysis could have prevented the Anthem breach.* Retrieved 2016-04-16, from http://www.forbes.com/sites/frontline/2015/02/24/behavioral-analysis-could-have-prevented-the-anthem-breach/

Kieseberg, P., Schrittwieser, S., Mulazzani, M., Huber, M., & Weippl, E. (2011, September). Trees cannot lie: Using data structures for forensics purposes. In (pp. 282–285). IEEE. Retrieved 2016-03-03, from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6061250 doi: 10.1109/EISIC.2011.18

Kim, J., Park, A., & Lee, S. (2016, August). Recovery method of deleted records and tables from ESE database. *Digital Investigation*, *18*, S118–S124. Retrieved 2016-08-12, from http://linkinghub.elsevier.com/retrieve/pii/S1742287616300342 doi: 10.1016/j.diin.2016.04.003

Kroenke, D., & Auer, D. (2010). *Database concepts* (4th ed.). Upper Saddle River: Prentice Hall.

Kruse, W. G. (2002). *Computer forensics : incident response essentials*. Boston: Addison-Wesley.

Kuchta, K. J. K. (2000, April). Computer forensics today. *Information Systems Security*, *9*(1), 29. Retrieved 2016-02-29, from http://search.ebscohost.com/login.aspx?direct=true&db=aph&AN=2881043&site=ehost-live

Leshney, S. C. (2008). *Discovering digital evidence from virtual machines: An exploratory study*. Unpublished master's thesis, Purdue University. Retrieved 2016-03-10, from http://docs.lib.purdue.edu/dissertations/AAI1469702

Lipiński, S. (n.d.). *Random String in PostgreSQL | Random Thoughts ... About Random Things.* Retrieved 2018-10-01, from https://www.simononsoftware.com/random-string-in-postgresql/

Lipiński, S. (2011, April). *Common Problem with random(min, max) | Random Thoughts ... About Random Things.* Retrieved 2018-10-01, from https://www.simononsoftware.com/common-problem-with-random-min-max/

Matthew, N., & Stones, R. (2005). *Beginning databases with PostgreSQL* (2nd ed.). New York: Apress. Retrieved 2016-03-09, from http://link.springer.com/10.1007/978-1-4302-0018-5

Meng, C., & Baier, H. (2019, July). bring2lite: A structural concept and tool for forensic data analysis and recovery of deleted SQLite records. *Digital Investigation*, *29*, S31–S41. Retrieved 2021-07-07, from https://linkinghub.elsevier.com/retrieve/pii/S1742287619301677 doi: 10.1016/j.diin.2019.04.017

Miklau, G., Levine, B., & Stahlberg, P. (2007). Securing history: Privacy and accountability in database systems. In (pp. 387–396). Asilomar, California.

Mockaroo - Random Data Generator and API Mocking Tool | JSON / CSV / SQL / Excel. (n.d.). Retrieved 2021-07-05, from https://www.mockaroo.com/

National Institute of Standards and Technology. (2001, November). *General Test Methodology for Computer Forensic Tools.* Retrieved 2016-03-10, from http://www.cftt.nist.gov/Test%20Methodology%207.doc

Olivier, M. S. (2009, March). On metadata context in database forensics. *Digital Investigation*, *5*(3-4), 115–123. Retrieved 2016-03-01, from http://linkinghub.elsevier.com/retrieve/pii/S1742287608000972 doi: 10.1016/j.diin.2008.10.001

OPM. (n.d.). *Cybersecurity Incidents.* Retrieved 2016-04-16, from http://www.opm.gov/cybersecurity/cybersecurity-incidents/

Palmer, G. (2001). A road map for digital forensic research. In *First Digital Forensic Research Workshop, Utica, New York* (pp. 27–30).

