Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Security-Aware Semantic Validation Framework (SSVF): An Execution-Aware Semantic Computing Approach for Intelligent Relational Database Security

Kwame Addo-Buahing

nanagodspee@live.com

Ghana Communication Technology University

Research Article

Keywords: Relational Database Security, Execution Plan Analysis, Semantic Computing, Knowledge Graphs, Explainable Artificial Intelligence (XAI), Design Science Research

Posted Date: July 8th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10260089/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Title:
Security-Aware Semantic Validation Framework (SSVF): An Execution-Aware Semantic Computing Approach for Intelligent Relational Database Security

Authors & Affiliations
Kwame Addo-Buahing[1]*
[1] Faculty of Computing and Information Systems (FoCIS)
Ghana Communication Technology University (GCTU)
Accra, Ghana.

Abstract

Relational Database Management Systems (RDBMSs) are fundamental to modern digital infrastructures but remain vulnerable to sophisticated cyber threats, particularly SQL injection attacks that exploit application–database interactions to compromise data confidentiality, integrity, and availability. Existing database security approaches, including signature-based, machine learning, and deep learning techniques, primarily analyse query syntax or statistical features, providing limited capability to evaluate the underlying execution behaviour of database queries. This study proposes the **Security-Aware Semantic Validation Framework (SSVF)**, a novel execution-aware framework that transforms database execution plans into structured semantic knowledge for intelligent security validation.

Developed using **the Design Science Research (DSR)** methodology, SSVF integrates execution-plan analysis, ontology-driven semantic modelling, graph-based knowledge representation, behavioural validation, and explainable security reasoning within a unified formal architecture. The framework is formally specified through the transformation pipeline, **Q → E → G → V → D**, which converts SQL queries into semantic graph representations, validates execution behaviour against semantic constraints, and produces transparent security decisions. By treating execution semantics rather than query syntax as the primary source of security intelligence, SSVF enables the detection of anomalous database behaviour through semantic interpretation of execution plans.

The framework is theoretically evaluated in terms of semantic consistency, computational feasibility, explainability, and security effectiveness. The study contributes a unified semantic architecture, a formal computational model, and an explainable validation paradigm that establish a foundation for future implementation and empirical evaluation of intelligent relational database security systems.

Keywords: Relational Database Security; Execution Plan Analysis; Semantic Computing; Knowledge Graphs; Explainable Artificial Intelligence (XAI); Design Science Research.

1. Introduction

Relational Database Management Systems (RDBMSs) constitute the backbone of modern digital information infrastructures by enabling the storage, management, and retrieval of critical organizational data across domains including finance, healthcare, e-commerce, manufacturing, government, and cloud computing. As database technologies continue to evolve toward distributed, cloud-native, heterogeneous, and data-intensive environments, they have become increasingly valuable targets for sophisticated cyberattacks seeking to compromise data confidentiality, integrity, and availability. The rapid evolution of database architectures, including distributed SQL engines, cloud databases, and hybrid data platforms, has further expanded the attack surface, creating new security challenges that extend beyond traditional database protection mechanisms [1-5]. Consequently, safeguarding relational databases has become a fundamental challenge in Computer Science and cybersecurity, demanding innovative security mechanisms capable of addressing increasingly sophisticated and adaptive attack techniques.

Among the numerous threats targeting relational databases, SQL injection (SQLi) remains one of the most persistent and destructive attack vectors because it exploits vulnerabilities in application database interactions to manipulate SQL statements and gain unauthorized access to sensitive information. Despite decades of research, SQL injection continues to evolve alongside modern application architectures, cloud-native services, and AI-enabled software systems, making it a persistent security concern. Although secure coding practices, parameterized queries, web application firewalls, intrusion detection systems, and machine learning-based detection techniques have significantly improved database protection, most existing solutions primarily analyse query syntax, lexical patterns, or previously observed attack signatures. Consequently, these approaches often provide limited visibility into the underlying execution behaviour of database queries, reducing their effectiveness against sophisticated attacks that preserve syntactic correctness while producing malicious execution outcomes [1, 6-10].

To overcome this limitation, the execution plan generated by a relational database query optimiser provides a formal semantic representation of how the database engine intends to execute a SQL statement. It describes execution operators, access methods, join strategies, filtering operations, aggregation processes, and dependency relationships that collectively characterize the operational semantics of query execution. Unlike the textual representation of SQL statements, execution plans expose the behavioural properties of database processing, making them a valuable source of security intelligence for identifying abnormal execution patterns. Although execution plans have been extensively studied for query optimisation, cost estimation, and performance tuning, their potential as a foundation for proactive security validation remains largely unexplored in existing database security research [11-15].

Recent advances in semantic computing, ontology engineering, knowledge graphs, graph theory, and explainable artificial intelligence (XAI) have provided powerful mechanisms for representing, integrating, and reasoning over complex relational structures. Unlike conventional feature-based representations, semantic graph models preserve entities, semantic relationships, contextual knowledge, and structural dependencies, thereby enabling intelligent reasoning, behavioural validation, and explainable decision-making. These capabilities have demonstrated considerable promise across cybersecurity applications, including anomaly detection, attack graph analysis, cyber threat intelligence, and security knowledge representation. Moreover, recent developments in explainable AI have reinforced the importance of transparent decision-making in intelligent cybersecurity systems, particularly where security decisions must be auditable and interpretable. Their increasing adoption suggests significant potential for advancing intelligent database security through execution-aware semantic modelling and graph-based behavioural validation [16-22].

To address these limitations, this paper proposes the **Security-Aware Semantic Validation Framework (SSVF)**, a novel execution-aware framework for intelligent relational database security. The framework leverages database execution plans as semantic representations of query behaviour and integrates semantic modelling with behavioural validation to identify abnormal execution patterns before query execution. As illustrated in Figure 1, SSVF establishes a unified framework that combines execution-plan analysis, semantic representation, security validation, and explainable decision support, thereby providing a coherent architectural foundation for proactive, explainable, and intelligent database security.

