Advanced Forensic Analysis Techniques for Insider Threat Detection in Database Management Systems

Kholod Saeed Talea AlQahtani, Mounir Frikha, M. M. Hafizur Rahman
Department of Computer Networks and Communications-College of Computer Sciences and Information Technology, King Faisal University, Al-Ahsa, Saudi Arabia

***Abstract*—This paper presents a real-time database forensic framework designed to detect insider threats within database management systems (DBMSs). Existing database forensic approaches, as identified through a systematic literature review employing the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) methodology, are predominantly reactive and post-mortem in nature, lacking real-time SQL-layer visibility, forensic correlation, and evidence integrity assurance. To address these gaps, the proposed framework integrates native Microsoft SQL Server auditing mechanisms with a centralized ELK (Elasticsearch-Logstash-Kibana) pipeline to enable continuous evidence collection, automated correlation, and real-time visualization. The framework is evaluated against six insider-threat scenarios within the SPL ForensicDB—a simulated enterprise logistics database environment modeled after Saudi Post Logistics (SPL)—encompassing unauthorized access, data exfiltration, privilege escalation, data manipulation, backdoor creation, and audit suppression. Experimental results demonstrate high detection accuracy (precision: 0.92, recall: 0.88), a low false-positive rate of 3%, and alert latency consistently below five seconds, with minimal system overhead of 3.2% CPU utilization. The framework further ensures forensic integrity through SHA-256-verified, tamper-resistant audit logs and a structured chain-of-custody preservation mechanism compliant with ISO/IEC 27037:2012, making it suitable for both proactive security monitoring and legally defensible digital forensic investigations.**

Keywords*—Database management systems; insider threat detection; SQL server; forensic auditing; ELK stack; digital forensics

I. Introduction

The rapid development of digital technologies has fundamentally transformed how organizations store, manage, and secure data. A Database Management System (DBMS) is a software solution that enables users and applications to manage, retrieve, store, organize, and create data efficiently [1]. It serves as an interface between the database and its users, ensuring secure, consistent, and accurate information handling. DBMSs manage diverse user populations through concurrency control, data protection via security and integrity constraints, and backup-and-recovery mechanisms. Prominent examples include MySQL, Oracle, PostgreSQL, and Microsoft SQL Server [2].

The use of DBMSs has become central to contemporary organizations as the volume and sensitivity of stored information continue to grow. As data-driven operations have become the norm, the security and integrity of database environments have emerged as critical priorities. Insider threats—malicious or negligent actions by individuals with authorized access—are currently among the most harmful and difficult-to-trace categories of cyberattack [3][20]. According to the IBM Cost of a Data Breach Report [19], insider-related incidents account for a disproportionate share of total breach costs relative to external attacks, primarily because insiders operate within trusted boundaries and possess legitimate credentials.

This paper seeks to develop, deploy, and evaluate a real-time SQL-layer forensic auditing and insider-threat detection system built on Microsoft SQL Server, combining its native auditing capabilities with centralized log correlation and alerting. SQL Server is among a limited number of commercial DBMS platforms satisfying three core forensic requirements: tamper resistance, forensic auditability, and chain-of-custody preservation [4]. Research on database forensic readiness has shown that open-source DBMSs such as MySQL and PostgreSQL cannot support forensics without third-party extensions (e.g., pgaudit) [5], which cannot be considered tamper-proof or capable of preserving a defensible chain of custody essential for admissible forensic evidence. SQL Server, by contrast, provides SQL Audit and Extended Events, which produce binary, append-only, tamper-resistant .sqlaudit files satisfying the validation requirements of modern forensic audit research [6].

A. Research Gap and Motivation

Database-layer insider threats represent one of the most insidious forms of cyberattack because malicious insiders operate with legitimate credentials and SQL-level privileges that bypass conventional network- and OS-level security controls. Current database forensic solutions are predominantly post-mortem, disjointed, or based on outdated tracing mechanisms, rendering them ineffective at providing real-time visibility, evidence integrity assurance, or scalability to enterprise contexts. This paper addresses a specific and identified research gap: the absence of an authenticated, real-time, SQL-layer forensic pipeline capable of simultaneously supporting insider-threat detection, evidentiary integrity, and low operational burden within a production-scale database environment.

B. Novel Contribution

This paper proposes and empirically validates a unified SQL-layer forensic audit framework combining SQL Server Audit, Extended Events, and an ELK (Elasticsearch-Logstash-Kibana) pipeline to enable near-real-time insider-threat detection, correlation, and visualization. The contribution lies not in describing individual system components, but in demonstrating that native SQL Server mechanisms can be operationalized into a single forensic-ready pipeline capable of delivering:

- Real-time insider-threat detection at the SQL execution layer.
- Cryptographically verifiable evidence integrity via SHA-256 hashing.
- Performance overhead sustainable within enterprise production environments.

The remainder of this paper is organized as follows. Section II presents a systematic literature review of related work in database forensic auditing and insider-threat detection. Section III describes the experimental methodology, including the simulated environment configuration and attack scenario design. Section IV presents detection effectiveness and validation metrics. Section V addresses forensic integrity and evidence readiness. Section VI discusses operational and enterprise implications. Section VII summarizes the scientific contributions. Section VIII provides a comparative discussion of findings in relation to prior research. Section IX concludes the paper and outlines directions for future work.