Perlroth, V. G. a. N. (2016, December). *Yahoo Says 1 Billion User Accounts Were Hacked.* Retrieved 2017-02-27, from https://www.nytimes.com/2016/12/14/technology/yahoo-hack.html

Peschka, J. (2011a, March). *PostgreSQL Row Storage Fundamentals.* Retrieved 2021-07-13, from http://facility9.com/2011/03/24/postgresql-row-storage-fundamentals/ (Section: post)

Peschka, J. (2011b, April). *PostgreSQL Update Internals.* Retrieved 2021-07-13, from http://facility9.com/2011/04/19/postgresql-update-internals/ (Section: post)

Peschka, J. (2011c, April). *PostgreSQL Update Internals | jeremiah.* Retrieved 2019-08-15, from https://facility9.com/2011/04/postgresql-update-internals/

Pieterse, H., & Olivier, M. (2012). Data hiding techniques for database environments. In G. Peterson & S. Shenoi (Eds.), *Advances in Digital Forensics VIII* (Vol. 383, pp. 289–301). Berlin, Heidelberg: Springer Berlin Heidelberg. Retrieved 2016-02-14, from http://link.springer.com/10.1007/978-3-642-33962-2_20

PL/pgSQL - SQL Procedural Language. (2021, May). Retrieved 2021-07-05, from https://www.postgresql.org/docs/10/plpgsql.html

PostgreSQL. (2021, July). Retrieved 2021-07-05, from https://www.postgresql.org/

Prosise, C., Mandia, K., & Pepe, M. (2003). *Incident response & computer forensics* (2nd ed.). New York: McGraw-Hill/Osborne.

Rogers, M. (2003, May). The role of criminal profiling in the computer forensics process. *Computers & Security*, *22*(4), 292–298. Retrieved 2016-02-29, from http://linkinghub.elsevier.com/retrieve/pii/S016740480300405X doi: 10.1016/S0167-4048(03)00405-X

Scientific Working Group on Digital Evidence. (2016, June). *SWGDE Digital & Multimedia Evidence Glossary* (Tech. Rep. No. Version 3.0). Retrieved 2017-03-22, from https://www.swgde.org/documents/Current%20Documents/SWGDE%20Digital%20and%20Multimedia%20Evidence%20Glossary

Seals, T. (2017, March). *4bn Leaked records, 10K new vulns: 2016 Massive year for cybercrime.* Retrieved 2017-04-10, from https://www.infosecurity-magazine.com/news/4bn-leaked-records-massive-year/

Stahlberg, P., Miklau, G., & Levine, B. N. (2007). Threats to privacy in the forensic analysis of database systems. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data* (p. 91). ACM Press. Retrieved 2016-03-01, from http://portal.acm.org/citation.cfm?doid=1247480.1247492 doi: 10.1145/1247480.1247492

Suffern, L. (2010, December). A study of current trends in database forensics. *Journal of Digital Forensic Practice*, *3*(2-4), 67–73. Retrieved 2016-02-14, from http://www.tandfonline.com/doi/abs/10.1080/15567281.2010.500646 doi: 10.1080/15567281.2010.500646

Sumathi, S., & Esakkirajan, S. (2007). *Fundamentals of Relational Database Management Systems* (Vol. 47; J. Kacprzyk, Ed.). Berlin, Heidelberg: Springer Berlin Heidelberg. Retrieved 2017-03-05, from http://link.springer.com/10.1007/978-3-540-48399-1 doi: 10.1007/978-3-540-48399-1

Transaction Processing Performance Council (TPC). (n.d.). *TPC-C* (Tech. Rep. No. Version 5.11.0). Retrieved 2017-01-18, from http://www.tpc.org/tpc_documents_current_versions/pdf/tpc-c_v5.11.0.pdf

Wagner, J., Rasin, A., Glavic, B., Heart, K., Furst, J., Bressan, L., & Grier, J. (2017, August). Carving database storage to detect and trace security breaches. *Digital Investigation*, *22*, S127–S136. Retrieved 2017-08-18, from http://linkinghub.elsevier.com/retrieve/pii/S1742287617301937 doi: 10.1016/j.diin.2017.06.006

Wagner, J., Rasin, A., & Grier, J. (2015, August). Database forensic analysis through internal structure carving. *Digital Investigation*, *14*, S106–S115. Retrieved 2016-02-14, from http://linkinghub.elsevier.com/retrieve/pii/S1742287615000584 doi: 10.1016/j.diin.2015.05.013