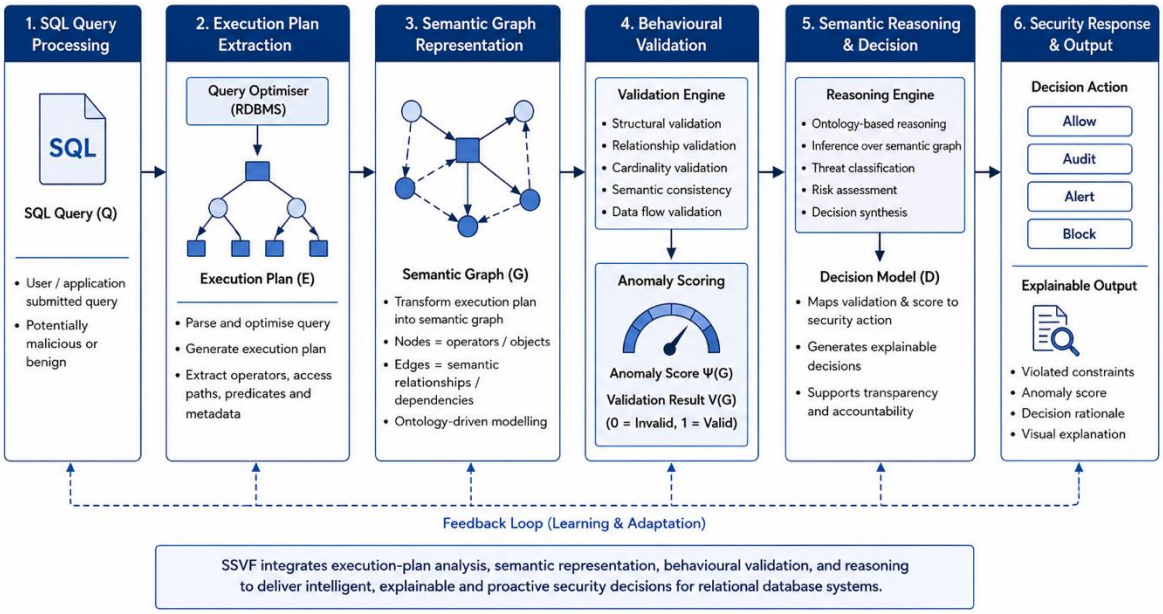


Fig. 1 Overall architecture of the proposed Security-Aware Semantic Validation Framework (SSVF), illustrating the integration of execution-plan analysis, semantic graph representation, semantic validation, behavioural validation, explainable reasoning, and intelligent security decision support

This study aims to develop a novel execution-aware semantic validation framework for intelligent relational database security by integrating execution-plan analysis, semantic computing, ontology engineering, graph-based knowledge representation, and behavioural validation within a unified Design Science Research (DSR) artefact. The proposed framework advances database security beyond conventional syntax-oriented detection by introducing execution semantics as the primary source of security intelligence. Its principal contributions include: (i) the design of the Security-Aware Semantic Validation Framework (SSVF); (ii) a unified architectural foundation for execution-aware semantic validation; (iii) a formal basis for integrating semantic representation, behavioural validation, and explainable security decisions; and (iv) a scalable architectural foundation for subsequent phases of the SSVF research programme.

The remainder of this paper is organized as follows. **Section 2** reviews the related literature and identifies the research gap that motivates this study. **Section 3** presents the Design Science Research methodology adopted to develop the proposed framework. **Section 4** introduces the Security-Aware Semantic Validation Framework (SSVF) and describes its architectural components. **Section 5** presents the formal specification of the framework, including its mathematical and semantic foundations. **Section 6** provides a theoretical evaluation of the proposed framework. **Section 7** discusses the findings and their implications for intelligent database security. Finally, **Sections 8, 9**, and **10** present the research contributions, limitations and future research directions, and the concluding remarks, respectively.

2. Related Work and Research Gap

2.1 Database Security and SQL Injection Detection

Relational database security has evolved from conventional access control mechanisms to intelligent defence strategies capable of addressing increasingly sophisticated cyber threats. Traditional security techniques including authentication, authorization, role-based access control (RBAC), encryption, and auditing provide essential protection for database resources but primarily enforce predefined security policies rather than analyse the operational behaviour of database queries during execution [1-4, 23]. As modern database environments become increasingly distributed, cloud-native, and application-driven, attackers continue to exploit vulnerabilities that bypass these preventive mechanisms, making behavioural analysis an essential component of database security.

Among database attacks, SQL injection (SQLi) remains one of the most critical threats because it enables adversaries to manipulate SQL statements, circumvent authentication mechanisms, and compromise sensitive information through malicious query execution. Existing countermeasures include secure coding practices, parameterized queries, static and dynamic code analysis, web application firewalls, intrusion detection systems, and, more recently, machine learning-based detection models [24-25, 10]. While these approaches have improved SQLi detection accuracy, they predominantly analyse lexical features, syntactic structures, or statistical patterns extracted from SQL statements rather than the semantic behaviour of query execution [6-7, 9, 26-27].

Recent artificial intelligence and deep learning approaches further enhance attack detection by learning behavioural characteristics from historical datasets. However, their effectiveness depends heavily on feature quality, labelled training data, and model generalisation, often limiting their ability to explain security decisions or adapt to previously unseen attack strategies [28-31]. Consequently, existing database security research remains largely syntax-oriented and data-driven, with limited emphasis on execution semantics as the basis for intelligent security validation. This limitation motivates the development of execution-aware semantic approaches capable of analysing database behaviour beyond query syntax, thereby providing a stronger foundation for explainable and proactive database security.

2.2 Semantic Computing and Graph-Based Security Analysis

Semantic computing has emerged as a significant paradigm for representing, integrating, and reasoning over complex data by explicitly modelling entities, relationships, and contextual knowledge. Within Computer Science, ontology engineering, knowledge graphs, and graph-based representations provide formal mechanisms for representing semantic dependencies that conventional relational or feature-based models often fail to preserve. These technologies have been widely adopted in knowledge representation, semantic web applications, intelligent information retrieval, and artificial intelligence because they support structured reasoning, semantic interoperability, and explainable decision-making [32-35, 17-18, 20].

In cybersecurity, graph-based models have demonstrated considerable effectiveness in representing attack paths, threat intelligence, network dependencies, and behavioural relationships. Attack graphs, knowledge graphs, and graph analytics enable security analysts to identify structural patterns, infer hidden relationships, and analyse complex interactions among system entities. Compared with traditional signature-based or statistical approaches, graph-based security analysis offers greater contextual awareness and supports explainable reasoning by preserving both structural and semantic relationships within the analysed environment [16, 36-38, 19].

Despite these advances, the application of semantic computing and graph-based knowledge representation to relational database security remains relatively limited. Existing

studies primarily focus on network security, cyber threat intelligence, and knowledge management, while comparatively little attention has been given to modelling database execution semantics as structured knowledge for intelligent security validation. Consequently, there remains a clear opportunity to integrate semantic computing, ontology engineering [32, 39-40], and graph-based reasoning into database security frameworks capable of analysing execution behaviour, identifying semantic anomalies, and supporting explainable security decisions.

2.3 Execution-Plan-Based Database Security

Database execution plans provide a formal representation of how a Relational Database Management System (RDBMS) executes SQL queries by specifying access paths, relational operators, join strategies, filtering predicates, and execution dependencies generated by the query optimiser. Traditionally, execution plans have been investigated within the fields of query optimisation, cost estimation, and performance tuning, where they support query analysis and optimisation by describing the physical execution strategy selected by the database engine [11, 41-42, 12-13].

More recently, several studies have explored execution plans as a source of behavioural information for anomaly detection, workload analysis, and database performance monitoring [43-44]. These investigations demonstrate that execution plans capture richer operational semantics than SQL statements alone because they describe how queries are executed rather than merely how they are written. Consequently, execution plans provide a more informative representation for analysing database behaviour and identifying deviations that may indicate abnormal or malicious activities. Despite this potential, existing research has largely focused on performance optimisation and operational analytics, with comparatively limited attention devoted to their application in intelligent database security.

Furthermore, current execution-plan-based approaches rarely incorporate semantic computing, ontology engineering, or graph-based knowledge representation to model execution behaviour and support explainable security reasoning [42, 45-46]. The absence of semantic modelling limits their ability to represent structural relationships, infer behavioural dependencies, and provide transparent explanations for security decisions. This gap highlights the need for an execution-aware semantic framework that transforms execution plans into structured semantic knowledge capable of supporting behavioural validation, anomaly detection, and explainable database security. The proposed SSVF addresses this limitation by integrating execution-plan semantics with graph-based knowledge representation and intelligent security validation within a unified Computer Science framework.

