SoK: What does it Mean to Benchmark Database Forensics?

Ben Lenard[b,c], Alexander Rasin[b], James Wagner[a]

[a] *University of New Orleans, New Orleans, LA, USA*
[b] *DePaul University, Chicago, IL, USA*
[c] *Argonne National Laboratories, Lemont, IL, USA*

Abstract

Relational Database Management Systems are the backbone of modern enterprises and public-sector services, and are thus frequent targets of security incidents, insider threats, and thorough regulatory audits. Consequently, databases have become key sources of digital evidence, requiring investigators to reconstruct past activity from audit logs, transaction logs, and backups. Although benchmarking frameworks such as those developed by the Transaction Processing Performance Council (TPC) are widely used to evaluate database performance, they do not capture forensic requirements such as evidentiary completeness, tamper-evidence, chain of custody, or regulatory compliance under GDPR and CCPA.

This survey examines the emerging domain of forensic database benchmarking. We gathered prior research on database forensics, secure logging, and tamper-evident data structures; we analyze modern forensic-ready features in commercial and open-source systems (SQL Server Ledger, Oracle Blockchain Tables, PostgreSQL pgAudit, Db2 Audit, Aurora Database Activity Streams, Oracle Real Application Security and IBM Guardium) and assess why existing benchmarks are insufficient. We propose forensic workloads, metrics, and methodologies that incorporate adversarial stressors, deleted-record recovery, and backup analysis. We also identify open research problems and call for a community-driven forensic benchmark suite. The result is an idea for evaluating not only database performance but also forensic soundness, bridging the gap between system engineering, compliance, and digital investigations.

Keywords: Forensic database benchmarking, Performance evaluation, Forensic Soundness, Benchmark metrics

1. Introduction

RDBMS' are the operational backbone of modern information systems; most business rely on RDBMS for their operational needs. This can range from e-commerce, financial transactions, healthcare records, scientific research, to government services. In addition to the primary role in transaction processing and analytics, databases also serve as repositories of digital evidence. When an individual misuses privileges, or when an external attacker exploits vulnerabilities, or when an organization undergoes a regulatory audit, the RDBMS becomes a key source of evidence about what transpired.

The importance of database evidence has grown since the rise of regulatory compliance frameworks; with these frameworks, litigation became more common, ether from a government or from private parties, with the database being the key source of evidence. In other words, the database needs to be "source of truth" that can not be maliciously altered. For example: Sarbanes–Oxley Act (SOX), the Health Insurance Portability and Accountability Act (HIPAA), the Payment Card Industry Data Security Standard (PCI DSS), and privacy laws such as the General Data Protection Regulation (GDPR) and the California Consumer Privacy Act (CCPA) mandate detailed logging, access control, and retention & purging policies. These laws transform RDBMS auditing into a legal and financial necessity since failures of RDBMS auditing or evidence retention can lead to failed investigations and severe financial penalties. As a result, reputational risk could also be a significant factor.

Despite these frameworks and legislation, the benchmarking community has not kept up with forensic needs. The Transaction Processing Performance Council (TPC) benchmarks (TPC-C, TPC-DS, TPC-E) remain the gold standard for assessing performance, focusing on throughput, latency, and cost per transaction for different data models. However, these benchmarks are fundamentally performance-centric and do not support evaluation of forensic capabilities in a DBMS. In other words, an RDBMS could excel in TPC-C performance while being incapable of reconstructing a critical transaction trail after a security incident. Without standardized measures for forensic properties such as audit completeness, tamper resistance, and backup recoverability, organizations cannot meaningfully compare RDBMS as forensic and evidential instruments (rather than just data stores).

While vendors have begun to respond with forensic-ready features, the gold standard of benchmarks have not

Email addresses: blenard@anl.gov (Ben Lenard), arasin@depaul.edu (Alexander Rasin), jwagner4@uno.edu (James Wagner)

caught up. We believe that in the absence of a standard, each vendor is going to continue creating their own proprietary solutions to the problem. For example, Microsoft SQL Server Ledger uses cryptographic digests and Merkle trees to prove that records have not been altered [1]. Oracle introduced Blockchain Tables and Immutable Tables that combine append-only semantics with retention enforcement [2]. PostgreSQL supports pgAudit for fine-grained auditing and has an ecosystem of forensic WAL analysis tools [3]. Amazon Aurora provides Database Activity Streams, which externalize audit logs to prevent insider tampering [4]. IBM Db2 includes a sophisticated audit subsystem with policy-driven logging [5]. IBM Guardium [6] extends security coverage by intercepting and monitoring activity across heterogeneous DBMS as well as Oracle's Real Application Security.

The proliferation of these products and proprietary features raise several important questions:

- How effective are these forensic features under real-world workloads and adversarial conditions?
- What performance overheads and storage costs do they impose?
- How do theses tools or frameworks interact with the workload and resulting performance while maintaining compliance requirements such as CCPA and GDPR's right to data destruction?

Since each vendor provides their own solution to forensic features, we ask: How can these solutions be compared objectively and reliably? This paper makes the following contributions to database forensics benchmarking:

- A review of prior research in database forensics, secure logging, and tamper-evident storage, highlighting how forensic requirements differ from traditional performance and benchmarking objectives.
- A systematic review of forensically-relevant features and evidentiary capabilities across major DBMSes.
- An analysis of configuration, operational, and environment factors (e.g., database parameters or logging settings) that may influence forensic outcomes.
- Discuss benchmarking metrics and standardized reporting required by database benchmark tools. We argue that forensic benchmarks should be treated as first-class companions to performance benchmarks.

Just as TPC set standards for database performance, forensic benchmarks must set standards for evidentiary strength. Traditional database benchmarks focus on performance characteristics such as throughput, latency, and cost efficiency. In contrast, forensic benchmarks are meant to evaluate how well a DBMS preserves, exposes, and enables reconstruction of evidentiary artifacts under both benign and adversarial conditions.