II. Literature Review

A. Database Forensics and SQL-Layer Auditing

Research in database forensic auditing has evolved considerably, with scholars recognizing that classic OS-level artefacts are insufficient for investigating advanced insider attacks [6]. Early implementations relied on OS logs and general event logs, which lack the granularity necessary to reconstruct unauthorized DML operations, privilege escalations, or schema modifications at the SQL layer. Early research on log management [7] highlights that many trace formats lack inherent tamper resistance, rendering them vulnerable to manipulation by insiders and potentially inadmissible as forensic evidence. An alternative perspective based on external monitoring agents and third-party auditing platforms [8] introduces deployment complexity and may fail to capture engine-level execution traces.

Early work [9] identified that malicious activity at the DBMS layer frequently leaves implicit artefacts in transaction logs, cache buffers, and session metadata. However, these early contributions were largely confined to post-mortem analysis, permitting investigators to reconstruct incidents only after operational or financial damage had occurred. Khanuja and Adane [17] proposed a database forensic analysis framework that systematized the collection and analysis of SQL-layer artefacts, while Wagner [18] demonstrated DBMS forensic auditing using Extended Events (XEvents), establishing a technical precedent for the SQL-layer pipeline approach adopted in the present work. Wagner et al. [9] extended this research by examining database memory forensics and cache-pattern analysis for log verification, confirming that in-memory artefacts supplement binary audit logs in forensic reconstruction.

Studies on insider behaviour consistently report that such attacks unfold slowly and quietly, blending into routine operational tasks [10]. Behavioural profiling research [4][11] shows that insiders typically exhibit patterns such as accessing information beyond their assigned role, exporting sensitive data in bulk, or engaging in unusual after-hours activity. However, most proposed models analyse these behaviours only after the incident has occurred, relying on exported log files rather than real-time event streams [22][15]. This retrospective limitation creates a substantial detection gap, compelling investigators to reconstruct events after the fact rather than intervening during the active attack window.

Wei et al. [15] proposed an unsupervised anomaly detection scheme applied to database logs for proactive forensic investigation, demonstrating effectiveness on insider-threat scenarios; however, that approach lacked real-time pipeline integration and did not address cryptographic evidence integrity. Similar deep learning-based approaches for system log anomaly detection have also been explored in [24], further highlighting the potential of advanced analytics in forensic log analysis. Alzaabi and Mehmood [13] conducted a comprehensive review of machine learning methods for insider-threat detection, identifying precision and recall as primary evaluation metrics and underscoring the persistent challenge of high false-positive rates in anomaly-based systems. Manral and Somani [16] examined forensic capabilities in the presence of superuser insider threats, highlighting the critical importance of tamper-resistant logging mechanisms that cannot be disabled by privileged insiders—a requirement directly addressed by the SQL Audit binary format in the present study. Henriques et al. [14] surveyed forensics and compliance auditing for critical infrastructure, identifying ELK-based pipelines as among the most promising approaches for real-time log correlation and forensic visualization. Pachghare and Chopade [23] conducted a critical review of database forensics, identifying performance overhead as a key practical barrier to continuous auditing adoption, with enterprise acceptability generally requiring CPU overhead below 5%.

The theoretical foundation of this research rests on the Digital Forensic Readiness concept, which stipulates that systems should be designed to collect, preserve, and analyse evidence proactively rather than reactively [12]. Collectively, the reviewed literature—surveyed using the PRISMA methodology—supports the development of a unified SQL-based forensic pipeline combining real-time monitoring, automated correlation, and cryptographic evidence preservation.

B. Commercial Database Security Systems: A Comparative Overview

In addition to academic approaches, several commercial database security and auditing solutions have been widely adopted in enterprise environments. IBM Guardium and Imperva SecureSphere primarily rely on agent-based monitoring and network-level traffic inspection to detect suspicious database activities in real time [22]. While these systems provide strong detection capabilities, they often introduce substantial deployment complexity and may lack fine-grained forensic reconstruction at the SQL execution level due to their reliance on network-layer rather than engine-layer event capture.

Oracle Audit Vault focuses on centralized log collection and compliance auditing, offering a consolidated view of database activity across heterogeneous environments. However, its detection capabilities are generally less responsive in real-time attack scenarios compared to dedicated monitoring pipelines, and it does not natively provide cryptographic chain-of-custody mechanisms at the individual file level. In contrast, the proposed framework leverages native SQL Server auditing mechanisms combined with an ELK-based pipeline to provide near-real-time detection with minimal system overhead. The emphasis on tamper-resistant audit logs and SHA-256 cryptographic validation enables both operational monitoring and legally defensible evidence preservation, distinguishing the proposed approach as a lightweight, forensically robust, and practically deployable alternative within enterprise database environments [14].

III. Methodology

This research adopts a technical experimental approach to design and validate a SQL forensic auditing architecture using Microsoft SQL Server 2019 Enterprise. The experimental environment was constructed to simulate the operational characteristics of Saudi Post Logistics (SPL), incorporating realistic customer, financial, and audit datasets. A specialized forensic database—SPL ForensicDB—was designed to execute scripted insider attacks and capture the resulting evidence traces. The overall system architecture, as illustrated conceptually in Fig. 1, encompasses three phases: data collection at the SQL layer, processing and normalization via Logstash, and real-time alerting and visualization through Elasticsearch and Kibana.

