



ARTICLE

Temporal-Aware SQL Lineage Graph Analysis and Change-Impact Prediction Based on Noise Assessment

Chao Liu¹, Hong Mu¹, Min Wang², Enliang Wang² and Xuejian Zhao^{2,*}

¹Nanjing Digital Data Technology Co., Ltd., Nanjing, China

²Key Lab of Broadband Wireless Communication and Sensor Network Technology, Ministry of Education Nanjing University of Posts and Telecommunications, Nanjing, China

*Corresponding Author: Xuejian Zhao. Email: zhaoxj@njupt.edu.cn

Received: 10 May 2026; Accepted: 14 July 2026; Published: 22 October 2026

ABSTRACT: To address the limitations of enterprise-level data governance in handling dynamic evolution and predicting change impact within SQL lineage analysis, this paper proposes a noise-assessment-based, temporal-aware SQL lineage analysis method for evolving SQL lineage graphs. The approach constructs a Temporal Lineage Graph (TLG) that unifies multi-snapshot structures, semantic stability, and temporal dependencies into a single modeling framework, thereby enabling dynamic retrospection of data evolution processes. Building on this foundation, an unsupervised anomaly detection mechanism based on noise diversity is introduced, in which structural, attribute, and temporal noise instances are used to form a reference envelope around the observed temporal lineage graph. Random projection is employed to approximate high-dimensional convex-hull distances, enabling subgraph-level anomaly identification and purification. Furthermore, a bounded influence propagation algorithm integrating an information bottleneck constraint and a Noisy-OR mechanism is developed to compute high-confidence critical impact paths and node rankings from the purified lineage graph. Experiments conducted on an enterprise log dataset containing 32,456 nodes and 185,743 edges demonstrate that the proposed method achieves P@5, R@10, and NDCG@10 scores of 0.823, 0.812, and 0.847, respectively, for change-impact prediction, and reduces the recorded average root-cause localization time from 45 to 29 min. The results on the evaluated enterprise dataset indicate that the integrated framework improves change-impact ranking accuracy and supports temporal lineage retrospection under the selected graph-based comparison setting.

KEYWORDS: Temporal lineage graph; noise assessment; anomaly detection; change-impact prediction; data governance

1 Introduction

With the deepening of enterprise digitalization, column-level and expression-level data lineage has become increasingly crucial in compliance auditing, quality tracking, and impact assessment. Industry surveys and research reports indicate that data scale and governance complexity continue to rise, while a considerable proportion of governance initiatives still face high failure risks—reflecting, from another perspective, that traditional lineage perspectives dominated by static snapshots are insufficient to meet diagnostic and decision-making needs in dynamic environments [1]. Although modern lineage platforms can collect execution events and expose dependency histories, their primary temporal unit is commonly a job- or run-level event rather than an explicitly labeled, multi-snapshot representation of SQL dependency evolution. Consequently, additional analytical support is still required when users need to determine not

only that an execution occurred, but also how a column-level dependency, transformation logic, or stability state changed across snapshots. As a result, they struggle to effectively answer key questions such as “How has the computation logic of a given field evolved over the past months?” or “Will historical queries be affected by current structural changes?” [2–5].

The underlying causes of the insufficiency of static perspectives can be examined from several aspects. In engineering practice, lineage graphs are typically instantiated as directed graphs at a single point in time, ignoring the cross-temporal evolution of entities and dependencies. Consequently, fault localization and impact prediction are constrained to the current state only. In real enterprise environments, anomaly patterns often exhibit a composite form combining structural, semantic, and temporal characteristics: long chains, cycles, and abnormal fan-in structures on the one hand, and abrupt shifts or frequent fluctuations across time on the other. Classical density- or neighborhood-based methods, as well as deep representation approaches, are primarily designed for static graphs and still show clear limitations in unlabeled scenarios and temporal signal modeling. Change impact within complex systems is further subject to an inherent constraint of limited transmissible information. Although path-counting or simple decay models are easy to implement, their lack of information-theoretic upper bounds can lead to overestimation or underestimation of impact scope. Existing dynamic graph representation learning techniques offer general descriptions of structural evolution, yet their outputs are mostly node embeddings rather than interpretable column-level or operator-level dependencies, leaving a mismatch with SQL semantics and change types. In engineering practice, automated tools for change management and impact analysis typically focus on process and deliverable management, while theoretical guarantees for defining propagation boundaries and ranking impacts remain insufficient.