A forensic database benchmark approaches the DBMS as an evidentiary instrument rather than a pure data-processing engine. The objective is not to optimize transaction rate, but to assess whether investigators can reliably reconstruct historical states, attribute actions to principals, and detect tampering. Typical forensic questions include: what data existed at a specific point in time, who accessed or modified that data, whether deletions were complete or partial, and whether artifacts can be validated independently of vendor-specific tooling.

While performance remains relevant, it is considered a secondary metric that quantifies the operational overhead of forensic readiness. Effectively, forensic benchmarking complements traditional benchmarks such as TPC-C or TPC-DS by evaluating evidentiary strength in addition to query execution speed.

Defining Forensic Database Benchmarking. We define a forensic database benchmark as a reproducible combination of workloads, observable artifacts, and evaluation metrics designed to measure a DBMS's ability to preserve, expose, reconstruct, and validate evidentiary artifacts under both benign and adversarial conditions. Formally, a forensic benchmark can be expressed as a tuple:

$$B = (W, O, M)$$

- W: workloads, including both normal operations and adversarial actions
- O: observable artifacts (e.g., logs, backups, memory)
- M: metrics that quantify evidentiary properties

Unlike traditional benchmarks to optimize for throughput or latency, forensic benchmarks evaluate evidentiary soundness while capturing the performance overhead. Thus, to benchmark database forensics means to systematically evaluate a DBMS's ability to reconstruct, validate, and attribute historical events under controlled workloads, using measurable evidentiary metrics.

2. Related Work

Chopade and Pachghare present a comprehensive review of the last decade of database forensics research, covering both relational and NoSQL systems [7]. They highlight progress and challenges in key areas such as data recovery, transaction log analysis, tamper detection, and forensic process models. They identified the need for standardized benchmarks and forensic readiness in DBMSes.

Arafat Al-Dhaqm et al. [8] conduct a systematic review of existing investigation process models in the domain of database forensics, analyzing 40 prior models and identifying inherent overlaps, inconsistencies, and fragmentation across them. The authors highlight that many models are narrowly focused and tied to specific database types, limited to certain phases of investigation, or lacking clear definitions and structure, making it difficult to adopt a unified

approach across diverse DBMS platforms. They discuss the need for a more generalized and harmonized process model that abstracts the essential stages of database forensic investigation (such as preparation, data acquisition, examination, analysis, and reporting) while remaining flexible to the heterogeneity of database systems. The review also emphasizes the need for standardization of terminology, clearer task definitions, and stronger links to practical forensic readiness in databases.

Alhussan et al. [9] survey the field of database forensic investigations (DBFI) and highlight key issues—such as the lack of standardization, the heterogeneity of database systems, and ambiguity in terminology and processes. To address these challenges, they propose a new investigation process model, the Unified Forensic Model (UFM), which consolidates the various existing models into five major stages: Initialization, Acquiring, Investigation, Restoring & Recovering, and Evaluation. UFM was validated against 18 prior DBFI models, covering 64 distinct processes and by demonstrating on a simple case scenario. The result is a flexible and reusable model intended to guide forensic practitioners in varied database incident investigations.

Lenard et al. [10] presented SysGen, a tool and corresponding corpus designed to generate system-state databases and artifacts across major relational DBMS platforms for use in testing, benchmarking, and forensic analysis. The system supports customizable generation of database schemas, workloads, and system states (including data files, log files, and metadata) such that investigations of system behavior, performance, recovery, or forensic tool evaluation can use a reproducible and representative dataset. The paper includes a pre-built corpus covering multiple DBMS types, argues for the value of common reference artifacts in database research (especially for reproducibility and cross-system comparison).

Snodgrass et al. [11] argue that simply maintaining audit logs in a database is insufficient for forensic credibility because the logs themselves can be compromised by insiders, auditors, or bugs. To address this, the authors propose embedding within the DBMS a tamper-detection mechanism based on cryptographically strong one-way hash functions, together with additional metadata and a notary service. Each transaction's relevant data (for example, tuple values + timestamp) is hashed and the hash is either stored or sent to an external notary, providing a commitment that prevents undetectable modifications later. The paper includes a prototype implementation on a high-performance storage engine and shows that the approach introduces acceptable performance overheads. Overall, the contribution showed that it is feasible for a DBMS to provide validatable audit logs that can support forensic inspection of whether the audit trail itself has been tampered with. Subsequently, vendor technologies offering comparable capabilities emerged: for example, SQL Server Ledger in 2022 [1]. The goal of the Ledger is to prevent tampering with historical changes of the data. While the threat model is slightly different, audit log tampering versus table data tampering, the goal is to prevent insider treats. The Ledger tables are blockchain-inspired append-only, making them the audit log of user data operations.

Frühwirt et al. [12] presented a forensic approach targeting the InnoDB storage engine of MySQL by analyzing its low-level redo logs; these logs are intended for crash recovery and not readily modifiable by users. Because an attacker or malicious insider with administrative rights might hide or tamper with high-level audit logs, the internal redo log offers a more reliable forensic artifact. The paper details the internal structure of InnoDB redo log files, explains how to parse the log entries including statement type, tablespace IDs, page IDs, and data payloads, and demonstrates a prototype tool that successfully recovers executed SQL statements and reconstructs data changes from those logs. The result is that forensic analysts can build a more complete timeline of database activity, even when standard logs have been deleted or altered.

Shin [13] introduced a novel forensic methodology for Microsoft SQL Server that goes beyond traditional disk/log-only recovery by integrating transaction log analysis with in-memory page inspection of the database engine's buffer pool. In this method, a deletion event is first traced via the transaction log, then the corresponding data page is checked in the buffer pool, and finally the in-memory page is inspected to locate any residual deleted records. Through experiments were performed in an online SQL Server environment, the approach demonstrates recovery of deleted rows in real time, thus enabling forensic readiness in settings where DB engines run continuously. This method depends on the page evidence still being in memory which can be lost under memory pressure or anti-forensic commands; the experiments were performed under controlled conditions rather than heavy production loads.