The binary format of SQL Audit records enabled the use of sys.fn_get_audit_file() to transform audit streams into downstream-readable relational tables. This normalization step verifies the semantic consistency of logs prior to ELK pipeline ingestion. Filebeat was configured to continuously scan .sqlaudit and .xel directories and forward new events to Logstash, which performed parsing, enrichment, timestamp validation, and field normalization before indexing events in Elasticsearch. Kibana provided visualization dashboards enabling analysts to identify anomalies including bulk data extraction, repeated

authentication failures, suspicious privilege usage, and attempts to disable the auditing engine. The logical flow of the system is depicted in Fig. 2 and the overall architecture in Fig. 3.

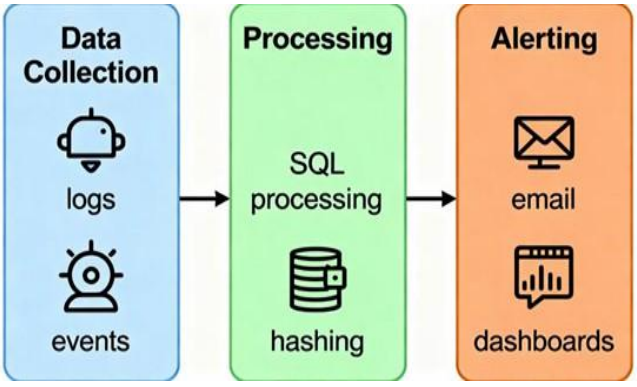

Fig. 1. Conceptual framework architecture showing data collection, processing, and alerting phases.

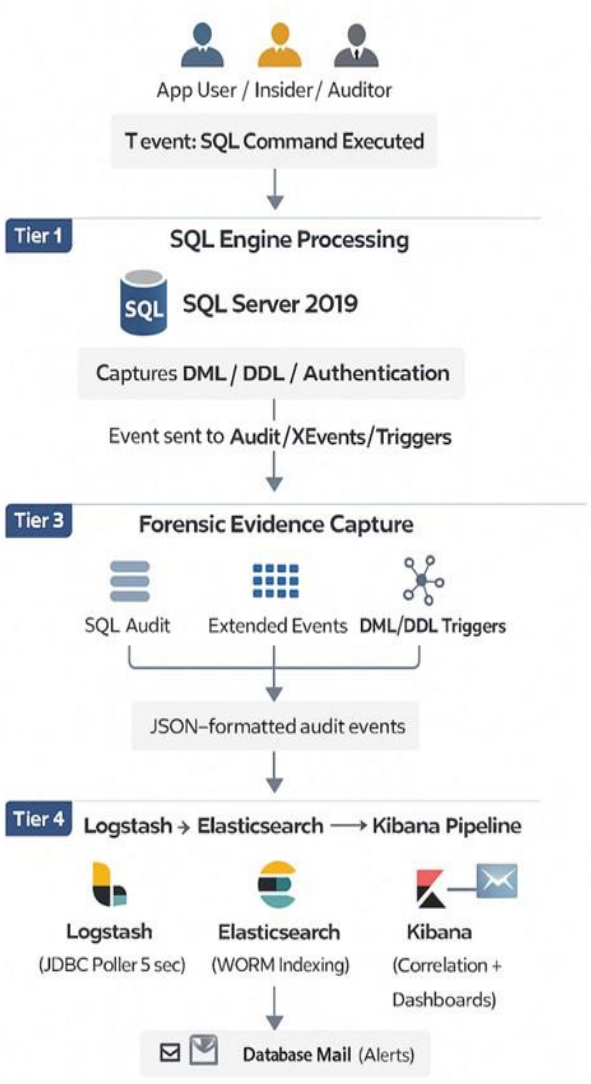

Fig. 2. Logical flow diagram.

Fig. 3. Forensic audit framework (system architecture).

Six insider-attack scenarios were scripted in T-SQL to evaluate the framework, as described in Table I. Performance was measured across five dimensions: precision, recall, false-positive rate, alert latency, and system performance overhead.

A. Experimental Environment Configuration and Log Sources

The framework was evaluated within a simulated enterprise operational environment modeled on Saudi Post Logistics (SPL) and running on Microsoft SQL Server 2019 Enterprise. The testbed incorporated realistic customer, financial, and audit data; RBAC structures; and controlled insider behaviour. Assessment employed a controlled experimental design in which insider attacks were executed as scripted T-SQL operations and evidence was ingested into the ELK pipeline from audit, trace, and trigger-based sources. The SQL Server instance was configured with secure authentication, structured log directories, and native audit mechanisms (Fig. 4). A dedicated SQL authentication login (SQLCONNECT) was created for forensic testing, with Mandatory Encryption enforced at the connection level.

