Where Provenance in Database Storage

Alexander Rasin(✉), Tanu Malik, James Wagner, and Caleb Kim

DePaul University, Chicago, IL 60604, USA
{arasin,tanu}@cdm.depaul.edu, {jwagne32,khim85}@mail.depaul.edu

Abstract. *Where* provenance is a relationship between a data item and the location from which this data was copied. In a DBMS, a typical use of *where* provenance is in establishing a copy-by-address relationship between the output of a query and the particular data value(s) that originated it. Normal DBMS operations create a variety of auxiliary copies of the data (e.g., indexes, MVs, cached copies). These copies exist over time with relationships that evolve continuously – (A) indexes maintain the copy with a reference to the origin value, (B) MVs maintain the copy without a reference to the source table, (C) cached copies are created once and are never maintained. A query may be answered from any of these auxiliary copies; however, this *where* provenance is not computed or maintained. In this paper, we describe sources from which forensic analysis of storage can derive *where* provenance of table data. We also argue that this computed where provenance can be useful (and perhaps necessary) for accurate forensic reports and evidence from maliciously altered databases or validation of corrupted DBMS storage.

Keywords: Where Provenance · Database Forensics
DBMS Anti-Tampering

1 Introduction

Where Provenance is defined as the addresses of the data values that were used to evaluate the query. It is similar to *Why Provenance* in tracing query inputs, but focuses on the location of that data. In the relational model, value location is defined as the row (tuple) and the value's location within that row. We propose to extend this concept to support database forensic analysis by computing *where* provenance based on the physical address of data copies in DBMS storage.

Database Management Systems (DBMSes) generate a multitude of data copies as part of their normal operation. For example, a materialized view (MV) stores the pre-computed results of a query drawn from the data tables in order to improve query performance. An index contains a copy of values from the indexed column(s) combined with a pointer back to the source table in order to speed up record access. Many other copies of data are created by DBMS engine actions such as caching, log entries, or internal storage defragmentation.

These and other internal copies of data can be extracted from DBMS storage with the help of *database carving* (briefly described in Sect. 2) and used for evidence of database tampering or storage corruption. Such findings must be supported by a forensic analysis framework that integrates *where* provenance to formalize storage analysis and offer provable results. Recent work by Wagner et al. [1] relied on ad-hoc case analysis (e.g., if the index value does not match the value in table record, report this as a likely indication of tampering) to report malicious activity. Such reports currently require significant effort from forensic analysts – we describe two recent cases that would greatly benefit from integration of *where* provenance into the process of forensic analysis:

Example 1. A consultant from Mandiant/FireEye (a major forensic firm) was working on a case involving a hard drive captured from the suspect. Through manual inspection of drive image, he came to suspect that the drive contained a PostgreSQL database that was uninstalled by the owner. Reconstructing raw data was the first step – but if the case went to court, the analyst could use *where* provenance to prove that the accuracy of reconstructed data report.

Example 2. A forensic analyst from Royal Canadian Mounted Police was investigating a financial fraud case. One of the sources of evidence was a snapshot of RAM from suspect's computer that contained a MySQL database (the snapshot of the hard drive was never recovered in this case). While RAM can contain data from DBMS tables, all of the in-RAM values are *copies* of the original tables. In order to establish MySQL data contents from RAM snapshot with a measure of confidence, a *where* provenance derivation could be used.

In addition to these examples, there are other security and audit applications of *where* provenance that we outline in Sect. 3. Fully deriving and continuously tracking where provenance remains a goal for future work. In this paper, we define the categories of data copies created within all major DBMSes. We consider the causal relationship between the tables and auxiliary structures in DBMS storage, including **active** data, **accessible** data, **abandoned** data.

2 Background and Related Work

Relational databases store data in page units of fixed size – even logs are often stored in system tables. Pages in relational databases (including IBM DB2, SQL Server, Oracle, PostgreSQL, MySQL, Apache Derby, MariaDB, and Firebird) follow the same basic layout structure. The work in [2] described how this layout can be generally parameterized, reconstructed and even automatically learned by loading synthetic data and observing storage behavior. Database page carving (implemented as DBCarver [3]) is a method based on this analysis that reconstructs database file contents without relying on the file system or DBMS. It is also capable of extracting the non-queryable data values, which include: (a) index values and pointers, (b) deleted records, including partially overwritten records, (c) cache contents, including pages and intermediate query results, (d) audit logs.

3 Motivating Where Provenance in DBMSes

A forensic analysts will seek to discover and prove what is or was previously stored in the database tables, or to determine what actions user may have undertaken within the DBMS. While traditional provenance explains query output by investigating the data sources and the computation process of the query, in forensic cases the target of analysis is the data table itself. For each additional data copy (index, MV, RAM), *where* provenance of that copy will serve as support and evidence for contents of the original table.