Choi et al. [14] investigated how deleted, updated or inserted records in a Microsoft SQL Server database can be recovered from transaction log remnants residing in the unallocated space of the file system; for example, portions of the log file no longer mapped to active database pages or have been de-referenced by the DBMS. They presented a method that scans the raw disk for fragments of the SQL Server transaction log, identifies relevant log record types (`INSERT`, `DELETE`, `UPDATE`), and reconstructs the original data—including large binary objects (`BLOB`s); this is done by interpreting the log record format and linking them to the database schema. Their experiments show that even when the active log is truncated or archival logs are missing, many transaction log fragments remain in unallocated sectors and can be pieced together to yield high-fidelity recovery of past database operations.

Yoon et al. developed a forensic investigation framework tailored for MongoDB [15]. It defines procedures to identify, preserve, and analyze NoSQL forensic artifacts. The work provides one of the first systematic forensic approaches for schema-less databases and sets the foundation for reproducible analysis in NoSQL environments.

Nemetz et al. introduce the SQLite Forensic Corpus, a

standardized dataset to benchmark and evaluate forensic tools [16]. The corpus includes controlled anomalies and anti-forensic manipulations, enabling reproducible validation of tool accuracy and robustness. This work established an essential benchmark resource in forensic research.

Nissan et al. proposed ANOC, a carver for NoSQL databases to reconstruct data from raw images without using the database engine [17]. The system supports B-Tree and key-value storage structures and recovers data for Berkeley DB, ZODB, and etcd. It serves as a high-performance benchmark to evaluate NoSQL forensics.

Wagner el al. [18] introduced DBCarver, a forensic tool designed to recover and analyze database content even in cases where traditional methods fail due to deletion of files. DBCarver works directly on disk images or memory snapshots versus actual files on a filesystem. The tool uses a "page carving" approach, similar to file carving, that identifies and reconstructs database pages and records—including deleted or non-queryable data that is no longer visible to the database engine.

DBCarver [19] was evaluated across multiple row-store relational DBMSes (including SQLite, MySQL, PostgreSQL, SQL Server, Oracle, and DB2) and it could reliably recover both active and deleted data. Wagner et al. highlight the advantages of being platform agnostic; the tool works with any corrupted database as well as recover deleted content from free space or memory. However, it does face limitations: deleted data can be permanently lost once overwritten, carving large datasets is resource-intensive, and the system depends on accurate knowledge of DBMS storage layouts, which may change across versions.

Lenard et al. [20] investigated the auxiliary parts of the database storage, including data in backups. The paper showed that backups capture storage at the block or page level and frequently include old versions of data that should have been erased. As a result, even when rows are logically deleted in a live database, their remnants may continue to exist permanently within backup sets. The paper tested multiple widely used DBMSes, including MySQL, PostgreSQL, Oracle, SQL Server, and IBM Db2. In each case, they showed that deleted rows still appeared in backup files, confirming that the issue was not specific to a single system but a general property of how RDBMSes interact with backup mechanisms.

Literature Collection Methodology. This survey follows a structured literature review approach. We queried major digital libraries including IEEE Xplore, ACM Digital Library, SpringerLink, and Elsevier using keywords such as "database forensics", "audit logs", "data recovery", "tamper detection", and "forensic benchmarking".

Studies were included if they addressed (1) forensic analysis of DBMS, (2) recovery or reconstruction techniques, or (3) audit and logging mechanisms relevant to evidentiary use. Papers focused solely on performance optimization without forensic relevance were excluded.

3. Platform Support

RDBMS vendors are increasingly embedding forensic readiness into DBMS features. As databases become the cornerstone of investigations, having proof of non-tampering is key to any analysis.

3.1. Tamper Resistant Tables

SQL Server Ledger provides organizations with a mechanism for ensuring the integrity and authenticity of the data within the database. By leveraging blockchain technology, Ledger creates an immutable, cryptographically verifiable record of all changes to tables, preventing tampering or undetected modifications. This is especially valuable for industries that must meet strict compliance or auditing requirements, as it allows external auditors and internal stakeholders to independently validate data integrity without relying solely on trust in the database administrator. For example trading platforms and exchanges where trillions of dollars of assets exchange hands daily. Beyond compliance, Ledger enhances transparency and accountability, simplifies forensic investigations, and helps protect against insider threats or malicious actors by offering a provable history of data changes. In other words, by adding the Ledger, the ability to alter the database's contents is almost negated, re-enforcing the database as the source of true evidence.

Similar to the Ledger, Oracle built Blockchain Tables to implement row-level cryptographic chaining for append-only tables. They allow users to store and manage data in an immutable, tamper-evident way, directly within the database. The blockchain characteristics enable append-only inserts, preventing updates or deletes, and maintaining a cryptographic chain of rows to ensure integrity. Each row is linked to the previous one using a hash, so any attempt to alter past data is immediately detectable. Similar to the Ledger, this makes blockchain tables valuable for use cases such as audit trails and regulatory compliance.

Similar to Oracle Blockchain tables, and SQL Server Ledger, IBM Db2 has a tech preview in 11.5.5 of Hyperledger Fabric data sources via federation. Its goal is to have blockchain data accessible in their database as tamper-evident, append-only storage designed to guarantee data integrity. Since it is currently a tech preview, it is not guaranteed to be rolled into the Db2 platform.

Table 1 summarizes support for blockchain tables in the databases we have discussed. You might have noticed that Postgres and MySQL/MariaDB do not appear in the Table 1, but that's because they do not support this functionality as of this time.