2.4 Research Gap

The literature demonstrates significant progress in database security, SQL injection detection, semantic computing, graph analytics, and execution-plan analysis [1, 6, 17-19]. Conventional database security mechanisms provide robust protection through access control, encryption, auditing, and intrusion detection, while machine learning and artificial intelligence techniques have improved the identification of previously unseen attack patterns. Similarly, semantic technologies and graph-based models have advanced knowledge representation and reasoning across diverse cybersecurity applications, and execution plans have become indispensable for query optimisation and performance analysis. Collectively, these contributions have strengthened different aspects of database security; however, they have largely evolved as independent research directions.

Despite these advances, a fundamental gap remains in the integration of these technologies into a unified framework for intelligent relational database security. Existing SQL injection detection techniques continue to rely predominantly on syntactic analysis, statistical learning, or predefined attack signatures, whereas execution-plan research remains largely confined to performance optimisation. Likewise, semantic computing and graph-based approaches have received limited attention in modelling execution semantics for security validation, and explainable reasoning is seldom incorporated into database security frameworks. Consequently, current approaches provide limited capability to interpret database execution behaviour, identify semantic anomalies, or generate transparent and explainable security decisions.

This research addresses these limitations by proposing the **Security-Aware Semantic Validation Framework (SSVF)**, an execution-aware semantic framework that integrates execution-plan analysis, semantic computing, ontology engineering, graph-based knowledge representation, and behavioural validation within a unified Design Science Research artefact. Unlike existing approaches, SSVF treats database execution semantics as the primary source of security intelligence, providing a formal foundation for semantic validation, explainable threat analysis, and intelligent database security. The following section presents the **Design Science Research (DSR) methodology** adopted to guide the design, development, and theoretical evaluation of the proposed **Security-Aware Semantic Validation Framework**.

3. Research Methodology

3.1 Design Science Research

This study adopts the **Design Science Research (DSR)** methodology to guide the systematic design, development, and theoretical evaluation of the proposed Security-Aware Semantic Validation Framework (SSVF). DSR is widely recognised within Computer Science research as an appropriate methodology for developing innovative artefacts that address identified research problems through rigorous design and evaluation. Unlike explanatory research methodologies that primarily investigate existing phenomena, DSR focuses on creating and evaluating purposeful artefacts capable of solving practical and scientific challenges while simultaneously contributing new theoretical knowledge [47-48].

The selection of DSR is motivated by the objective of this research, which extends beyond analysing existing database security mechanisms to designing a novel execution-aware semantic framework for intelligent relational database security. The proposed SSVF constitutes the primary research artefact, integrating execution-plan analysis, semantic computing, ontology engineering, graph-based knowledge representation, and behavioural validation within a unified architectural framework. Consequently, DSR provides an appropriate methodological foundation for systematically translating the identified research gaps into a coherent and scientifically rigorous framework.

Following established DSR principles, the development of SSVF follows six iterative Design Science Research activities: **problem identification**, **objective definition**, **artefact design**, **artefact development**, **theoretical demonstration and evaluation**, and **research communication**. These activities ensure that the proposed framework is explicitly derived from the limitations identified in the existing literature while maintaining traceability between the research problem, framework design, and expected scientific contributions. The resulting methodology emphasises methodological rigour, reproducibility, and implementation readiness, thereby aligning with the long-term objectives of the SSVF research programme.

3.2 Research Method Implementation

The implementation of the DSR process in this study begins with a comprehensive review of existing database security research to identify unresolved challenges associated with

syntax-oriented security mechanisms, limited semantic reasoning, and the absence of execution-aware validation. These observations establish the research problem and define the functional requirements for an intelligent semantic database security framework.

Guided by these requirements, the SSVF is designed as a layered architectural artefact that transforms database execution plans into semantic knowledge capable of supporting behavioural validation and explainable security reasoning. The framework is formally specified through architectural components, mathematical definitions, and semantic relationships that collectively establish a rigorous foundation for intelligent database security analysis.

The proposed framework is subsequently demonstrated through its architectural specification and formally evaluated using established Design Science Research evaluation principles. Rather than implementing a software prototype within this study, the evaluation examines the framework's architectural completeness, semantic consistency, theoretical soundness, and its capacity to address the research gap identified in **Section 2**. This theoretical evaluation establishes the scientific validity of the framework while providing a foundation for subsequent implementation and empirical validation in later phases of the SSVF research programme.

4. Security-Aware Semantic Validation Framework

4.1 Framework Overview

The Security-Aware Semantic Validation Framework (SSVF) is an execution-aware semantic framework designed to enhance relational database security by transforming database execution plans into structured semantic knowledge for intelligent behavioural validation. Unlike conventional database security approaches that primarily analyse SQL syntax or predefined attack signatures, SSVF utilises execution semantics as the principal source of security intelligence, enabling the identification of abnormal query behaviour through semantic interpretation rather than lexical analysis.

As illustrated in **Figure 2**, the framework comprises six interconnected layers that collectively support semantic validation and intelligent security reasoning. Beginning with SQL query processing, the framework extracts execution plans generated by the relational database optimiser and transforms them into semantic graph representations through ontology-driven knowledge modelling. The resulting semantic structures are subsequently analysed using behavioural validation and semantic reasoning mechanisms to identify execution anomalies, classify potential security threats, and generate explainable security decisions. By integrating execution-plan analysis, semantic computing, graph-based knowledge representation, and intelligent reasoning within a unified architecture, SSVF establishes a unified architectural foundation for proactive and explainable relational database security.