Wagner, J., Rasin, A., & Grier, J. (2016, August). Database image content explorer: Carving data that does not officially exist. *Digital Investigation*, *18*, S97–S107. Retrieved 2016-08-12, from http://linkinghub.elsevier.com/retrieve/pii/S1742287616300500 doi: 10.1016/j.diin.2016.04.015

Wagner, J., Rasin, A., Heart, K., Jacob, R., & Grier, J. (2019, July). DB3F & DF-Toolkit: The Database Forensic File Format and the Database Forensic Toolkit. *Digital Investigation*, *29*, S42–S50. Retrieved 2020-03-26, from https://linkinghub.elsevier.com/retrieve/pii/S1742287619301598 doi: 10.1016/j.diin.2019.04.010

Wright, P. M. (2007, March). *Oracle forensics in a nutshell.* Retrieved 2016-03-03, from https://oracleforensics.wordpress.com/2007/03/25/oracle-forensics-overview/

Wright, P. M. (2008). *Oracle forensics: Oracle security best practices* (D. Burleson, Ed.). Kittrell, North Carolina: Rampant TechPress.

APPENDIX A. SCHEMA

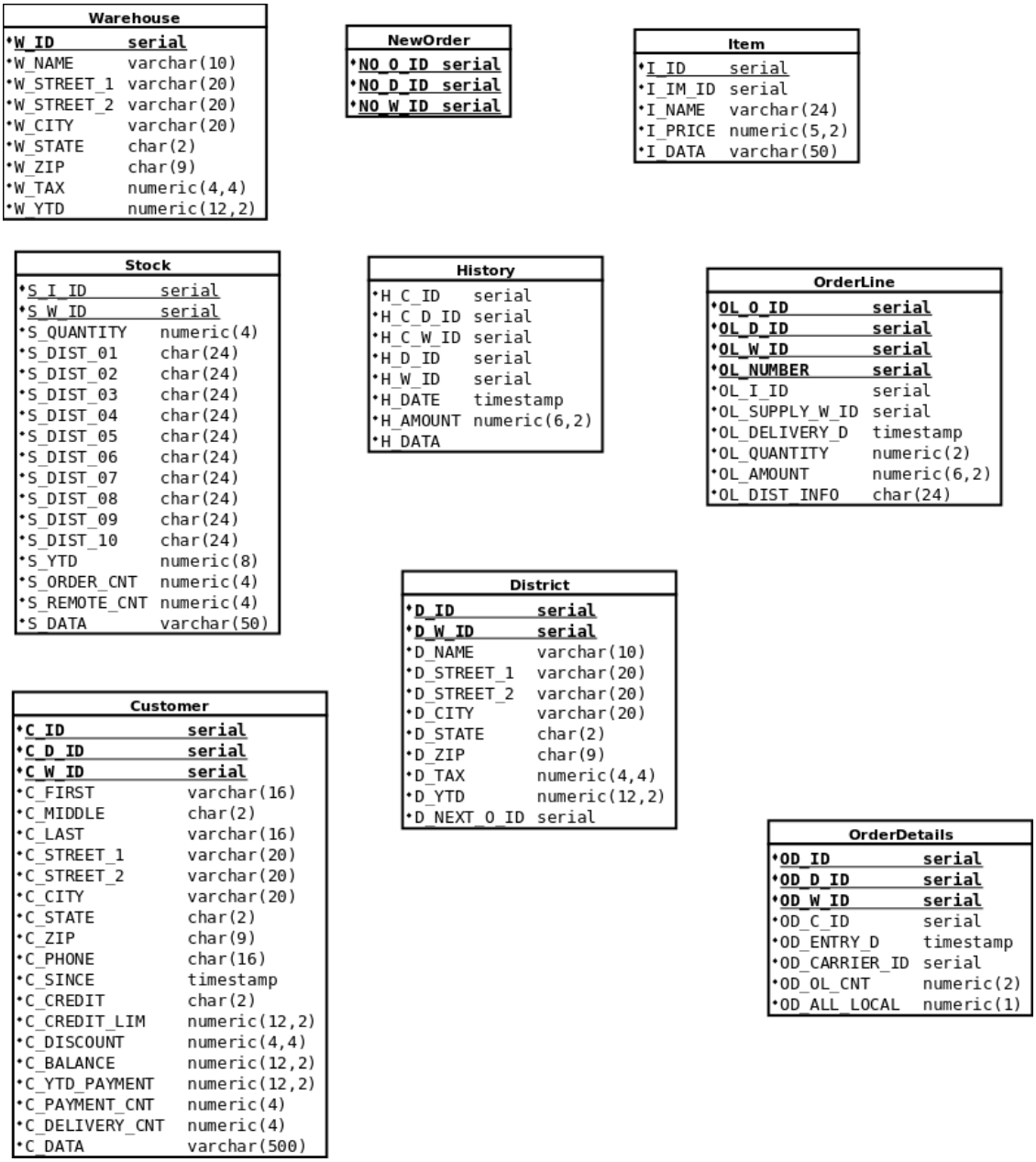

Figure A.1. Entity Relationship Diagram

Bold represents primary keys

APPENDIX B. FOREIGN KEYS

```
ALTER TABLE Stock ADD CONSTRAINT stock_fkey1 FOREIGN KEY (S_I_ID)
REFERENCES Item(I_ID) ON UPDATE CASCADE ON DELETE CASCADE;

ALTER TABLE Stock ADD CONSTRAINT stock_fkey2 FOREIGN KEY (S_W_ID)
REFERENCES Warehouse(W_ID) ON UPDATE CASCADE ON DELETE CASCADE;