3.2. Database Activity Logging

SQL Server provides robust auditing features through SQL Server Audit, which was introduced in SQL Server 2008 and expanded in later versions. This feature allows administrators to capture events at the server, database,

Table 1: Comparison of Blockchain Tables in SQL Server, Oracle, and IBM Db2

Feature/ DBMS	**SQL Server Ledger**	**Oracle Blockchain Tables**	**IBM Db2 Hyperledger data sources**
First Release	SQL Server 2022	Oracle Database 21c	Db2 11.5.6
Core Idea	Creates tamper-evident records using a ledger view with cryptographic digests that can be verified externally	Blockchain table type with append-only inserts and row chaining	Blockchain table type with append-only inserts and row chaining
Data Model	Any table can be enabled as a ledger table (regular or updatable)	Special table created with `BLOCKCHAIN` clause	Special table created with `Federation` clause
Immutability	Prevents undetected changes; cryptographic proof via hashes and digests	`INSERT`-only, no `UPDATE` or `DELETE`; enforced immutability	External data source
Cryptographic Link	Uses SHA-256 digests stored in a system ledger. Can be exported to verify	Rows are linked using hash values	N/A
Verification	External digest verification (auditors don't need access to SQL Server)	Built-in verification using DBMS functions	N/A
Use Cases	Compliance, auditing, external verification of financial/health/IoT data	Audit trails, supply chain, regulatory compliance, provenance tracking	Regulatory compliance, healthcare, financial records, audit logs
Integration	Works seamlessly with standard SQL Server tools; can store digests outside the database for third-party verification	Fully inside Oracle DB, no external blockchain needed	Not fully inside Db2 DB, external blockchain needed

and object level, storing them either in the Windows Security Log, Application Log, or a file target. SQL Server Audit supports fine-grained policies, enabling tracking of schema changes, permission grants, and data access. Additionally, the Change Data Capture (CDC) and Change Tracking features can log row-level modifications for data lineage and replication use cases. CDC can be any number of products from IBM CDC to Oracle Golden Gate.

Oracle's Unified Auditing architecture is the most comprehensive auditing frameworks among enterprise databases. This consolidates audit trails from various sources (traditional auditing, Fine-Grained Auditing, Database Vault, and Label Security) into a single repository. Unified Auditing supports both standard and fine-grained audit policies, allowing organizations to monitor access to sensitive tables, detect anomalous queries, and enforce regulatory compliance. Oracle's ecosystem also integrates seamlessly with Oracle Audit Vault and Database Firewall (AVDF), which aggregates audit logs from multiple sources and provides advanced monitoring, alerting, and reporting.

IBM Db2 also includes a variety of auditing features centered around security auditing and database activity monitoring. The built-in audit facility lets administrators capture system events (like authentication attempts, authority grants, and object modifications) into audit logs stored in special tables. These logs can be queried for forensic analysis or exported for compliance review. Db2 also supports event monitors, which can track activities like connections, SQL statements, and deadlocks for deeper diagnostic logging.

PostgreSQL takes a modular approach to logging and auditing, relying heavily on configuration and extensions. Out of the box, PostgreSQL can be configured to log all queries, connections, and errors using the log_statement and log_duration parameters, writing results to the PostgreSQL log files. For stricter compliance requirements, the pgaudit extension provides fine-grained logging of SQL statements, DDL operations, and role changes, capturing this activity in the standard PostgreSQL log format. One can then feed these logs into a centralized monitoring solutions like syslog.

MySQL supports auditing primarily through plugins. The most widely used is the MySQL Enterprise Audit Plugin, which is part of the commercial Enterprise Edition. This plugin allows policy-based auditing of connections, queries, and schema modifications, with logs stored in either XML or JSON formats. Open-source alternatives like the MariaDB Audit Plugin are also available for community deployments. Beyond auditing, MySQL's general query log and binary log capture query activity and transactional changes, which can be repurposed for forensic investigations or compliance auditing. However, these require careful configuration and management to avoid performance overhead and excessive storage consumption.

Agnostic to the RDBMS engine, IBM Guardium is a comprehensive data security and compliance platform that integrates with Db2, Oracle, SQL Server, and other major databases. It provides real-time database activity monitoring (DAM), policy enforcement, and automated alerting to detect suspicious behavior. Guardium collects audit logs from multiple sources and correlates them with user activity, helping organizations comply with strict regulations like GDPR, HIPAA, and SOX. Unlike built-in auditing tools, Guardium operates externally, meaning it is not tied to the RDBMS kernel as a built-in solution.

Although these forensics-ready features enhance data

integrity, their validation mechanisms remain largely proprietary. SQL Server Ledger, Oracle Blockchain Tables, and IBM Db2 Hyperledger sources all use unique cryptographic models and verification procedures that prevent direct comparison. Without a vendor-neutral evaluation framework, it is impossible to assess whether these mechanisms offer equivalent evidentiary strength or performance trade-offs. A unified forensic benchmark would therefore allow objective, cross-DBMS comparison and promote transparency, enabling practitioners to evaluate forensic assurance alongside traditional performance metrics.

Tables 2 and 3 summarize the auditing and logging features available in major databases.

3.3. Auditing vs. Forensic Analysis

It is important to distinguish auditing from forensic analysis. Auditing is a proactive process that records system activity, whereas forensics is a post-incident process that reconstructs, validates, and interprets past events.

While auditing mechanisms provide the raw data necessary for investigation, they do not guarantee evidentiary completeness, integrity, or recoverability. Forensic benchmarking therefore evaluates not only whether data is recorded, but whether it is sufficient and reliable for reconstruction under adversarial conditions.

4. Forensic Tooling Beyond the Core DBMS

IBM Guardium is an advanced data security and compliance platform designed to protect sensitive data environments, including databases, data warehouses, and big data systems. It provides continuous monitoring, auditing, and real-time policy enforcement to detect and prevent unauthorized access, data breaches, and insider threats. It provides data activity monitoring (DAM), vulnerability assessment, as well as data discovery and classification. Furthermore, it provides compliance reporting for regulations such as GDPR, HIPAA, SOX, and CCPA [6].

Oracle Real Application Security (RAS) is an advanced access control framework integrated into the Oracle Database that provides fine-grained, application-level security for modern, multi-tier environments. It extends traditional database privileges and Virtual Private Database (VPD) models by introducing a unified, policy-driven mechanism that aligns security with application semantics rather than database objects alone.

The key difference between RAS and IBM Guardium is that Guardum operates externally to the database as a separate product whereas RAS is designed to enhance the security within the Oracle database itself.

5. Gold Standard in Database Benchmarking

The Transaction Processing Consul (TPC) benchmarks are designed to capture different aspects of database performance under workloads that resemble real-world applications. Each benchmark targets a specific class of systems, such as transactional processing or analytical decision support, and measures not only raw speed but also efficiency and scalability.