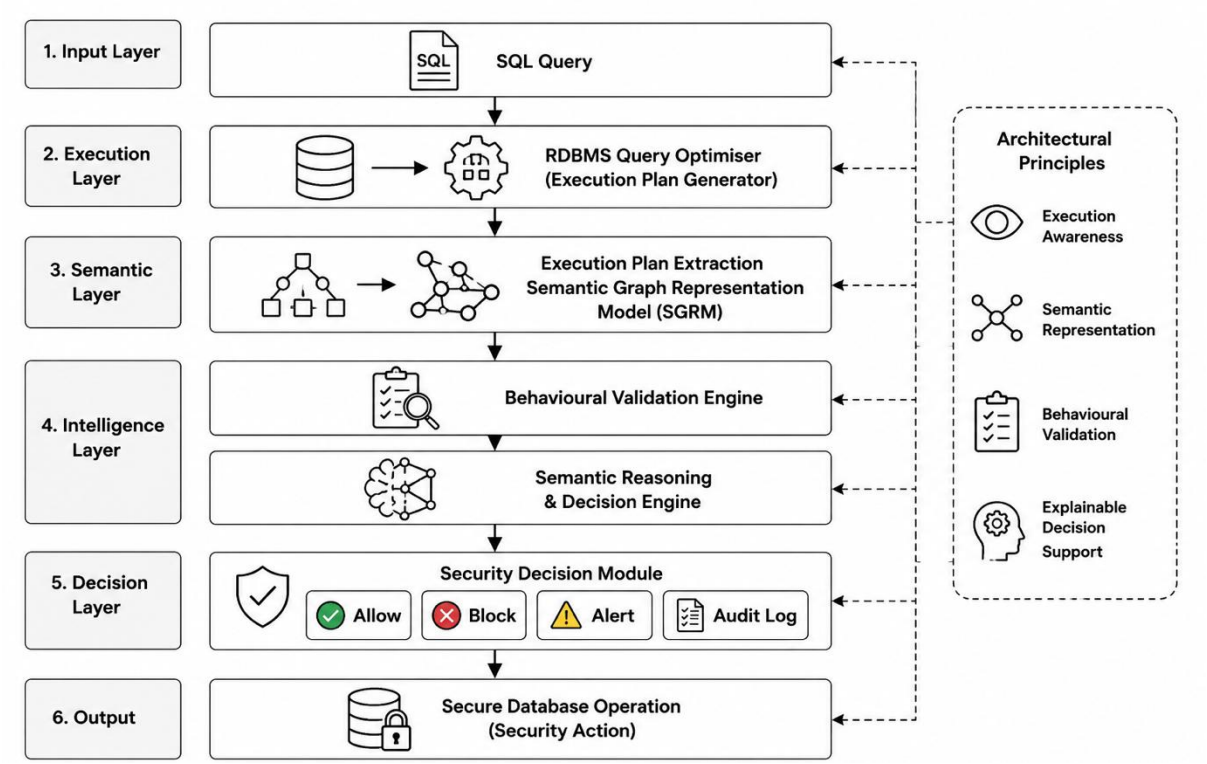


Fig. 2 Architecture of the proposed Security-Aware Semantic Validation Framework (SSVF), illustrating the transformation of SQL query execution into semantic knowledge through execution-plan extraction, semantic graph representation, behavioural validation, semantic reasoning, and intelligent security response

The framework is designed according to four fundamental architectural principles. First, **execution awareness** ensures that security analysis is derived from the operational semantics of database execution rather than solely from SQL syntax. Second, **semantic representation** preserves execution relationships through ontology-based graph structures that facilitate machine reasoning. Third, **behavioural validation** evaluates execution patterns against formally defined semantic constraints to detect anomalous database behaviour. Finally, **explainable decision support** provides transparent reasoning for every security decision, thereby improving interpretability and supporting trustworthy intelligent database security. Together, these principles establish the architectural foundation upon which the remaining components of SSVF are formally specified.

4.2 Framework Architecture

The Security-Aware Semantic Validation Framework (SSVF) adopts a layered architecture that systematically transforms database query execution into semantic knowledge for intelligent security analysis. Each architectural layer performs a distinct function while interacting with adjacent layers through well-defined semantic interfaces. This modular organisation promotes scalability, extensibility, and interoperability, enabling individual components to evolve without affecting the overall framework.

As illustrated in **Figure 2**, the architectural workflow begins when a SQL query is submitted to the Relational Database Management System (RDBMS). Rather than analysing the SQL statement directly, SSVF obtains the execution plan generated by the query optimiser, which provides a structured representation of the physical execution strategy. The execution plan is subsequently analysed to extract execution operators, relational dependencies, access methods, predicates, and other execution metadata required for semantic modelling.

The extracted execution information is then transformed into an ontology-driven semantic graph that captures execution entities together with their structural and behavioural relationships. This semantic representation establishes a machine-interpretable knowledge model capable of preserving execution semantics beyond conventional relational or textual representations. The semantic graph subsequently serves as the knowledge foundation for behavioural validation and semantic reasoning.

The behavioural validation layer evaluates the semantic graph against formally defined validation rules to identify execution inconsistencies, anomalous behavioural patterns, and potential security violations. The resulting semantic evidence is processed by the reasoning layer, which classifies execution behaviour, generates explainable security decisions, and determines the appropriate security response. Depending on the validation outcome, the framework either authorises query execution or produces security alerts for further investigation. Through this layered architecture, SSVF integrates execution-plan analysis, semantic modelling, behavioural validation, and explainable reasoning into a unified Computer Science framework for intelligent relational database security.

4.3 Framework Components

Each architectural layer is realised through a corresponding functional component. The Security-Aware Semantic Validation Framework (SSVF) comprises six functional components that collectively transform database execution behaviour into semantic knowledge for intelligent security validation. Each component performs a specialised task while exchanging structured semantic information with adjacent components, thereby ensuring a coherent and execution-aware validation process.

The **SQL Query Processing Component** serves as the entry point of the framework. It receives SQL statements submitted by users or applications and forwards them to the Relational Database Management System (RDBMS) for query optimisation. Rather than inspecting the SQL statement itself, SSVF utilises the execution plan generated by the database optimiser as the primary representation of query behaviour.

The **Execution Plan Extraction Component** acquires the generated execution plan and extracts its operational characteristics, including execution operators, relational dependencies, access paths, join strategies, predicates, and execution metadata. These elements collectively describe the physical execution behaviour of the query and provide the semantic information required for subsequent processing.

The **Semantic Representation Component**, implemented through the **Semantic Graph Representation Model (SGRM)**, transforms the extracted execution information into an ontology-driven semantic graph. This component represents execution operators, database objects, and their relationships as interconnected semantic entities, thereby preserving both structural and behavioural semantics in a machine-interpretable form. As the semantic representation layer of SSVF, SGRM provides the knowledge foundation upon which subsequent validation and reasoning processes operate.

The **Behavioural Validation Component** evaluates the semantic graph against predefined semantic validation rules and behavioural constraints to determine whether the execution behaviour conforms to expected operational semantics. Rather than relying solely on syntactic signatures or statistical features, this component identifies inconsistencies through semantic interpretation of execution relationships and behavioural dependencies.

The **Semantic Reasoning and Decision Component** analyses the validation results to infer the security state of the execution behaviour. Using ontology-driven reasoning and graph-based inference, the component classifies execution outcomes, explains detected anomalies, and generates transparent security decisions. The explainable nature of this component supports analyst interpretation and facilitates trustworthy intelligent database security.

Finally, the **Security Response Component** produces the framework's operational outcome. Depending on the reasoning results, it authorises normal query execution, generates security alerts, records validation evidence for auditing, or initiates appropriate mitigation actions. Collectively, these six components establish an integrated execution-aware framework that combines semantic computing, graph-based knowledge representation, behavioural validation, and explainable reasoning to support intelligent relational database security.

4.4 Operational Workflow

The Security-Aware Semantic Validation Framework (SSVF) operates through a structured, multi-stage workflow that systematically transforms SQL query execution into semantic knowledge for security analysis and decision-making. This workflow ensures that every database operation is evaluated not only at the syntactic level but also at the semantic and behavioural levels, thereby enabling execution-aware security validation.

The process begins when a user or application submits a SQL query to the database system. Instead of executing the query immediately, the database optimiser first generates an execution plan that specifies how the query will be physically executed. SSVF intercepts this execution plan and forwards it to the execution plan extraction layer, where structural and operational features are identified and isolated for semantic transformation.

The extracted execution elements are then passed to the Semantic Representation Component, where they are mapped into an ontology-driven semantic graph using the Semantic Graph Representation Model (SGRM). This transformation converts low-level execution details into structured semantic entities and relationships, enabling formal reasoning over query behaviour. At this stage, the execution plan transitions from a procedural description into a knowledge-based representation suitable for intelligent analysis.

Once the semantic graph is constructed, the Behavioural Validation Component evaluates it against predefined semantic rules and constraints derived from expected database execution behaviour. This stage identifies deviations, inconsistencies, and abnormal execution patterns that may indicate malicious intent or policy violations. The validation output is then forwarded to the Semantic Reasoning and Decision Component.