System databases and the SPL ForensicDB are hosted within the instance (Fig. 5). SQL Server records system-level events and authentication information in the default ERRORLOG directory (Fig. 6). All forensic audit files generated by SQL Server Audit and Extended Events were stored in a dedicated directory at C:\ForensicAuditing\SQLAudit\ (Fig. 7). Two audit objects were

defined (Fig. 8): AuditServerToFile, capturing server-level events, and ForensicServerAudit, capturing database-level DDL, DML, failed logins, and privilege changes. These audits form the native SQL evidence stream prior to ELK ingestion.

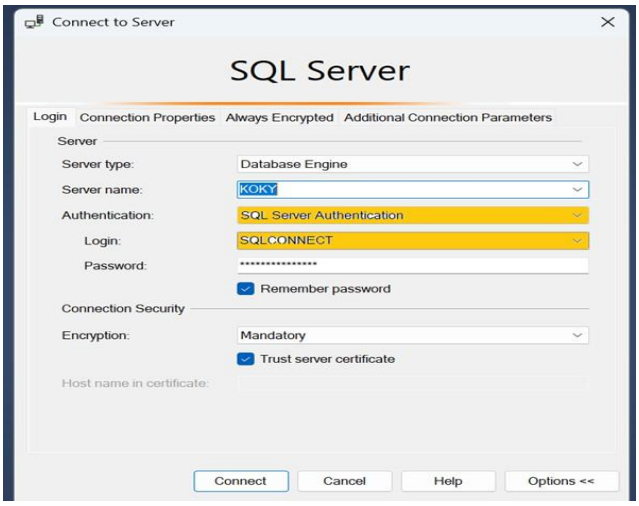

Fig. 4. SQL Server connection configuration showing SQL authentication and mandatory encryption.

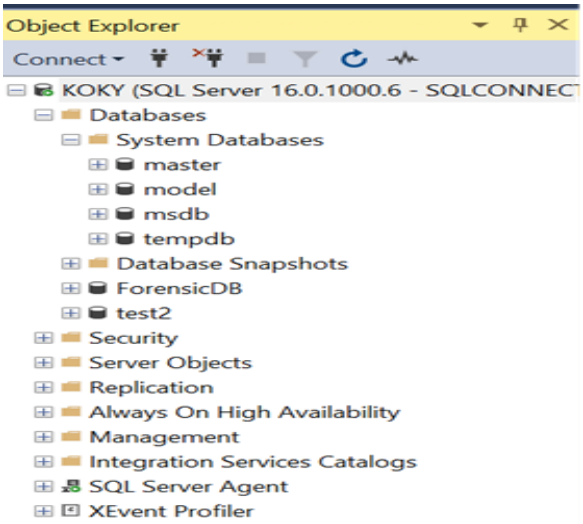

Fig. 5. SQL Server object explorer showing system databases and the forensicdb environment.

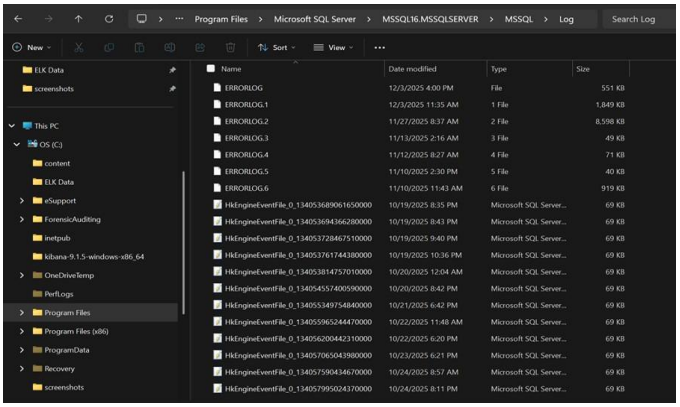

Fig. 6. Default SQL Server error log directory showing rotating ERRORLOG files.

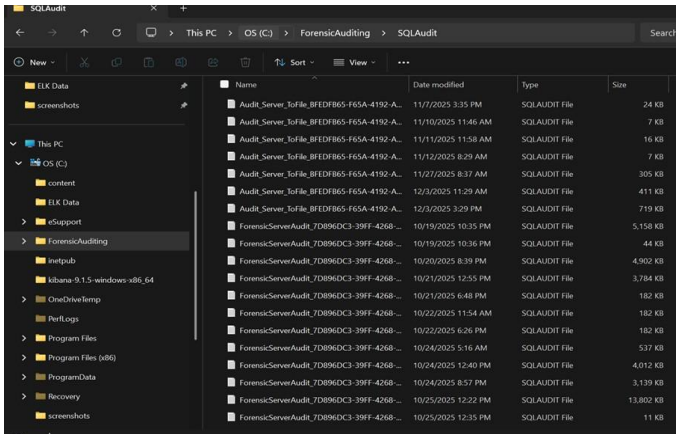

Fig. 7. SQL Audit directory containing generated .sqlaudit binary evidence files.

Fig. 8. Configured SQL Server Audits used for forensic event capture.