Figure 1 represents the overall flow of data copying that occurs inside a DBMS engine. After user data is loaded into tables (data loading process can create extra copies in RAM or logs), every access to these tables will cause more copying. A SQL command is initially copied into the audit log; after the query is logged, it proceeds to access the tables. Table access affects several parts of DBMS storage: modifications propagate into WAL, both read and write access caches pages in RAM (including intermediate results), and all auxiliary structures are cached in a similar manner.

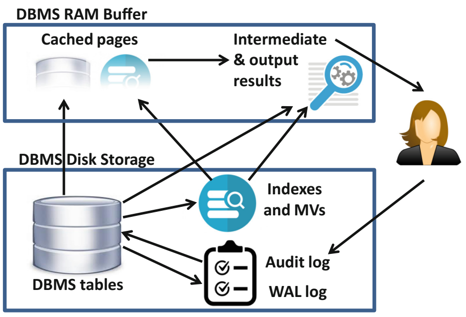

Fig. 1. The causality flow of data in DBMS.

The goal of this paper is to describe the copies that occur along the flow arrows in Fig. 1. Computing *where* provenance (not available in DBMSes) would also require *reversing* the arrows by extrapolating the connection back to the table. For example, a record found in a cached page is evidence of a tuple having been present (at some point) in a source table. The location of the cached record is known, but *where* provenance also needs the link between that copy and the original table record. Note that the original table record may already be *deleted* (can be restored) or even *erased* (cannot be restored) in which case *where* provenance offers evidence for the source data that ceased to exist.

The second application for *where* provenance is tracing back the arrow between audit log and data tables. The idea is that each forensic artifact (e.g., a deleted row) must have been caused by *some* SQL command. User commands (in Fig. 1) recorded in audit log cause changes to data tables. Therefore, if we find a storage artifact (e.g., a deleted row) that does not link back to an audit log command, this could be interpreted as a sign of log tampering.

4 Forensic Evidence in Where Provenance

The three categories of data copies include (1) actively maintained data copies which encompasses indexes, MVs, and cached copies, (2) accessible data copies (not actively maintained, may be out-of-date) including old MV values, audit and WAL logs, and (3) abandoned data copies that consist of all deleted values (in tables, MVs, and indexes), old cached values and discarded DBMS pages.

Once *where* provenance of data copies is computed, it will be unified into a report describing (1) the data values contained within the target of the investigation, (2) the relative confidence in each reported value, and (3) an extrapolated timeline information for each data value.

The target of the investigation can be either user data tables or WAL log. For data tables, Part-#1 would include *every value and every record* for which some evidence of existence (at any time) was identified. This will include data from primary evidence sources (data tables), secondary evidence sources (cached table pages, indexes, MVs), tertiary evidence sources (indexes over MVs, cached index pages), and so on. In cases like Example 2 in Sect. 1 (only RAM data is available), the entire report will be based on secondary evidence or lower.

A reported value may derive from conflicting facts (e.g., on-disk table page and in-RAM cached page disagreeing on what the value was). Part-#2 would therefore seek to unify multiple reports about each value. A value with multiple agreeing sources would have higher confidence; a value with disagreeing or lower tier (e.g., tertiary) sources would have a relatively low confidence. Most importantly, confidence report should include reasons for how it was derived.

Finally, Part-#3 would further annotate all reported values with known timeline information. Evidence of each reported value will be associated with the time range during which it (likely) existed. For example, audit logs may help determine the exact time when the value was created and subsequently deleted.

5 Conclusion

DBMS storage is a rich source of data copies created during normal operations and accessible through forensic analysis. These copies can serve as evidence of database state or proof of DBMS content tampering. *Where* provenance is the mechanism that can create a formal analytical framework to explain and quantify accuracy and of the forensic evidence reliability drawn from storage analysis.

A report of all known data augmented with confidence rating and timeline knowledge will no doubt greatly help forensic and security analysts in their job. Copies of the data are available – but these copies lack the connection to their source; in order to reason about the evidence they offer, copy flow in DBMS storage must be reverse engineered.

Acknowledgments. This material is based upon work supported by the National Science Foundation Grant CNS-1656268.

References

1. Wagner, J., et al.: Carving database storage to detect and trace security breaches. Digit. Invest. **22**, S127–S136 (2017)
2. Wagner, J., Rasin, A., Grier, J.: Database image content explorer: Carving data that does not officially exist. Digit. Invest. **18**, S97–S107 (2016)
3. Wagner, J., Rasin, A., Malik, T., Hart, K., Jehle, H., Grier, J.: Database forensic analysis with DBCarver. In: CIDR (2017)