The reasoning component applies ontology-based inference and graph reasoning techniques to interpret validation results, classify execution behaviour, and determine the security implications of the query. It generates an explainable justification for each decision, ensuring that security outcomes are transparent and interpretable. Based on this reasoning output, the Security Response Component executes the final decision, which may include permitting query execution, blocking the query, raising a security alert, or logging the event for auditing purposes.

Through this sequential yet tightly integrated workflow, SSVF ensures that relational database queries are evaluated through a continuous pipeline of execution-aware semantic transformation, behavioural validation, and explainable reasoning. This workflow forms the operational backbone of the framework and bridges the gap between database execution mechanics and intelligent security decision-making.

5. Formal Specification of the Framework

5.1 Formal Framework Definition

To establish a rigorous theoretical foundation, the Security-Aware Semantic Validation Framework (SSVF) is defined as an execution-aware semantic system that transforms relational database query execution into structured semantic knowledge for intelligent security validation. The framework integrates execution-plan analysis, semantic graph representation, behavioural validation, and security decision support within a unified computational model. The framework is formally represented as:

$$SSVF = \langle Q, E, G, V, D \rangle \quad (1)$$

where Q denotes the set of SQL queries submitted to the relational database system, E represents execution plans generated by the query optimiser, G denotes the semantic graph derived from execution semantics, V is the semantic validation function evaluating behavioural correctness, and D is the security decision function producing the final system action. The framework operates as a sequential transformation process:

$$Q \rightarrow E \rightarrow G \rightarrow V \rightarrow D \quad (2)$$

and the overall mapping is expressed as:

$$D = f(V(G(\Phi(E(Q))))) \quad (3)$$

As expressed in **Equation (3)**, $\Phi(\cdot)$ denotes the transformation function that converts execution plans into semantic graph representations.

5.2 Semantic Graph Formalization

The semantic representation layer is defined as a directed labelled graph:

$$G = (V, E_G, L_V, L_E) \quad (4)$$

where V is the set of vertices representing execution entities, $E \subseteq V \times V$ denotes directed execution dependencies, $L_V: V \rightarrow \Sigma_V$ is the node labelling function, and $L_E: E_G \rightarrow \Sigma_E$ is the edge labelling function. Each vertex is represented as:

$$v = \langle\ id, type, attributes\ \rangle \quad (5)$$

where id is a unique identifier, $type \in \Sigma_V$ defines the semantic category (e.g., join, scan, filter, table, index), and $attributes$ capture execution metadata such as cost, cardinality, and access paths. Each edge is defined as:

$$e = (v_i, v_j, r) \quad (6)$$

where $r \in \Sigma_E$ represents the semantic relationship between execution entities, including dependencies such as feeds-into, depends-on, accesses, and filters. The transformation from execution plans to semantic graphs is formally defined by Equation (7) and (8) as:

$$\Phi: E_G \rightarrow G, G = \Phi(E_G) \quad (7, 8)$$

The resulting graph satisfies structural and semantic constraints. First, execution dependencies form a directed acyclic structure, ensuring that:

$$G \in DAG \quad (9)$$

Second, all node labels conform to ontology definitions such that:

$$L_V(v) \in O_V \quad (10)$$

Third, every execution entity is traceable to its origin in the execution plan, ensuring behavioural traceability across the transformation process.

This formulation transforms execution plans into a structured semantic representation, enabling relational query execution to be analysed as interconnected behavioural knowledge rather than linear procedural instructions.

5.3 Semantic Validation Model

The semantic validation function is defined as:

$$V: G \rightarrow \{0,1\} \quad (11)$$

where

$$V(G) = 1 \quad (12)$$

indicates valid execution behaviour and

$$V(G) = 0 \quad (13)$$

indicates anomalous or potentially malicious behaviour. Validation is governed by an anomaly-driven decision mechanism based on a constraint evaluation function. The final decision is expressed as:

$$V(G) = \begin{cases} 1, & \text{if } \Psi(G) \leq \tau \\ 0, & \text{if } \Psi(G) > \tau \end{cases} \quad (14)$$

where $\Psi(G)$ represents the anomaly score derived from constraint violations. Let

$$C = \{C_1, C_2, \ldots, C_n\} \tag{15}$$

denote the constraint set governing execution behaviour. The anomaly score is computed according to **Equation (11)**:

$$\Psi(G) = \sum_{i=1}^{n} w_i \cdot \delta(C_i(G)). \tag{16}$$

where w_i is the weight of constraint C_i, and

$$\delta(C_i(G)) = 1 \text{ if constraint is violated, otherwise } 0. \tag{17}$$

The validation constraints capture structural consistency, behavioural deviation, and execution integrity. Structural consistency ensures that all execution dependencies correspond to valid semantic relationships defined in the ontology. Behavioural deviation evaluates divergence from baseline execution behaviour $G_{baseline}$ using a bounded threshold ϵ. Execution integrity ensures that all execution paths maintain valid relational semantics without violating expected database access patterns. This formulation enables security validation to be expressed as constraint-driven anomaly detection over execution semantics encoded within the graph structure.

5.4 Security Decision Model

The security decision function maps validated execution behaviour into operational security actions:

$$D: G \rightarrow A \tag{18}$$

where

$$A = \{Allow, Block, Alert, Audit\} \tag{19}$$

defines the set of possible security responses. The decision function, defined in **Equation (14)**, depends on both validation outcomes and anomaly scores:

$$D(G) = f\big(V(G), \Psi(G)\big) \tag{20}$$

The decision policy is formally specified in **Equation (15)**:

$$D(G) = \begin{cases} Allow, & \text{if } V(G) = 1 \wedge \Psi(G) \leq \tau_1 \\ Audit, & \text{if } V(G) = 1 \wedge \tau_1 < \Psi(G) \leq \tau_2 \\ Alert, & \text{if } V(G) = 0 \wedge \Psi(G) \leq \tau_3 \\ Block, & \text{if } \Psi(G) > \tau_3 \end{cases} \tag{21}$$

The explanation function is defined in **Equation (16)**:

$$E(G) = \{\, C_i \mid C_i(G) = 0 \,\} \cup \{\Psi(G), V(G)\} \tag{22}$$

This ensures that every security decision is accompanied by traceable reasoning derived from constraint violations and anomaly scores. At the system level, the complete execution-aware decision pipeline is expressed in **Equation (17)** as:

$$D(G) = f\left(V\left(\Phi\big(E(Q)\big)\right)\right) \tag{23}$$

which links SQL query input to execution plan transformation, semantic graph construction, behavioural validation, and final security decision generation.

This completes a formally grounded execution-aware security model that converts relational database execution semantics into explainable and enforceable security actions.

6. Theoretical Evaluation of SSVF

The Security-Aware Semantic Validation Framework (SSVF) is theoretically evaluated based on its ability to establish a formally consistent transformation from SQL query inputs into execution-aware semantic security decisions. The framework defines a deterministic pipeline $Q \rightarrow E \rightarrow G \rightarrow V \rightarrow D$, ensuring that every query submitted to a relational database system undergoes structured interpretation through execution plans, semantic graph

construction, behavioural validation, and decision synthesis. This transformation is functionally complete in the sense that each stage preserves semantic fidelity while progressively enriching the representation of query execution behaviour.