To address the aforementioned challenges, a noise-assessment-based temporal SQL lineage analysis method is proposed for change-impact prediction over evolving lineage graphs. The main contributions are summarized as follows:

1. A Temporal Lineage Graph (TLG) model is designed, in which static SQL dependencies, cross-snapshot temporal edges, change-type labels, and stability information are unified within a single graph schema.
2. A noise-diversity-based reference-envelope method is instantiated for temporal SQL lineage graphs, using structural, attribute, and temporal noise instances to characterize admissible lineage variations and identify suspicious subgraphs in an unlabeled setting.
3. An information-bottleneck-constrained influence propagation mechanism is introduced, in which mutual-information bounds, graph-structural decay, and Noisy-OR aggregation are incorporated to generate interpretable affected-node rankings and critical impact paths.

At the implementation level, random projection and distance-preserving bounds are employed to address the estimation of convex-hull distances in high-dimensional geometry, thereby ensuring computational feasibility and robustness of the detection process. The overall framework emphasizes an integrated closed loop of temporal consistency → anomaly purification → bounded propagation, aiming to jointly achieve the three core objectives of retracing historical trajectories, identifying anomalous patterns, and quantifying future impact.

2 Related Work

Data lineage has been extensively explored in both industry and the open-source ecosystem, covering the full pipeline from SQL Abstract Syntax Tree (AST) parsing to dependency visualization, and from governance workflow integration to change-report generation. The formal foundations of data lineage can be traced back to Buneman et al.’s definitions of *why-provenance* and *where-provenance* [2], while

Green et al. [3] unified multiple provenance models through a semiring algebra framework, providing a solid theoretical basis for understanding data dependencies in SQL queries. At the system implementation level, the Unified Lineage System tracks data provenance at scale [4], while Smoke [5] achieves interactive column-level lineage queries via in-memory indexing structures. These foundational systems emphasize fine-grained provenance extraction and interactive lineage querying. Their original focus is not the joint analysis of multi-snapshot SQL dependency evolution, change-type labeling, anomaly-aware purification, and bounded impact ranking considered in this work. Recent production lineage ecosystems have substantially expanded temporal lineage support. OpenLineage defines an object model for datasets, jobs, and runs, including runtime job-run updates and design-time job and dataset metadata events [6]. Marquez is an open-source metadata service for collecting, aggregating, and visualizing ecosystem metadata [7]. DataHub supports upstream and downstream lineage exploration and lineage impact analysis [8]. These platforms demonstrate that temporal lineage is no longer absent from practical data-governance systems. Their main focus, however, is lineage collection, metadata management, event history, and dependency exploration. The focus of the present work is different: it materializes SQL-derived multi-snapshot dependencies at column and expression granularity, assigns explicit STRUCT, SEMANTIC, and STABLE labels to cross-snapshot relationships, and combines this representation with anomaly-aware attenuation and bounded quantitative impact ranking. Therefore, TLG is intended as an analytical layer that can complement event-oriented lineage platforms rather than replace their lineage-collection, cataloging, or visualization capabilities.