B. Simulated Database Environment: SPL ForensicDB

To ensure practical relevance, the SPL ForensicDB was constructed to mirror the schema complexity of a logistics and financial enterprise system. The Entity-Relationship Diagram (ERD) in Fig. 9 consists of four core modules:

- Human Resources Module: Employees (EmpID, Name, Department, ClearanceLevel); Users (UserID, Username, PasswordHash, RoleID).
- Customer Data Module: Customers (CustID, NationalID, Phone, Address, RiskScore)—containing PII data targeted in unauthorized-access and exfiltration scenarios.
- Financial Module: Transactions (TransID, Amount, Date, Sender, Receiver)—targeted for data manipulation scenarios.
- Audit Module: AuditLogMaster (LogID, EventType, TableName, OldValue, NewValue, ModifiedBy, Timestamp)—preserving before/after forensic attribution records via DDL/DML triggers.

The modules collectively replicate the real-world interdependencies between HR data, customer PII, and financial operations within SPL, ensuring that simulated attack scenarios are representative of genuine enterprise threat conditions.

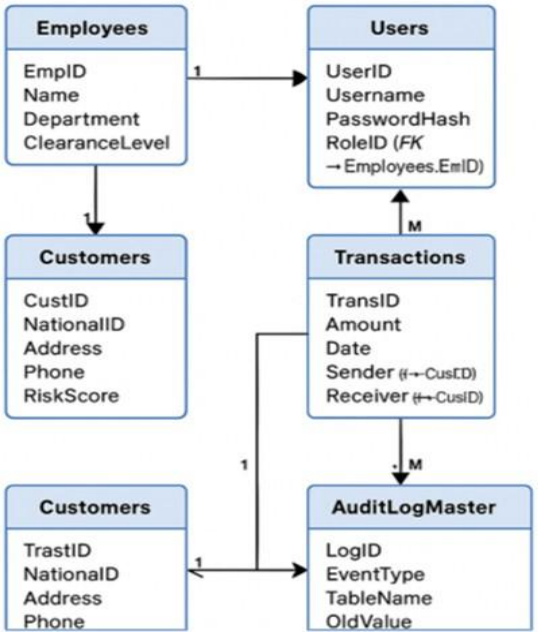

Fig. 9. ERD Relationship Diagram

C. Access Control and User Roles

To simulate realistic insider threats, the following user roles were defined:

- SysAdmin (SA): Full control, simulating a compromised administrator.
- DB Auditor: Read-only access to audit logs, representing the forensic investigator role.
- App User: Limited access via stored procedures, representing a normal employee.
- Malicious Insider: A user with db_datareader and db_datawriter privileges attempting unauthorized privilege escalation.

D. Insider-Threat Scenario Coverage

As summarized in Table I, six distinct insider-threat scenarios were designed to represent realistic and high-impact attack categories in enterprise database environments. Each scenario was engineered to test both detection accuracy and forensic completeness, ensuring that multilayered evidence could be reconstructed across SQL Audit logs, Extended Events traces, and DML/DDL triggers.

TABLE I. INSIDER THREAT SCENARIOS

Scenario ID	Insider-Threat Category	Description
S1	Unauthorized data access	Illicit reading of sensitive PII records
S2	Data exfiltration	Bulk extraction of customer data
S3	Privilege escalation	Unauthorized role and permission abuse
S4	Data manipulation	Financial record tampering
S5	Backdoor creation	Covert SQL login creation
S6	Audit suppression	Attempts to disable or tamper with audit logging

IV. DETECTION EFFECTIVENESS AND VALIDATION METRICS

Detection performance was quantitatively evaluated across all six scripted insider-threat scenarios. The framework successfully identified all six attack scenarios, with performance results summarized in Table II. These metrics confirm that the framework achieves high accuracy in detecting insider activity within the active attack window, enabling a timely forensic response without compromising production database performance. A post-execution review of detection logic, illustrated in Fig. 10, confirmed that all six attack sequences generated correlated alerts within the defined latency threshold, and no scenario evaded the pipeline.

TABLE II. SUMMARY OF DETECTION METRICS

Metric	Result	Forensic Meaning
Precision	0.92	Low false-alarm rate; investigators only review relevant alerts
Recall	0.88	High ability to detect real insider actions
False Positive Rate	3%	Minimizes alert fatigue and improves trust in system out-put
Median Alert Latency	2–3 sec	Enables near–real-time investigation of active insider at-tacks
CPU Overhead	3.2%	Meets <5% enterprise audit requirement without affecting production