From a security perspective, SSVF provides a significant departure from traditional SQL injection detection mechanisms by shifting the analysis focus from syntactic patterns to execution-level semantics. Whereas conventional approaches rely on lexical inspection or statistical classification of query strings, SSVF operates on execution plans, which encode the actual operational behaviour of the database engine. This enables the framework to distinguish between syntactically valid but semantically malicious queries and legitimate queries that conform to expected execution behaviour. Consequently, the framework introduces a semantic separation between normal and anomalous execution spaces, where malicious behaviour manifests as measurable deviation within execution graph structures.

The semantic expressiveness of SSVF is grounded in its graph-based representation model $G = (V, E_G, L_V, L_E)$, which captures execution operators, relational dependencies, and metadata attributes in a unified structure. This representation ensures that execution behaviour is not reduced to isolated features but preserved as an interconnected knowledge system. As a result, execution semantics become explicitly machine-interpretable, enabling structured reasoning over query behaviour and supporting deeper analysis of relational dependencies that are typically lost in feature-based security models.

In terms of computational feasibility, the framework operates within polynomial complexity bounds determined by graph construction and constraint evaluation processes. The transformation of execution plans into semantic graphs follows linear scalability with respect to the number of execution entities and dependencies, while validation operates over a finite set of semantic constraints. This theoretical complexity indicates that the framework is suitable for deployment in real-world database environments where query execution requires low-latency security validation.

Finally, SSVF provides inherent explainability through its constraint-driven validation mechanism, where security decisions are directly traceable to violated semantic rules and quantified anomaly scores. This traceability ensures that every decision produced by the framework can be interpreted in terms of execution behaviour, enabling transparent auditing and supporting trustworthy security automation in relational database systems.

7. Discussion and Implications

The Security-Aware Semantic Validation Framework (SSVF) introduces a shift in relational database security by repositioning execution plans as the primary unit of security analysis. This departure from conventional syntax-centric and signature-driven approaches reframes database security as a semantic and behavioural problem situated at the execution layer of query processing. The implication of this shift is that security validation is no longer constrained to what a query *looks like* but extends to how the database engine *intends to execute it*. This transition strengthens the analytical depth of security systems by embedding operational semantics directly into the decision-making pipeline.

From a Computer Science perspective, the framework aligns database security with principles of semantic computing and graph-based knowledge representation. By modelling execution plans as structured semantic graphs, SSVF enables relational query behaviour to be interpreted as a machine-interpretable knowledge system rather than a linear instruction sequence. This allows security analysis to exploit structural dependencies between operators, data objects, and execution flows, thereby exposing behavioural irregularities that remain hidden in traditional relational or string-based representations. The implication is a more expressive and context-aware security model that bridges database systems and knowledge representation theory.

In practical cybersecurity terms, SSVF addresses a persistent limitation in existing SQL injection detection techniques, namely their dependence on syntactic features that can be easily preserved under obfuscation or encoding attacks. By operating at the level of execution semantics, the framework diminishes the effectiveness of adversarial strategies that manipulate query structure without altering execution intent. This introduces a stronger security invariant based on behavioural consistency, where malicious intent is inferred through deviation from expected execution patterns rather than lexical anomalies. As a result, SSVF strengthens the resilience of database systems against adaptive and structurally disguised attacks.

The integration of explainable reasoning further extends the implications of the framework into the domain of trustworthy artificial intelligence. Unlike black-box detection models, SSVF provides traceable decision paths grounded in constraint violations and anomaly scoring over semantic graphs. This enables security analysts to interpret not only the final decision but also the structural and behavioural evidence that produced it. Such transparency is particularly significant in high-assurance database environments where auditability and justification of security actions are critical requirements.

Overall, the proposed framework establishes a conceptual bridge between database execution theory, semantic modelling, and intelligent cybersecurity. It suggests that future database security systems may increasingly rely on execution-aware semantic reasoning as a foundational paradigm. This positions SSVF not only as a detection mechanism but also as a broader architectural paradigm toward intelligent, explainable, and semantically grounded database security systems.

8. Research Contributions

The Security-Aware Semantic Validation Framework (SSVF) makes a set of distinct contributions to the field of relational database security by redefining how query execution is interpreted, analysed, and transformed into security intelligence. Unlike conventional approaches that treat database security as a syntactic filtering or statistical classification problem, this work establishes execution semantics as the primary analytical substrate for intelligent security reasoning. This shift introduces a new conceptual layer in which relational database behaviour is interpreted through structured semantic representations derived from execution plans.

A key contribution of the framework is the introduction of an execution-aware semantic transformation pipeline that formalises the progression from SQL query submission to security decision generation. By structuring the process as $Q \rightarrow E \rightarrow G \rightarrow V \rightarrow D$, the framework provides a deterministic and formally verifiable pathway for mapping database queries into interpretable security outcomes. This contribution is significant because it establishes a unified computational model that integrates database execution theory with semantic computing in a formally defined manner.

Another major contribution lies in the development of a semantic graph representation model for database execution behaviour. By transforming execution plans into directed labelled graph structures, SSVF enables relational execution semantics to be preserved in a machine-interpretable form that supports structural reasoning. This representation moves beyond feature-based modelling and introduces a knowledge-centric perspective in which execution operators, dependencies, and relational interactions become explicit components of a semantic network.

The framework further contributes a constraint-driven behavioural validation mechanism that enables the detection of anomalous execution patterns through formal semantic rules. Rather than relying on purely probabilistic classification, SSVF defines validation as a structured evaluation of execution consistency against a set of semantic constraints. This

provides a mathematically grounded mechanism for identifying deviations in database behaviour, strengthening both the precision and interpretability of security detection.

Finally, the integration of explainable decision-making within the framework constitutes an important contribution to trustworthy database security systems. By ensuring that every security decision is traceable to violated constraints and quantifiable anomaly scores, SSVF provides transparency in automated security enforcement. This explainability dimension strengthens its applicability in high-assurance environments where interpretability and auditability are essential requirements.

9. Limitations and Future Work

While the Security-Aware Semantic Validation Framework (SSVF) establishes a formally grounded and execution-aware approach to relational database security, certain limitations remain inherent to its theoretical and architectural design. These limitations primarily arise from its abstraction level, dependency on execution-plan availability, and reliance on predefined semantic constraints for behavioural validation.

One fundamental limitation is the dependency on the database query optimiser for the generation of execution plans. Since SSVF operates on execution-level semantics, its effectiveness is contingent on the availability, consistency, and expressiveness of execution plans produced by the underlying relational database management system. Variations in optimiser implementations across database platforms may therefore introduce heterogeneity in semantic representation, potentially affecting the consistency of the framework's semantic representations and analytical outcomes.

A second limitation relates to the construction and calibration of semantic constraints used in behavioural validation. The accuracy of anomaly detection is influenced by the quality of the constraint set C and the definition of baseline execution behaviour $G_{baseline}$. In complex or highly dynamic database environments, establishing stable behavioural baselines may present challenges, particularly in workloads characterised by frequent schema changes or adaptive query patterns.

The framework also assumes the availability of well-defined ontology structures for mapping execution plans into semantic graphs. In practice, the design and maintenance of such ontologies may require significant domain expertise, and incomplete or poorly aligned ontological models may reduce the precision of semantic representation. This introduces a dependency on the maturity of ontology engineering practices within the database security domain.