The most widely recognized transactional benchmark is TPC-C, which simulates a warehouse order-processing environment. It models multiple transaction types such as new orders, payments, deliveries, and stock checks, stressing concurrency control, transaction logging, and overall throughput under heavy multi-user access. By doing so, TPC-C captures the performance of RDBMSes in high-volume OLTP (Online Transaction Processing) scenarios.

Its successor, TPC-E, represents a more modern workload drawn from financial brokerage operations. It uses stricter schema rules, more complex transactions, and referential integrity to better reflect the intricacies of real business systems. Both TPC-C and TPC-E report metrics in terms of completed transactions per unit of time, providing a standardized way to measure OLTP capability.

On the analytical side, TPC-H focuses on decision support workloads by defining a suite of ad-hoc, parameterized queries over large datasets. These queries test a system's ability to optimize joins, indexing strategies, and execution plans for unpredictable query patterns. However, as analytical workloads evolved, the TPC-DS benchmark was introduced as a more comprehensive replacement.

TPC-DS models a data warehouse environment with 99 SQL query templates, incorporating extract-transform-load (ETL) tasks, reporting, and complex analytical queries. This benchmark stresses query optimization, resource management, and performance under mixed workloads, making it a robust measure of OLAP (Online Analytical Processing) systems.

Taken together, the TPC benchmarks capture transactional throughput, query optimization, concurrency management, and scalability across both OLTP and OLAP workloads. They also incorporate price/performance metrics, showing not just how fast a system runs but how efficiently it delivers performance per dollar. By enforcing standardized rules and requiring audited results, the TPC ensures fair, transparent comparisons across vendors, making these benchmarks a cornerstone in evaluating relational database systems.

5.1. Taxonomy of Forensic Tasks

Rather than enumerate all forensic activity, we categorize database forensic tasks into representative classes. A forensic benchmark must select representative tasks from each category to ensure comparability across systems.

- **Reconstruction**: recover deleted or modified data
- **Attribution**: identify actors responsible for changes
- **Tamper Detection**: detect unauthorized actions
- **Timeline Reconstruction**: order events accurately

Table 2: Auditing framework, immutability, and granularity in major databases.

DBMS	Auditing Framework	Data Integrity/Immutability	Granularity
SQL Server	SQL Server Audit; CDC; Change Tracking	Ledger tables w/ cryptographic digests	Server-, DB-, object-level
Oracle	Unified Auditing; FGA; AVDF	Blockchain Tables	Fine-grained (DDL/DML)
IBM Db2	Native audit; Event monitors; Blockchain Tables; Guardium	Blockchain Tables	System events; SQL; grants
PostgreSQL	`log_statement`; pgaudit	No native capability; append-only tables	Query-level (logs/extensions)
MySQL	Enterprise Audit Plugin (commercial); MariaDB Audit Plugin (open-source)	No native immutability; relies on external mechanisms	Connection-level, query-level via plugins

Table 3: Operational and compliance aspects in major databases. Note some products require extra licensing for full use of the features.

DBMS	Log Storage	Compliance Use Cases	Advanced Monitoring
SQL Server	Windows Event Log; Application Log; file targets	SOX, HIPAA, PCI DSS	External verification w/ digests
Oracle	Unified audit trail repository	GDPR, HIPAA, SOX, financial auditing	AVDF for cross-DB correlation
IBM Db2	Audit tables; Guardium repository	Financial, healthcare, government	Guardium real-time monitoring
PostgreSQL	Standard logs; syslog integration	FedRAMP, PCI DSS (configurable)	External SIEM integration
MySQL	XML/JSON files, query logs, binary logs	PCI DSS, SOX (with plugins)	Requires external monitoring or SIEM

5.2. Dimensions of Forensic Benchmark Readiness

A database forensics benchmark must evaluate properties that are largely absent from existing performance-centric benchmarks. At minimum, such a benchmark must exercise and quantify the following capabilities:

Logging and Provenance. The benchmark must evaluate the completeness and fidelity of audit logs across logical (SQL statements, transactions) and physical layers (write-ahead logs, redo logs). It should asses resistance to log truncation, selective deletion, and administrative abuse, and the ability to correlate logical and physical evidence.

Tamper Detection. The benchmark should test whether unauthorized modifications to data or logs can be detected. This includes evaluation of cryptographic integrity mechanisms such as hash chains, digests, or Merkle trees, and whether verification can be performed independently of vendor tooling.

Data Recovery. A core requirement is measuring the ability to reconstruct deleted or modified data from logs, pages, free space, memory, or backups. Recovery fidelity and temporal accuracy are essential forensic metrics.

Deletion and Erasure Semantics. The benchmark must distinguish between logical deletion, physical overwriting, and cryptographic erasure. It should also examine how deletions interact with backups, replicas, and retention polices under regulations such as GDPR and CCPA.

Operational Overhead. The benchmark must measure the performance, storage, and administrative overhead introduced by forensic features, including their impact on backup, restore, replication, and maintenance operations.

5.3. Benchmarking as a Multi-Objective Problem

Traditional benchmarks optimize for performance metrics such as throughput and latency. In contrast, forensic benchmarking must evaluate both performance and evidentiary soundness. We define two orthogonal dimensions:

- Performance (P): latency, throughput, resource use
- Forensic Soundness (F): completeness, integrity, recoverability

Thus, forensic benchmarking is a multi-objective evaluation problem that seeks to maximize F while minimizing degradation in P. Unlike TPC benchmarks, which define fixed workloads and optimize for throughput and latency, forensic benchmarks must incorporate adversarial actions, ground-truth validation, and evidentiary metrics. While TPC evaluates execution efficiency, forensic benchmarks evaluate reconstruction capability and integrity guarantees. These differences require fundamentally different workload design and evaluation criteria.