In the domain of dynamic graph representation learning, EvolveGCN [9] models the evolution of graph convolutional network parameters through recurrent updates. More recent work has strengthened the evaluation and systematic understanding of temporal graph learning through the Temporal Graph Benchmark [10], TGB 2.0 for multi-relational temporal graphs [11], a comprehensive survey of dynamic graph neural networks [12], and unified architectures and libraries for dynamic graph learning [13]. Although these methods perform well on social and interaction networks, they often lack column-level or operator-level interpretability and mechanisms aligned with SQL semantics in lineage scenarios. Their primary outputs are embeddings rather than verifiable evolutionary facts, making them insufficient to directly support auditing requirements such as identifying “which dependency changed at what time.”

Research on graph anomaly detection includes local-density methods represented by Local Outlier Factor (LOF) [14], as well as deep representation approaches such as DOMINANT [15]. Ma et al. [16] provide a systematic survey of deep learning applications in graph anomaly detection. For dynamic networks, Ekle and Eberle [17] provide a comprehensive review of anomaly-detection methods for evolving graphs, while StrGNN [18] targets anomalous edge detection in temporal graphs. The former is intuitive and efficient on static topologies but has limited ability to capture cross-temporal shifts or composite anomalies; the latter achieves good performance on attributed networks but often depends on labels and task-specific tuning, and remains insufficient in modeling temporal dimensions. This work introduces the notion of noise diversity, generating complementary families of structural, attribute, and temporal noise to construct a verifiable basis for detection through spectral coverage and geometric separability. Combined with distance-preservation bounds of random projection to approximate high-dimensional convex-hull distances, the method enables subgraph-level anomaly identification under an unlabeled setting. The proposed approach simultaneously leverages topological discrepancies, attribute perturbations, and temporal rearrangements to characterize composite variations in temporal SQL lineage graphs.

In the area of change-impact analysis, database schema evolution has long been a classical research topic. Recent reviews have revisited schema evolution in contemporary database environments [19]. Curino et al.’s PRISM++ system [20] enabled automated migration and application rewriting, while Tesseract [21] studied efficient online schema evolution for snapshot databases. These studies primarily focus on the

automation and safety of schema evolution processes, yet offer relatively weak theoretical modeling for impact propagation.

In network science, Kempe et al. [22] introduced influential propagation models such as the Independent Cascade and Linear Threshold models, while Chen et al. [23,24] proposed scalable algorithms for influence maximization. In recent years, information bottleneck theory [25] has been incorporated into graph representation learning, and Wu et al. [26] formulated the Graph Information Bottleneck (GIB) principle to strike a balance between expressiveness and robustness of graph structures. Building on these insights, this work introduces the information bottleneck as a constraint mechanism, integrating mutual-information upper bounds with graph-structural decay during propagation updates. This design suppresses unrealistic accumulation of impact along long paths and ensures that the resulting Top-K impact rankings possess clear physical meaning and verifiability. Compared with workflow-oriented solutions that rely solely on engineering heuristics, this semantic-consistent, information-bounded propagation formulation provides an interpretable mechanism for constraining impact accumulation along lineage paths.

In summary, existing work has achieved significant progress in static dependency extraction, dynamic graph modeling, graph anomaly detection, and influence propagation. However, within the unified context of temporal SQL lineage, challenges remain, including insufficient modeling of evolution, instability in unsupervised detection, and lack of upper bounds in propagation. This study systematically addresses these gaps by introducing temporal semantics via the Temporal Lineage Graph (TLG), unsupervised detection driven by noise diversity, and information-bottleneck-constrained influence propagation, thereby forming an effective alignment with practical engineering scenarios.

3 Method

3.1 System Architecture Overview

The proposed temporal-aware SQL lineage analysis method is implemented within a modular system architecture for dynamic lineage tracking, anomaly-pattern assessment, and change-impact prediction in enterprise data-governance scenarios. Unlike traditional static lineage tools, the method treats the temporal dimension as a core analytical element and organizes temporal graph construction, anomaly assessment, and impact propagation into a coherent processing pipeline.