From a computational perspective, although SSVF operates within polynomial complexity bounds, scalability considerations may emerge in extremely large-scale database systems with high query throughput. The overhead associated with graph construction, constraint evaluation, and semantic reasoning may require optimisation strategies or distributed processing mechanisms to ensure real-time applicability in production environments.

Future research directions focus on addressing these limitations while extending the theoretical and practical capabilities of the framework. One key direction involves the development of adaptive learning mechanisms for automatic generation and refinement of semantic constraints based on evolving database workloads. Another direction concerns the integration of lightweight ontology alignment techniques to reduce the dependency on manually engineered semantic models. Additionally, future work may explore distributed and incremental graph processing techniques to enhance scalability and support high-throughput database environments.

Further research is also expected to investigate the empirical evaluation and validation of SSVF through prototype implementation and benchmarking against existing SQL injection detection systems. This would enable quantitative evaluation of detection accuracy,

computational efficiency, and real-world applicability. Moreover, the integration of advanced explainable AI techniques with semantic graph reasoning presents an opportunity to further strengthen interpretability and decision transparency in database security systems.

10. Conclusion

This study presented the Security-Aware Semantic Validation Framework (SSVF), an execution-aware approach to relational database security that integrates semantic computing, graph-based knowledge representation, and behavioural validation within a unified formal model. The framework departs from traditional SQL injection detection mechanisms by shifting security analysis from syntactic patterns and statistical features to execution-level semantics derived from database query execution plans [1, 6].

By formalising the transformation pipeline $Q \rightarrow E \rightarrow G \rightarrow V \rightarrow D$, SSVF establishes a structured mechanism for converting SQL queries into semantically enriched execution representations and ultimately into explainable security decisions. This formulation provides both theoretical consistency and computational clarity, ensuring that each stage of the framework contributes meaningfully to security reasoning within relational database systems.

The integration of semantic graph modelling enables execution behaviour to be represented as structured knowledge, thereby supporting deeper analytical reasoning over relational dependencies and operational semantics [17-18]. In addition, the incorporation of constraint-driven behavioural validation introduces a formal mechanism for detecting anomalous execution patterns, strengthening the ability of the framework to identify malicious activity that bypasses conventional detection methods.

Furthermore, the explainability component of SSVF ensures that security decisions remain transparent and auditable by linking outcomes to violated constraints and quantified anomaly scores. This positions the framework within the broader research direction of explainable artificial intelligence in cybersecurity, where interpretability is a critical requirement for trustworthy automated decision systems [20].

Overall, SSVF establishes a rigorous execution-aware semantic foundation for intelligent relational database security and provides a formally grounded architectural basis for future implementation, empirical validation, and deployment within next-generation database security systems.

References

[1] Bertino, E. and Sandhu, R. (2005) ‘Database security—Concepts, approaches, and challenges’, IEEE Transactions on Dependable and Secure Computing, 2(1), pp. 2–19. https://doi.org/10.1109/TDSC.2005.9

[2] Damiani, E., De Capitani di Vimercati, S., Paraboschi, S. and Samarati, P. (2009), "Securing databases", in Liu, L. and Özsu, M.T. (Eds), Encyclopedia of Database Systems, Springer, Boston, MA, pp. 2459–2464.

[3] Silberschatz, A., Korth, H.F. and Sudarshan, S. (2020) Database System Concepts. 7th edn. New York: McGraw-Hill Education.

[4] Khan, W., Kumar, T., Zhang, C., Raj, K., Roy, A.M. and Luo, B. (2023), "SQL and NoSQL database software architecture performance analysis and assessments: A systematic literature review", Big Data and Cognitive Computing, Vol. 7 No. 2, Article 97, doi:10.3390/bdcc7020097.

[5] Dritsas, E. and Trigka, M. (2025), "A survey on database systems in the big data era: Architectures, performance, and open challenges", IEEE Access.

[6] Zuech, R., Khoshgoftaar, T.M. and Wald, R. (2015) ‘Intrusion detection and Big Heterogeneous Data: A survey’, Journal of Big Data, 2(3), pp. 1–41. https://doi.org/10.1186/s40537-015-0013-4

[7] Nasereddin, M., ALKhamaiseh, A., Qasaimeh, M. and Al-Qassas, R. (2023), "A systematic review of detection and prevention techniques of SQL injection attacks", Information Security Journal: A Global Perspective, Vol. 32 No. 4, pp. 252–265.

[8] Ajasa, A.D., Chizari, H. and Alam, A. (2025), "Database security and performance: A case of SQL injection attacks using Docker-based virtualisation and its effect on performance", Future Internet, Vol. 17 No. 4, Article 156, doi:10.3390/fi17040156.

[9] Kashyap, A. and Jana, A. (2025), "A survey on detection and prevention techniques of SQL injection attacks", International Journal of Information and Computer Security, Vol. 26 No. 4, pp. 332–371.

[10] Afrin, S., Elsayed, M.A. and Zincir-Heywood, N. (2025), "SQL-GENIE: SQL protection using generative modeling for anomaly detection against injection and evolved adversarial attacks", Proceedings of the 2025 IEEE 49th Annual Computers, Software, and Applications Conference (COMPSAC), IEEE, pp. 459–464.

[11] Chaudhuri, S. (1998) ‘An overview of query optimization in relational systems’, Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS ’98), Seattle, Washington, USA, 1–3 June 1998, pp. 34–43. https://doi.org/10.1145/275487.275492

[12] Leis, V., Gubichev, A., Mirchev, A., Boncz, P., Kemper, A. and Neumann, T. (2015), "How good are query optimizers, really?", Proceedings of the VLDB Endowment, Vol. 9 No. 3, pp. 204–215, doi:10.14778/2850583.2850594.

[13] Leis, V., Gubichev, A., Mirchev, A., Boncz, P.A., Kemper, A. and Neumann, T. (2018) 'Query optimization through the looking glass, and what we found running the Join Order Benchmark', The VLDB Journal, 27(5), pp. 643–668. https://doi.org/10.1007/s00778-017-0498-3

[14] Fischer, T., Hirn, D. and Grust, T. (2024), "SQL engines excel at the execution of imperative programs", Proceedings of the VLDB Endowment, Vol. 17 No. 13, pp. 4696–4708.

[15] Chasialis, K., Foufoulas, Y., Simitsis, A. and Ioannidis, Y.E. (2026), "Optimizing UDF Queries in SQL Data Engines", in Proceedings of the 29th International Conference on Extending Database Technology (EDBT 2026), pp. 247–260.

[16] Akoglu, L., Tong, H. and Koutra, D. (2015) 'Graph based anomaly detection and description: A survey', Data Mining and Knowledge Discovery, 29(3), pp. 626–688. https://doi.org/10.1007/s10618-014-0365-y

[17] Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., de Melo, G., Gutierrez, C., Kirrane, S., Gayo, J.E.L., Navigli, R., Neumaier, S., Ngonga Ngomo, A.C., Polleres, A., Rashid, S.M., Rula, A., Schmelzeisen, L., Sequeda, J., Staab, S. and Zimmermann, A. (2021) 'Knowledge graphs', ACM Computing Surveys, 54(4), Article 71, pp. 1–37. https://doi.org/10.1145/3447772

[18] Ji, S., Pan, S., Cambria, E., Marttinen, P. and Yu, P.S. (2022) 'A survey on knowledge graphs: Representation, acquisition and applications', IEEE Transactions on Neural Networks and Learning Systems, 33(2), pp. 494–514. https://doi.org/10.1109/TNNLS.2021.3070843

[19] Gong, Y., Cui, S., Liu, S., Jiang, B., Dong, C. and Lu, Z. (2024), "Graph-based insider threat detection: A survey", Computer Networks, Vol. 254, Article 110757, doi:10.1016/j.comnet.2024.110757.