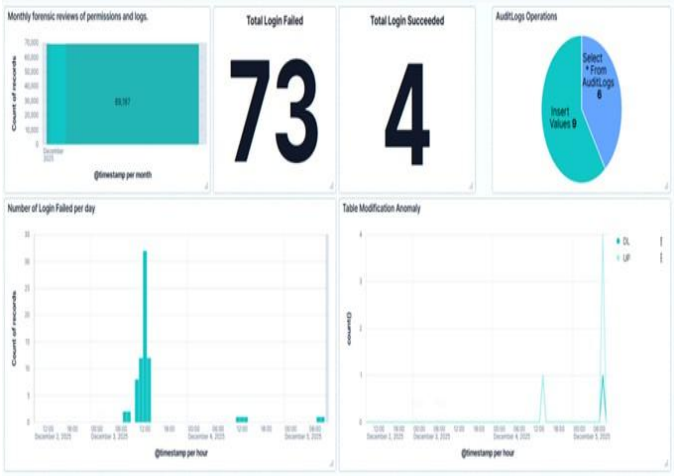

Fig. 10. Dashboard reflecting SQL audit events after malicious insider actions

V. FORENSIC INTEGRITY AND EVIDENCE READINESS

Beyond detection capability, the framework incorporates database-layer forensic readiness principles extending its utility to legally defensible evidence preservation. Upon acquisition, each evidence file was immediately processed to generate a SHA-256 cryptographic hash, consistent with the evidence integrity requirements of ISO/IEC 27037:2012 and the forensic integrity controls documented by Bush [6]. This process ensures that any post-acquisition tampering can be cryptographically detected, establishing an unbroken chain of custody from initial collection through analysis. The audit artefacts are stored as tamper-resistant binary .sqlaudit files with SHA-256 verification, NTP-synchronized timestamps, and WORM-like immutable indexing within Elasticsearch.

Table III presents the chain-of-custody validation results across all six evidence scenarios, confirming that all artefacts were successfully hashed, timestamp-synchronized, and alert-delivered within the defined latency bounds. Notably, Scenario S6 (Audit Suppression) validates the system's resilience against the most adversarial insider condition: an

attempt to disable auditing itself. The immutable Elasticsearch index confirmed in Table III for S6 demonstrates that even a forced audit engine shutdown generates a detectable and tamper-evident forensic record, addressing a critical gap identified by Manral and Somani [16]. This end-to-end evidence integrity makes the system applicable to legal and regulatory investigations—a requirement increasingly mandated under data governance frameworks [21].

TABLE III. CHAIN-OF-CUSTODY VALIDATION FOR INSIDER TEST EVIDENCE

Scenario ID	Evidence Captured	SHA-256 Ver-ified	Timestamp Synced	Output
S1	Customers read-perimeter breach	Yes	Yes	Email logged & delivered 3s
S2	Temp file exfiltration vector	Yes	Yes	Threshold breach alert 5s
S3	Unauthorized ALTER ROLE attempt	Yes	Yes	Role abuse visual flagged in Kibana
S4	UPDATE manipulation anomaly	Yes	Yes	0-value override detected & alerted
S5	Hidden login creation path	Yes	Yes	Backdoor login indexed & monitored
S6	Audit engine forced OFF	Yes	Yes	Immutable index lock confirmed in ELK

VI. OPERATIONAL AND ENTERPRISE IMPLICATIONS

The results confirm the operational feasibility of SQL-layer forensic auditing in enterprise settings. The co-existence of real-time monitoring with production workloads—demonstrated through a CPU overhead of 3.2%, well within the 5% enterprise acceptability threshold—substantially reduces the traditional trade-off between forensic comprehensiveness and operational performance. Centralized ELK-based visualization through Kibana (Fig. 10 and Fig. 11) enables analysts to rapidly identify anomalous patterns across temporal and behavioral dimensions, significantly reducing investigation time and analyst workload.

In practice, the framework provides organizations with an executable deployment roadmap for insider-threat detection, enhanced forensic preparedness, and improved accountability within database-centric infrastructures. The tamper-resistant binary audit format and SHA-256 chain-of-custody mechanism address a critical operational gap: the inadmissibility of forensic evidence from systems lacking verifiable integrity controls [7][16]. The framework thereby bridges the gap between security monitoring—a real-time operational function—and digital forensics—a post-incident legal function—within a single integrated pipeline.

VII. SUMMARY OF SCIENTIFIC CONTRIBUTION

The principal scientific contribution of this work lies in the empirical operationalization of a fully integrated SQL-layer forensic pipeline that bridges three previously disparate capabilities: real-time insider-threat detection, cryptographically verifiable evidence integrity, and sub-five-second alert latency within a production-representative database environment. Prior research has addressed these capabilities predominantly in isolation, limiting the practical applicability of individual contributions.

Database forensic research [17][18] demonstrated the evidential value of SQL-layer audit logs but did not operationalize them into real-time detection pipelines. Behavioural profiling studies [4][11] characterized insider-threat patterns effectively but lacked evidentiary integrity mechanisms required for legal admissibility. Evidence-integrity frameworks [7] established chain-of-custody methodologies but did not demonstrate integration with live SQL-layer detection at enterprise scale. The present work uniquely integrates all three capabilities within a single deployable pipeline and validates their simultaneous operation under realistic attack conditions.

The achievement of 3.2% CPU overhead during active forensic monitoring is particularly significant, empirically demonstrating that continuous SQL-layer auditing is operationally sustainable without dedicated hardware or substantial performance trade-offs—directly addressing the adoption barrier identified by Pachghare and Chopade [23]. The SHA-256 chain-of-custody mechanism, compliant with ISO/IEC 27037:2012, provides a legally defensible evidence trail that extends the system's applicability from operational security monitoring to formal digital forensic investigations. These combined properties represent a measurable advancement beyond descriptive system overviews and theoretical frameworks in the existing database forensics literature [5][12].