The overall system architecture consists of three core modules supported by a theoretical foundation layer, as illustrated in Fig. 1. The data flow begins with SQL logs and schema-change records from the enterprise data warehouse and proceeds through the following stages:

1. **Temporal Lineage Graph Construction Module.** This module transforms raw SQL logs and schema-change records into a structured, multi-layer temporal graph. While preserving static lineage relationships, it encodes temporal evolution information, providing a comprehensive data view for subsequent anomaly detection and impact prediction.
2. **Anomaly Detection Module.** Leveraging the theory of noise diversity, this module identifies structural, attribute-related, and temporal anomalous patterns that may indicate data-quality issues or architectural defects. Its outputs include anomalous subgraphs and an anomaly-confidence matrix M . The matrix assigns confidence values to suspicious nodes and edges, which are used to attenuate their contributions during subsequent impact propagation.
3. **Change-Impact Prediction Module.** Based on the purified lineage graph, this module quantifies the propagation scope of changes.

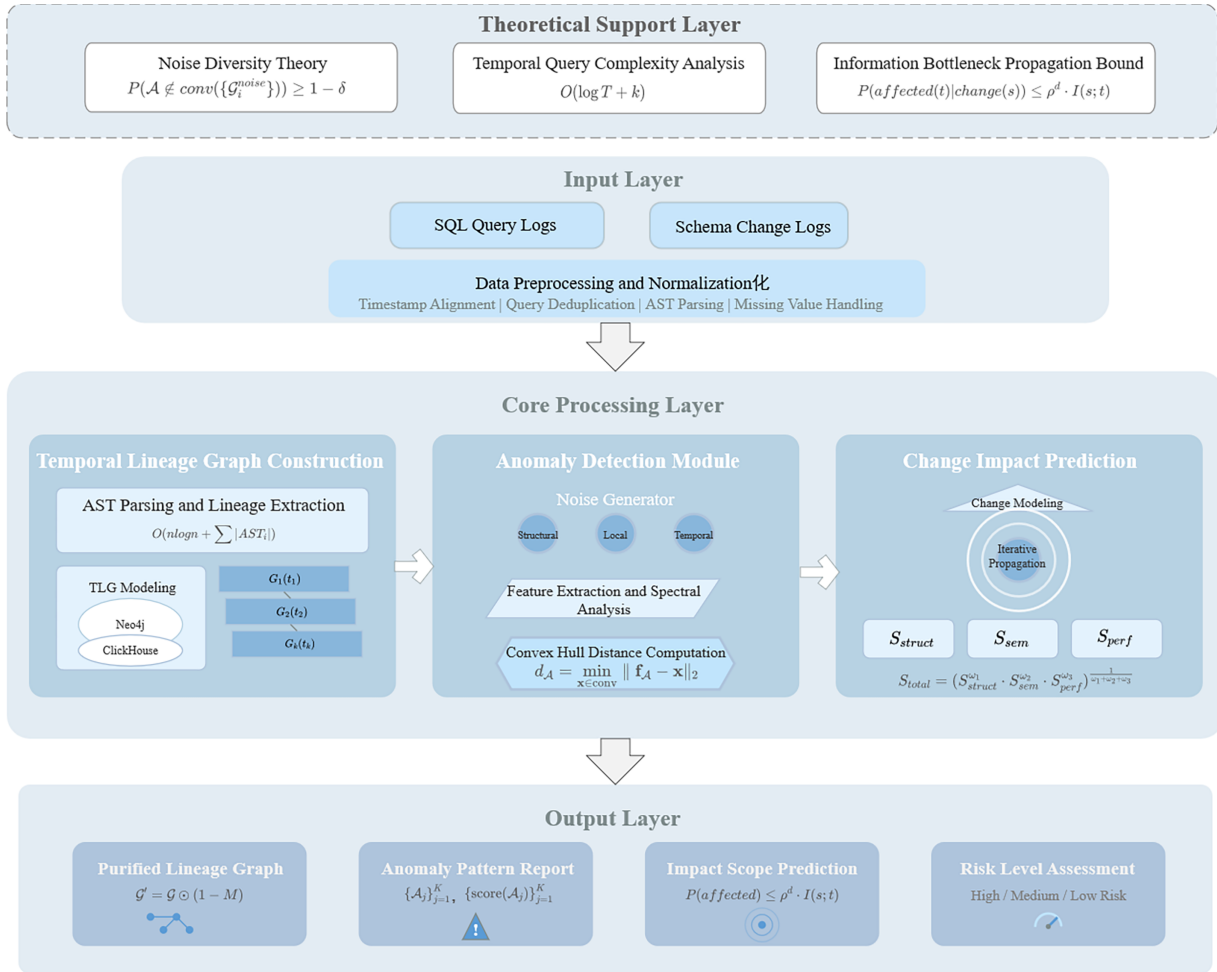


Figure 1: System architecture diagram.

The overall data flow is depicted in Fig. 2. This design ensures coherent data processing: the constructed lineage graph provides the structural foundation for anomaly detection, and the detection results serve as reliable input for impact prediction.

The system is implemented using a distributed architecture. The Temporal Lineage Graph is stored in the Neo4j graph database, supporting indexed node and neighborhood retrieval. The noise generation process for anomaly detection is accelerated through parallel computation, while change-impact prediction is executed via a message-passing interface to support distributed random walks.

3.2 Construction of the Temporal Lineage Graph

3.2.1 Data Preprocessing and Lineage Extraction

The quality of input data directly affects the accuracy of lineage analysis. This module ingests two types of raw data: timestamped SQL query logs $\{q_i = (sql_i, t_i)\}$ and database schema-change records $\{c_j = (change_j, t_j)\}$.

The preprocessing workflow includes timestamp normalization, query deduplication, and missing-value handling. First, timestamps are aligned by converting different formats into Unix time. For records with missing timestamps, linear interpolation is applied: $t_i \approx \frac{t_{i-1} + t_{i+1}}{2}$, with an associated uncertainty weight

$w_i = 0.5$. Query deduplication is performed based on the standardized SQL statements, where whitespace and case differences are removed before computing the MD5 hash.