[20] Sikos, L.F. (2023), "Cybersecurity knowledge graphs", Knowledge and Information Systems, Vol. 65, pp. 3511–3531, doi:10.1007/s10115-023-01860-3.

[21] Adadi, A. and Berrada, M. (2018), "Peeking inside the black-box: A survey on explainable artificial intelligence (XAI)", IEEE Access, Vol. 6, pp. 52138–52160, doi:10.1109/ACCESS.2018.2870052.

[22] Arrieta, A.B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., Chatila, R. and Herrera, F. (2020), "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI", Information Fusion, Vol. 58, pp. 82–115. doi:10.1016/j.inffus.2019.12.012.

[23] Samaraweera, G.D. and Chang, J.M. (2019), "Security and privacy implications on database systems in the big data era: A survey", IEEE Transactions on Knowledge and Data Engineering, Vol. 33 No. 1, pp. 239–258.

[24] Moissinac, B., Saad, E., Clay, M. and Berrondo, M. (2023), "Detecting SQL injection attacks using machine learning", in Proceedings of the Conference on Applied Machine Learning for Information Security (CAMLIS 2023), pp. 49–59.

[25] Arasteh, B., Aghaei, B., Farzad, B., Arasteh, K., Kiani, F. and Torkamanian-Afshar, M. (2024), "Detecting SQL injection attacks by binary gray wolf optimizer and machine learning algorithms", Neural Computing and Applications, Vol. 36 No. 12, pp. 6771–6792.

[26] Halfond, W.G.J., Viegas, J. and Orso, A. (2006) 'A classification of SQL-injection attacks and countermeasures', Proceedings of the IEEE International Symposium on Secure Software Engineering, Arlington, VA, USA, 26–29 March 2006, pp. 13–15.

[27] Hallo, M. and Suntaxi, G. (2022), "A survey on SQL injection attacks, detection and prevention techniques: A tertiary study", International Journal of Security and Networks, Vol. 17 No. 3, pp. 193–202.

[28] Sommer, R. and Paxson, V. (2010) 'Outside the closed world: On using machine learning for network intrusion detection', Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, USA, 16–19 May 2010, pp. 305–316. https://doi.org/10.1109/SP.2010.25

[29] Buczak, A.L. and Guven, E. (2016) 'A survey of data mining and machine learning methods for cyber security intrusion detection', IEEE Communications Surveys & Tutorials, 18(2), pp. 1153–1176. https://doi.org/10.1109/COMST.2015.2494502

[30] Akinsola, J.E., Efiong, J.E., Olajubu, E.A. and Aderounmu, G.A. (2023), "Artificial intelligence-based model for data security and mitigation against SQL injection attacks in web applications", Proceedings of the 2023 International Conference on Electrical, Computer and Energy Technologies (ICECET), IEEE, pp. 1–7.

[31] Ashlam, A.A., Badii, A. and Stahl, F. (2022), "Multi-phase algorithmic framework to prevent SQL injection attacks using improved machine learning and deep learning to enhance database security in real-time", Proceedings of the International Conference on Security of Information and Networks (SIN), IEEE, pp. 1–4.

[32] Gruber, T.R. (1995), "Toward principles for the design of ontologies used for knowledge sharing", International Journal of Human-Computer Studies, Vol. 43 Nos. 5–6, pp. 907–928, doi:10.1006/ijhc.1995.1081.

[33] Studer, R., Benjamins, V.R. and Fensel, D. (1998), "Knowledge engineering: Principles and methods", Data & Knowledge Engineering, Vol. 25 Nos. 1–2, pp. 161–197.

[34] Sheth, A., Ramakrishnan, C. and Thomas, C. (2005), "Semantics for the Semantic Web: The implicit, the formal and the powerful", International Journal on Semantic Web and Information Systems, Vol. 1 No. 1, pp. 1–18.

[35] Hitzler, P., Krötzsch, M. and Rudolph, S. (2020), Foundations of Semantic Web Technologies, 2nd ed., Chapman and Hall/CRC.

[36] Noel, S. and Jajodia, S. (2005), "Understanding complex network attack graphs through clustered adjacency matrices", Proceedings of the Annual Computer Security Applications Conference (ACSAC 2005), IEEE Computer Society.

[37] Undercoffer, J., Joshi, A. and Pinkston, J. (2003), "Modeling computer attacks: An ontology for intrusion detection", in Proceedings of the International Workshop on Recent Advances in Intrusion Detection (RAID 2003), Springer, Berlin, Heidelberg, pp. 113–135.

[38] Syed, Z., Padia, A., Finin, T., Mathews, L. and Joshi, A. (2016), "UCO: A unified cybersecurity ontology", Proceedings of the AAAI Workshop on Artificial Intelligence for Cyber Security, pp. 195–202.

[39] Noy, N.F. and McGuinness, D.L. (2001), Ontology Development 101: A Guide to Creating Your First Ontology, Stanford Knowledge Systems Laboratory Technical Report KSL-01-05, Stanford University, Stanford, CA.

[40] Euzenat, J. and Shvaiko, P. (2013), Ontology Matching, 2nd ed., Springer, Berlin Heidelberg.

[41] Garcia-Molina, H., Ullman, J.D. and Widom, J. (2008), Database Systems: The Complete Book, 2nd ed., Pearson.

[42] Elmasri, R. and Navathe, S.B. (2017), Fundamentals of Database Systems, 7th Global ed., Pearson.

[43] Marcus, R. and Papaemmanouil, O. (2018), "Deep reinforcement learning for join order enumeration", Proceedings of the First International Workshop on Exploiting Artificial Intelligence Techniques for Data Management (aiDM '18), ACM, Houston, TX, USA, pp. 1–4.

[44] Marcus, R. and Papaemmanouil, O. (2019), "Learning to optimize join queries with deep reinforcement learning", Proceedings of the Second International Workshop on Exploiting Artificial Intelligence Techniques for Data Management (aiDM '19), ACM, pp. 1–9, doi:10.1145/3329789.3329901.

[45] Paulheim, H. (2016), "Knowledge graph refinement: A survey of approaches and evaluation methods", Semantic Web, Vol. 8 No. 3, pp. 489–508, doi:10.3233/SW-160218.

[46] Robinson, I., Webber, J. and Eifrem, E. (2015), Graph Databases: New Opportunities for Connected Data, 2nd ed., O'Reilly Media.

[47] Hevner, A.R., March, S.T., Park, J. and Ram, S. (2004) ‘Design science in information systems research’, MIS Quarterly, 28(1), pp. 75–105.

[48] Peffers, K., Tuunanen, T., Rothenberger, M.A. and Chatterjee, S. (2007) ‘A design science research methodology for information systems research’, Journal of Management Information Systems, 24(3), pp. 45–77. https://doi.org/10.2753/MIS0742-1222240302

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- GraphicalAbstractSSVF.png