6. Database Workloads

Hsu at al. [21] examines the gap between standardized benchmarking workloads and the behavior of databases in real-world production environments. The authors analyze traces from actual enterprise systems, highlighting workload patterns such as transaction diversity, query complexity, concurrency levels, and access distributions. Their central argument is that while TPC benchmarks like TPC-C, TPC-H, and TPC-DS provide a valuable standard for

comparing performance, they often oversimplify or fail to capture the nuanced characteristics of production systems. This raises concerns about how well benchmark results translate to operational performance in live deployments.

One of the key findings in the paper by Hsu et al. [21] is that production workloads exhibit far greater variability than what TPC benchmarks model. For example, actual OLTP environments may include a broader mix of transactions beyond the narrow set defined by TPC-C, often with application-specific logic and dependencies. Similarly, analytical workloads observed in practice tend to include more skewed access patterns, ad-hoc queries, and mixed update-query interactions, whereas TPC-H and TPC-DS queries are carefully parameterized and uniformly distributed. This paper argues that this difference can lead to benchmarks overstating performance or underestimating system bottlenecks when applied to production contexts.

Another major observation is the interaction of mixed workloads in production systems. Many enterprises run transactional and analytical queries simultaneously, a characteristic not fully addressed by most TPC benchmarks, which usually isolate OLTP and OLAP into separate tests. In reality, concurrent execution of short transactions and long-running queries creates contention for CPU, I/O, and locks, leading to unpredictable performance. Hsu et al. emphasize that future benchmarks should more accurately represent these mixed workload conditions to provide a fair evaluation of modern database systems.

7. Quantitative Evaluation of Forensic Properties

The metrics in the audit log must capture evidence of what has occurred, to enable reconstruction of the event, and to offer verifiable forensics evidence that the event has occurred. In addition to classical query performance measures such as latency and throughput, forensic benchmarking requires new quantitative metrics that speak to evidentiary strength and reliability. Key measures include:

- **Recovery Recall Percentage**: proportion of deleted or tampered records successfully reconstructed from log or other sources of evidence.
- **Tamper-Detection Precision**: the rate of correctly identified alterations (by an attacker) versus false positives.
- **Log Coverage Ratio**: percentage of system activity captured across all logging layers.
- **Evidentiary Completeness Score**: ability to reconstruct full transactional context, including timestamps and actors.
- **Forensic Overhead**: CPU, I/O, and latency penalties introduced by forensic features when enabled.
- **Verifiability Index**: ease of cryptographic proof validation and independence from vendor-specific tools.

For example, recovery recall can be formally defined as:

$$\text{Recovery Recall} = \frac{|R \cap G|}{|G|}$$

where R is the set of recovered records, and G is the ground truth set. Tamper-detection precision is defined as:

$$\text{Precision} = \frac{TP}{TP + FP}$$

where TP and FP denote true and false positives.

Formalizing such metrics supports reproducible, data-driven evaluation of forensic soundness across platforms. In state-of-the-art TPC, benchmarking is centered around performance either as execution time of a query statement or in terms of transactions per second. When we factor in a specific workload scenario, even in the same data processing category, there is always going to be some real-world variability; two different industries running a Enterprise Resource Platform (ERP) could have different workloads.

While both forensic and performance types of benchmark might measure something within the database, their goals are fundamentally different, despite having a natural operational overlap. In other words, both measurements help identify best performing database enginge, but from different vantage points.

8. AI-Assisted Validation of Database Forensic Tools

This section does not introduce a new benchmarking dimension, but highlights how forensic benchmarks can serve as validation frameworks for AI-assisted forensic tools. As AI is developing and improving, there's a new trend of using AI to write code. Thus, forensic database benchmarks are the validation and self-improvement tools needed for AI-assisted development of forensic tools. As machine-learning and large-language-model (LLM)–based systems are increasingly used to automate evidence triage, anomaly detection, and timeline reconstruction, there is a growing need for ground-truth validation environments in which these systems can be tested, measured, and iteratively refined. A standardized forensic benchmark provides exactly such an environment: controlled workloads, known adversarial actions, labeled ground truth, and reproducible artifacts against which AI outputs can be objectively scored. Rather than relying on anecdotal case studies, AI systems can be evaluated using benchmark metrics such as recovery recall, attribution accuracy, tamper-detection precision, and temporal ordering correctness, as proposed for forensic benchmarking.

For example, an AI-based forensic assistant tasked to reconstruct a timeline from WAL segments, audit logs, and backups can be benchmarked against synthetic scenarios where deletions, log truncation, or backdated updates are injected in a known sequence. The benchmark's ground truth enables quantitative comparison between the AI-generated reconstruction and the actual event sequence,

exposing failure modes such as missed recovery opportunities, incorrect attribution, or false positives. Similarly, AI models trained to detect insider abuse can be validated against benchmark workloads simulating DBA-level attacks, such as forced VACUUM operations or selective audit log removal and measure if the system correctly flags inconsistencies between logical and physical artifacts.

Beyond validation, forensic benchmarks will also enable closed-loop improvement of AI forensic systems. Because benchmark scenarios are repeatable, AI models can be retrained or fine-tuned based on prior errors, then reevaluated under identical conditions to measure improvement. Over time, this creates a feedback cycle in which forensic benchmarks serve not only as evaluation tools but also acts as a training corpora for AI-driven investigators.

9. Example Forensic Benchmark Scenario

To illustrate how forensic benchmarking can be operationalized, we present a simple benchmark scenario.

Workload: insert 1M records, update a subset of records, delete a subset of records, force checkpointing and log truncation, execute backup operations, and simulate adversarial actions (e.g., log deletion)

Artifacts Collected: transaction logs (WAL/redo), audit logs, data pages and free space, and backup images.

Metrics Evaluated: recovery recall, tamper detection precision, timeline accuracy, and overhead.

Such a benchmark could be instantiated using systems such as PostgreSQL, SQL Server, or Db2, enabling cross-platform comparison of forensic capabilities under controlled conditions. This reinforces that configuration must be treated as an explicit dimension in forensic benchmarking, rather than an implicit environmental variable.