VIII. DISCUSSION

The findings confirm that the developed framework reliably captures insider activity at the SQL layer and provides forensically actionable insight. The framework successfully detected all six scripted insider-threat scenarios, capturing multilayered forensic evidence across SQL Audit logs, Extended Events traces, and DML/DDL triggers, with an overall precision of 0.92 and a recall of 0.88. The false-positive rate of 3% demonstrates that the detection logic effectively differentiates between genuine anomalies and normal operational activity. Alert latencies were consistently below five seconds across all scenarios, confirming near-real-time response capability sufficient for active-attack intervention.

The framework exhibited high forensic completeness. SQL Server Audit captured all applicable DDL, DML, and authentication events; Extended Events recorded engine-level data including bulk reads, query execution patterns, and performance anomalies; DML/DDL triggers preserved before-and-after values enabling precise forensic reconstruction of unauthorized changes. Kibana visualizations (Fig. 10 and Fig. 11) identified temporal spikes and abnormal access behaviours precisely corresponding to the simulated attack scenarios. Performance testing confirmed average CPU overhead of 3.2%, validating the feasibility of continuous forensic auditing in enterprise deployments and demonstrating the efficiency advantage of Extended Events over legacy SQL Trace mechanisms.

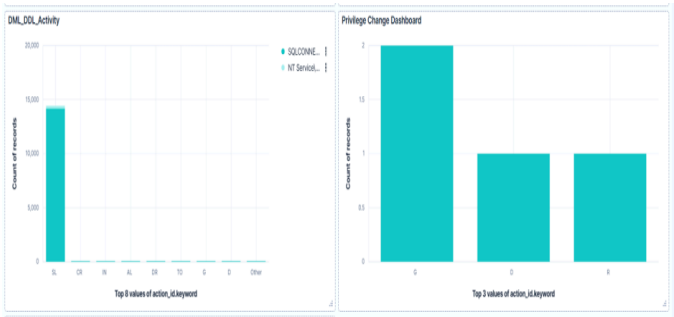

Fig. 11. Kibana privilege-change dashboard showing the impact of GRANT operations.

These findings are consistent with, and advance, prior contributions in the field. Wei et al. [15] demonstrated that unsupervised anomaly detection on database logs can support proactive forensic investigation; however, that approach lacked real-time pipeline integration and did not address cryptographic evidence integrity. The present framework advances this line of work by integrating detection with evidence preservation in a single operational pipeline. Manral and Somani [16] established the importance of forensic capabilities resilient to superuser manipulation—a property directly addressed through the tamper-resistant binary .sqlaudit format, which cannot be modified or deleted by privileged users without generating a detectable forensic artefact.

The false-positive rate of 3% compares favorably with behavioral profiling approaches reviewed by Alzaabi and Mehmood [13], which reported consistently higher false-alarm rates attributable to the difficulty of distinguishing malicious from legitimate privileged activity in purely behavioral models. The sub-five-second alert latency represents a substantial improvement over post-mortem forensic approaches [5][12], which offer no real-time intervention capability by definition. The 3.2% CPU overhead is consistent with—and below—the enterprise feasibility threshold identified by Pachghare and Chopade [23], who cited performance overhead as a primary barrier to continuous database auditing adoption. Henriques et al. [14] reached similar conclusions regarding the suitability of ELK-based pipelines for critical infrastructure forensics, further corroborating the architectural choices made in this work.

Compared with commercial solutions, the proposed framework offers complementary and, in several respects, superior properties for forensic use cases. Unlike IBM Guardium and Imperva SecureSphere, which rely on network-layer inspection and introduce substantial deployment overhead [22], the proposed system leverages native SQL Server engine mechanisms, minimizing infrastructure requirements while achieving greater SQL execution-level granularity. Unlike Oracle Audit Vault, which aggregates compliance logs rather than providing real-time detection, the ELK-based pipeline delivers sub-five-second alerting with a full forensic chain-of-custody. These distinctions position the framework as a practically deployable, forensically rigorous alternative suitable for resource-constrained enterprise environments seeking both security monitoring and legal evidentiary capabilities within a single system.

IX. Conclusion

This study has formulated, deployed, and thoroughly evaluated a SQL-layer forensic auditing framework that integrates SQL Server Audit, Extended Events, and an ELK-based evidence correlation pipeline. The analysis demonstrates that the proposed system achieves its design objectives across all evaluated dimensions: real-time insider-threat detection, cryptographic evidence integrity, and operational performance sustainability. The framework successfully detected all six scripted insider-threat scenarios with a precision of 0.92, a recall of 0.88, a false-positive rate of 3%, and alert latency consistently below five seconds, while maintaining CPU overhead of only 3.2%.

In addition to meeting functional and performance objectives, this work contributes to the theoretical understanding of how forensic readiness, behavioural analysis, and SQL-layer observability can be integrated into a coherent, production-deployable pipeline. It demonstrates that native database auditing mechanisms, when properly operationalized with modern analytics technologies, can deliver scalable insider-threat monitoring while satisfying the forensic integrity standards required for legal proceedings.

Future work will explore the extension of the framework to heterogeneous DBMS environments, the incorporation of machine learning-based anomaly scoring for adaptive threshold detection, and the addition of automated containment responses that not only detect insider attacks but also execute predefined countermeasures. Integration with Security Information and Event Management (SIEM) platforms and compliance with evolving data governance frameworks represent additional directions for broadening the framework's applicability.