ALTER TABLE History ADD CONSTRAINT history_fkey1
FOREIGN KEY (H_C_W_ID, H_C_D_ID, H_C_ID)
REFERENCES Customer(C_W_ID, C_D_ID, C_ID)
ON UPDATE CASCADE ON DELETE CASCADE;

ALTER TABLE History ADD CONSTRAINT history_fkey2
FOREIGN KEY (H_W_ID, H_D_ID) REFERENCES District(D_W_ID, D_ID)
ON UPDATE CASCADE ON DELETE CASCADE;

ALTER TABLE NewOrder ADD CONSTRAINT neworder_fkey1
FOREIGN KEY (NO_W_ID, NO_D_ID, NO_O_ID)
REFERENCES OrderDetails(OD_W_ID, OD_D_ID, OD_ID)
ON UPDATE CASCADE ON DELETE CASCADE;

ALTER TABLE OrderLine ADD CONSTRAINT orderline_fkey1
FOREIGN KEY (OL_W_ID, OL_D_ID, OL_O_ID)
REFERENCES OrderDetails(OD_W_ID, OD_D_ID, OD_ID)
ON UPDATE CASCADE ON DELETE CASCADE;

ALTER TABLE OrderLine ADD CONSTRAINT orderline_fkey2
FOREIGN KEY (OL_SUPPLY_W_ID, OL_I_ID)
REFERENCES Stock(S_W_ID, S_I_ID)
ON UPDATE CASCADE ON DELETE CASCADE;

ALTER TABLE District ADD CONSTRAINT district_fkey1
FOREIGN KEY (D_W_ID) REFERENCES Warehouse(W_ID)
ON UPDATE CASCADE ON DELETE CASCADE;

ALTER TABLE Customer ADD CONSTRAINT customer_fkey1
FOREIGN KEY (C_W_ID, C_D_ID) REFERENCES District(D_W_ID, D_ID)
ON UPDATE CASCADE ON DELETE CASCADE;

ALTER TABLE OrderDetails ADD CONSTRAINT orderdetails_fkey1
FOREIGN KEY (OD_W_ID, OD_D_ID, OD_C_ID) REFERENCES
Customer(C_W_ID, C_D_ID, C_ID)
ON UPDATE CASCADE ON DELETE CASCADE;
```

Listing B.1: Constraints