Ground truth for this scenario can be established by recording the exact sequence of operations and injected adversarial actions. This enables direct comparison between expected and reconstructed state, allowing quantitative scoring of recovery accuracy and tamper detection. Such scenarios can be generated using existing tools such as SysGen, which produce reproducible database states, logs, and artifacts across multiple DBMS platforms.

10. Impact of the Database and Operating System Configuration on Forensic Outcomes

Database configuration parameters and operating system tuning have a direct and often underappreciated impact on forensic recoverability and evidentiary completeness. While forensic benchmarks typically focus on workload behavior and logging features, configuration choices at both the DBMS and OS levels significantly alter what evidence is retained, overwritten, or observable during an investigation. As a result, forensic benchmarks must explicitly account for configuration state as a first-class variable rather than an implicit background condition.

At the database level, parameters governing logging, checkpointing, and memory usage are especially influential. For instance, aggressive checkpoint intervals or small redo/WAL retention windows can reduce recovery fidelity by overwriting transaction history more quickly, limiting the tool's ability to reconstruct deleted or modified rows. Conversely, enabling verbose auditing, fine-grained logging, or change-data-capture mechanisms increases evidentiary completeness but introduces measurable storage and performance overhead. Maintenance operations—such as VACUUM in PostgreSQL, REORG in Db2, or index rebuilds in SQL Server—can also function as inadvertent anti-forensic actions by reclaiming free space that might otherwise preserve deleted tuples. A forensic benchmark must therefore test both default and forensic-enhanced configurations, as well as adversarial settings where logging is minimized or selectively disabled, as reflected in the DBMS configuration dimension of the database forensic benchmark taxonomy. Furthermore, archive logging being enabled is also a significant source for forensic evidence; the receivers of the archive logs are often external to the database server, so tampering with these logs is more difficult, whereas if circular logging was enabled, a bad actor could force a log switch to clean up their tracks.

Operating system tuning further complicates forensic analysis. Parameters related to filesystem caching, memory pressure, and temporary storage materially affect evidence persistence outside the DBMS engine. For example, the use of RAM disks (tmpfs) for temporary tablespaces, or even to boot the operating system, redo logs, or audit staging areas may significantly improve performance but can eliminate critical forensic artifacts upon reboot or crash. Consider an in-memory database such as TimesTen or VoltDB where the archive logs are fed to another system, do you need a traditional disk based system, or can you boot the system into RAM only? This is a very common practice in the HPC community since at every reboot the system is redeployed and is free from alterations.

These interactions underscore why forensic benchmarking must evaluate systems holistically, spanning DBMS internals, configuration parameters, and OS behavior. Two databases running identical workloads and forensic features may yield dramatically different investigative outcomes solely due to tuning choices made for performance or cost reasons. Incorporating configuration variation into forensic benchmarks enables practitioners to quantify these trade-offs explicitly, revealing how operational optimizations can unintentionally weaken forensic soundness. This reinforces the central argument of this work: databases must be evaluated not only as execution engines, but as evidentiary systems whose behavior is shaped by every layer of the software stack.

11. Open Problems

Despite decades of progress in performance benchmarking, no TPC-style forensic benchmark exists for RDBM-

Ses. The community lacks a reproducible, open, and auditable framework to measure evidentiary completeness, tamper resistance, and recovery fidelity under adversarial conditions. Existing digital-forensic test corpora such as the SQLite Forensic Corpus [16] prove the feasibility of standardized datasets, yet no equivalent resource covers enterprise-scale SQL systems. Building on SysGen [10] to generate synthetic tampering scenarios, deleted-row traces, and backup artifacts across heterogeneous DBMS provides a valuable foundation for cross-platform benchmarking.

Another critical research gap is the absence of standardized forensic metrics and ground-truth datasets (several non-database corpora are widely available in the forensic community). Without defined measures of recovery accuracy, false-positive rate, or log-coverage, tool and platform comparisons remain subjective. Developing an open corpus with labeled evidence and reproducible workloads would enable repeatable experimentation and objective scoring. In addition, adversarial workload modeling is underexplored. Most forensic evaluations operate in benign environments rather than simulating insider attacks, log tampering, or selective erasure. A proper benchmark must incorporate controlled attack scripts and stress conditions to test resilience under realistic threat models.

Finally, forensic benchmarking must evolve to capture compliance trade-offs. Regulations (e.g., GDPR and CCPA) impose both retention and erasure mandates that interact directly with forensic logging and backup policies. Measuring these interactions, such as the cost of secure deletion or audit retention overhead, remains an open challenge.

While we have to consider the entire field of forensics, we can first focus on a few key areas such as logging, access controls, and backup and recovery. These functions are critical to database operations, and enabling this functionality to provide forensic details warrants the questions of the impact these features. That is, we should evaluate the performance overhead, evidential benefits, and any other impact on normal operational procedures.

While different industries that use the same software could have different workload characteristics as seen in Hsu et al. [21], the generic TPC benchmarks can be used to look at the impact of these forensic tools. While one might argue that the forensics aspect is a binary answer, which is somewhat true based on the technology (e.g., logging, fine grain access control). However, what is the cost of enabling these functionalities in a database? Does the database incur higher IO, CPU, or slower query performance? A low hanging fruit solution may be to add a dimension to the results of TPC benchmarks that indicates which forensics-enabling technologies were enabled when the benchmarks were executed. Do blockchain tables take longer for backup and recovery operations? And if so, how much do they impact other performance characteristics?

These aspects of database forensics seem to be fundamental as we introduce various technologies and as we enter an age of heightened security.

12. Conclusion

Although forensic workloads may occur less frequently than transactional workloads, their impact is disproportionately high. Failures in forensic reconstruction can invalidate legal evidence, lead to regulatory penalties, and undermine trust in critical systems.

This survey underscores that while RDBMSes are indispensable not only for business operations but also for audits, the benchmarking community has yet to address forensic soundness as a measurable property. Existing standards like the TPC suite excel at quantifying throughput, latency, and scalability, but they fail to determine if a database can reliably reconstruct past events or meet data retention and erasure obligations under frameworks such as GDPR and CCPA. Our analysis of commercial and open-source DBMS platforms—featuring SQL Server Ledger, Oracle Blockchain Tables, PostgreSQL pgAudit, Db2 Audit, Aurora Activity Streams, Oracle RAS, and IBM Guardium—shows clear progress toward "forensic-ready" capabilities, yet there remains no unified way to assess their evidentiary completeness or operational cost.