Funding

This work was funded by King Faisal University, Al-Ahsa, Saudi Arabia KFU261953.

Acknowledgments

This work was supported through the Annual Funding track by the Deanship of Scientific Research, Vice Presidency for Graduate Studies and Scientific Research, King Faisal University, Al-Ahsa, Saudi Arabia, under Project No. KFU261953.

Conflicts of Interest

All authors declare no conflict of interest.

References

[1] K. D. Malakar, S. Roy, and M. Kumar, "Database management system: Foundations and practices," in Geospatial Technologies in Coastal Ecologies Monitoring and Management, Cham, Switzerland: Springer Nature, Jul. 2025, pp. 191–255.

[2] A. Tezuysal and I. Ahmed, Database Design and Modeling with PostgreSQL and MySQL: Build Efficient and Scalable Databases for Modern Applications Using Open Source Databases. Birmingham, U.K.: Packt Publishing, Jul. 2024.

[3] M. M. Alani, "Big data in cybersecurity: A survey of applications and future trends," Journal of Reliable Intelligent Environments, vol. 7, no. 2, pp. 85–114, 2021.

[4] P. Hakonen, "Insider threat detection using behavior analytics," in Detecting Insider Threats. Cham, Switzerland: Springer, 2022.

[5] A. Al-Dhaqm et al., "Digital forensics subdomains: The state of the art and future directions," IEEE Access, vol. 9, pp. 152476–152502, Oct. 2021.

[6] J. Bush, Practical Database Auditing for Microsoft SQL Server and Azure SQL. New York, NY, USA: Apress, 2022, ISBN: 978-1-4842-8633-3. DOI: 10.1007/978-1-4842-8634-0.

[7] E. N. Chukwuani and C. D. Ikemefuna, "Blockchain-based chain-of-custody models for tamper-proof evidence preservation in digital forensics investigations," 2023.

[8] A. M. Adebowale and O. B. Akinnagbe, "Cross-platform financial data unification to strengthen compliance, fraud detection and risk controls," World Journal of Advanced Research and Reviews, vol. 20, no. 3, pp. 2326–2343, 2023.

[9] J. Wagner, M. I. Nissan, and A. Rasin, "Database memory forensics: Identifying cache patterns for log verification," Forensic Science International: Digital Investigation, vol. 45, Art. no. 301567, Jul. 2023.

[10] N. Catrantzos, Managing the Insider Threat: No Dark Corners and the Rising Tide Menace. Boca Raton, FL, USA: CRC Press, Nov. 2022.

[11] O. K. Emmanuel, J. Aria, D. Jose, and C. Diego, Data-Driven Cybersecurity: How Analytics Can Predict and Prevent Insider Attacks, 2023.

[12] A. Selim and I. Ali, "The role of digital forensic analysis in modern investigations," Journal of Emerging Computer Technologies, vol. 4, no. 1, pp. 1–5, Dec. 2024.

[13] F. R. Alzaabi and A. Mehmood, "A review of recent advances, challenges, and opportunities in malicious insider threat detection using machine learning methods," IEEE Access, vol. 12, pp. 30907–30927, Feb. 2024.

[14] J. Henriques, F. Caldeira, T. Cruz, and P. Simoes, "A survey on forensics and compliance auditing for critical infrastructure protection," IEEE Access, vol. 12, pp. 2409–2444, Jan. 2024.

[15] Y. Wei, K. Chow, and S. Yiu, "Insider threat prediction based on an unsupervised anomaly detection scheme for proactive forensic

investigation," Forensic Science International: Digital Investigation, vol. 38, p. 301126, Oct. 2021.

[16] B. Manral and G. Somani, "Establishing forensics capabilities in the presence of superuser insider threats," Forensic Science International: Digital Investigation, vol. 38, p. 301263, Sept. 2021.

[17] H. K. Khanuja and D. S. Adane, "Database forensic analysis framework," Computer Science & Engineering International Journal, vol. 2, no. 3, pp. 27–41, 2012.

[18] J. Wagner, "DBMS forensic auditing via XEvents," in Proc. IEEE International Conference on Data Engineering (ICDE), 2018.

[19] IBM Corporation, Cost of a Data Breach Report 2024. IBM Security, 2024.

[20] N. F. Nassir, U. F. Rauf, Z. Zainol, and K. A. Ghani, "Revealing the multi-perspective factors behind insider threats in cybersecurity," Journal of Media and Information Warfare, vol. 17, pp. 65–82, Oct. 2024.

[21] B. Basic, P. Udovicic, and O. Orel, "In-database auditing subsystem for security enhancement," in Proc. 44th International Convention on Information, Communication and Electronic Technology (MIPRO), IEEE, Sept. 2021, pp. 1642–1647.

[22] G. B. Sadowski, K. K. Bimo, and T. Booth, "Critical capabilities for SIEM," Gartner Research Note, 2020.

[23] R. Pachghare and H. K. Chopade, "Critical review on database forensics," Digital Investigation, vol. 29, pp. 180–197, 2019.

[24] Z. Chen, J. Liu, W. Gu, Y. Su, and M. R. Lyu, "Experience report: Deep learning-based system log analysis for anomaly detection," arXiv preprint arXiv:2107.05908, Jul. 2021.