We argue that forensic benchmarking must evolve into a first-class complement to performance benchmarking with standardized workloads, metrics, and adversarial playbooks to evaluate resilience under deletion, tampering, and recovery scenarios. Such a benchmark would enable objective cross-DBMS comparison of both security efficacy, bridging the divide between database engineering, compliance auditing, and digital investigations. Ultimately, establishing a community-driven, open forensic benchmark suite will promote reproducibility, transparency, and accountability—ensuring that databases are measured not only by how fast they process information, but by how faithfully they preserve the truth.

Beyond establishing a benchmark, several systemic gaps must be addressed. The community needs forensic datasets, standardized evaluation metrics, and adversarial test harnesses that reflect real-world threats. Furthermore, an independent governing consortium analogous to the TPC should be formed to certify results, publish specifications, and maintain an open repository of benchmark data. By institutionalizing these practices, database forensics can transition from fragmented, vendor-specific evaluations to a reproducible science of evidentiary assurance. In doing so, we elevate databases from mere performance instruments to trustworthy, certifiable sources of digital truth.

Acknowledgments

This work was partially funded by the Louisiana Board of Regents Grant LEQSF(2022-26)-RD-A-30 and by US National Science Foundation Grant IIP-2016548. Argonne National Laboratory's work was supported by the U.S. Department of Energy, Office of Science, under contract DE-AC02-06CH11357.

References

[1] VanMSFT, Ledger overview - sql server.
URL https://learn.microsoft.com/en-us/sql/relational-databases/security/ledger/ledger-overview?view=sql-server-ver17

[2] U. Schwinn, Oracle database native blockchain and immutable tables (Mar 2024).
URL https://blogs.oracle.com/coretec/post/blockchain-or-immutable-tables

[3] Pgaudit, Pgaudit/readme.md at main · pgaudit/pgaudit.
URL https://github.com/pgaudit/pgaudit/blob/main/README.md

[4] Monitoring amazon aurora with database activity streams.
URL https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/DBActivityStreams.html

[5] Introduction to the db2 audit facility.
URL https://www.ibm.com/docs/en/db2/11.5.x?topic=activities-introduction-db2-audit-facility

[6] Ibm guardium.
URL https://www.ibm.com/products/guardium

[7] P. Chopade, V. Pachghare, A review on database forensics research for last ten years, Procedia Computer Science 152 (2019) 431–438.

[8] A. Al-dhaqm, S. Abd Razak, S. H. Othman, A. Ali, F. A. Ghaleb, A. S. Rosman, N. Marni, Database forensic investigation process models: A review, IEEE Access 8 (2020) 48477–48490. doi:10.1109/ACCESS.2020.2976885.

[9] A. A. Alhussan, A. Al-Dhaqm, W. M. S. Yafooz, A.-H. M. Emara, S. Bin Abd Razak, D. S. Khafaga, A unified forensic model applicable to the database forensics field, Electronics 11 (9) (2022). doi:10.3390/electronics11091347.
URL https://www.mdpi.com/2079-9292/11/9/1347

[10] B. Lenard, L. Smith, J. Wagoner, Sysgen: System state corpus generator for database forensic benchmarking, Journal of Digital Forensics, Security and Law 15 (2) (2020) 85–98.

[11] R. T. Snodgrass, S. S. Yao, C. Collberg, Tamper detection in audit logs, in: Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30, VLDB '04, VLDB Endowment, 2004, p. 504–515.

[12] P. Frühwirt, P. Kieseberg, S. Schrittwieser, M. Huber, E. Weippl, Innodb database forensics: Enhanced reconstruction of data manipulation queries from redo logs, Information Security Technical Report 17 (4) (2013) 227–238, special Issue: ARES 2012 7th International Conference on Availability, Reliability and Security. doi:https://doi.org/10.1016/j.istr.2013.02.003.
URL https://www.sciencedirect.com/science/article/pii/S1363412713000137

[13] J. Shin, Memory-driven forensic analysis of sql server: A buffer pool and page inspection approach, Sensors 25 (11) (2025). doi:10.3390/s25113512.
URL https://www.mdpi.com/1424-8220/25/11/3512

[14] H. Choi, S. Lee, Forensic analysis of sql server transaction log in unallocated area of file system, Forensic Science International: Digital Investigation 46 (2023) 301605. doi:https://doi.org/10.1016/j.fsidi.2023.301605.
URL https://www.sciencedirect.com/science/article/pii/S2666281723001178

[15] M.-J. Yoon, S.-H. Kim, J.-T. Kim, Forensic investigation framework for mongodb nosql database, in: Proceedings of the International Conference on Digital Forensics and Cyber Crime, 2016.

[16] R. Nemetz, M. Ruff, M. Huber, The sqlite forensic corpus: Benchmarking forensic tools with standardized datasets, Digital Investigation 26 (2018) 40–50.

[17] D. Nissan, K. Rossi, M. Cho, Anoc: Automated nosql database carver for forensic data recovery, Forensic Science International: Digital Investigation (2025).

[18] J. Wagner, A. Rasin, J. Grier, Database forensic analysis through internal structure carving, Digital Investigation 14 (2015) S106–S115, the Proceedings of the Fifteenth Annual DFRWS Conference. doi:https://doi.org/10.1016/j.diin.2015.05.013.

[19] J. Wagner, et al., Database image content explorer: Carving data that does not officially exist, in: DFRWS, 2016.

[20] B. Lenard, A. Rasin, N. Scope, J. Wagner, What is lurking in your backups?, Springer International Publishing, 2021, p. 401–415.

[21] W. W. Hsu, A. J. Smith, H. C. Young, Characteristics of production database workloads and the tpc benchmarks, IBM Systems Journal 40 (3) (2001) 781–802.