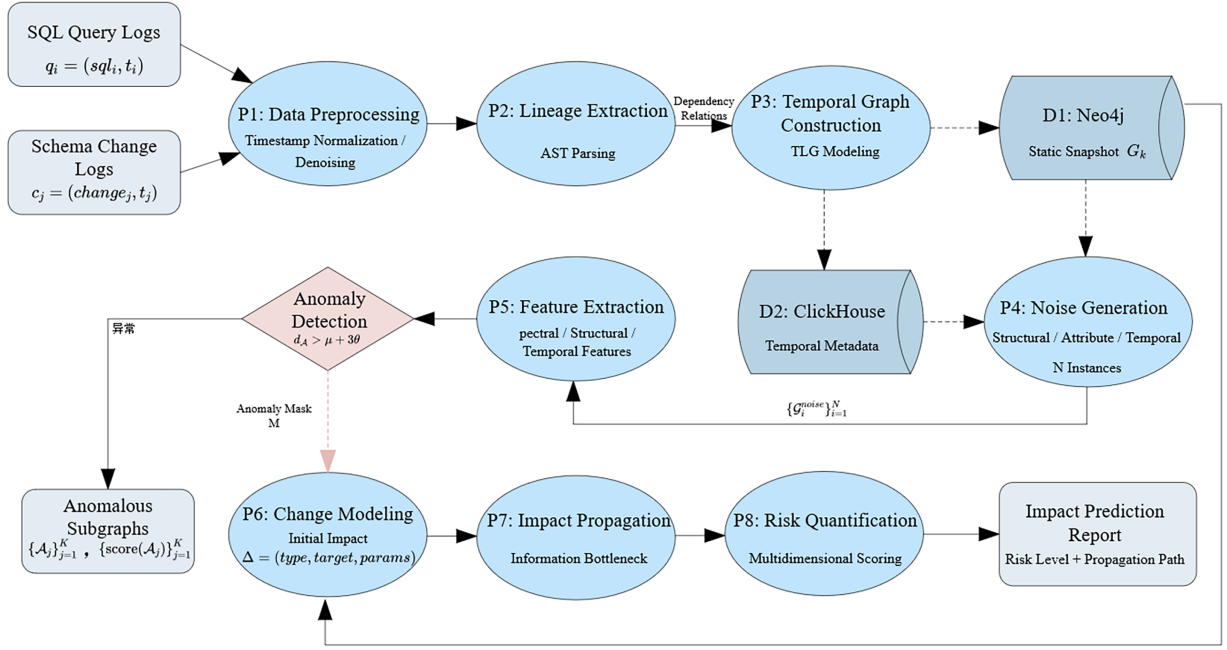


Figure 2: Data flow diagram.

The core of lineage extraction lies in parsing the Abstract Syntax Tree (AST) of SQL statements. For each SQL statement, the AST is recursively traversed to extract data flows from source tables and columns in the SELECT clause to target tables and columns in INSERT/UPDATE statements. Complex structures, such as nested queries and multi-table JOINS, are handled via a context stack to ensure that indirect dependencies are captured. When conflicts arise, such as type mismatches, probabilistic edges $e = (u, v, w)$ are generated, where the weight $w \in [0, 1]$ reflects the confidence level of the dependency.

The time complexity of the extraction algorithm is:

$$Time = \sum_{i=1}^N O(|AST_i|) \text{ with caching: } (1-h) \sum O(|AST_i|) \quad (1)$$

where N is the number of log entries and $|AST_i|$ denotes the size of the AST for the (i) -th query. By introducing a caching mechanism, the actual average time complexity becomes:

$$Time \approx (1-h) \sum O(|AST_i|) \quad (2)$$

where h represents the cache hit rate.

3.2.2 Temporal Graph Modeling and Storage

Based on the extracted lineage relationships, a multi-layer temporal graph structure is constructed. The timeline $[T_0, T_n]$ is divided into k snapshots according to business requirements, with each snapshot S_k corresponding to a static lineage graph $G_k = (V_k, E_k)$.

Temporal edges are constructed to connect adjacent snapshots, forming a complete evolution trajectory. For entities with the same name in consecutive snapshots, a temporal edge (v_k, v_{k+1}, Δ) is added, where $\Delta \in \{\text{STRUCT}, \text{SEMANTIC}, \text{STABLE}\}$ encodes the type of change: structural change, semantic change, or stability. The stability of each node is quantified using a normalized entropy measure. First, the raw entropy is computed as:

$$H_v = - \sum_{\delta \in \mathcal{D}} p_v(\delta) \log p_v(\delta) \quad (3)$$

where $\mathcal{D} = \{\text{STRUCT}, \text{SEMANTIC}, \text{STABLE}\}$ is the set of change types, $|\mathcal{D}| = 3$, and $p_v(\delta)$ is the frequency of type δ changes for node v within the observation window.

The normalized stability metric is defined as:

$$S_v = 1 - \frac{H_v}{\log |\mathcal{D}|} \quad (4)$$

with $s_v \in [0, 1]$. Lower entropy indicates higher node stability, while higher entropy signals frequent changes.

The Temporal Lineage Graph (TLG) is stored using a hybrid approach to balance query performance and storage cost. Static snapshots are stored in the Neo4j graph database, leveraging its native graph storage engine to optimize traversal performance. Temporal metadata is stored in the columnar database ClickHouse, supporting efficient time-range queries. The two storage layers are linked via node IDs, achieving a temporal query complexity of $O(\log T)$.

3.3 Anomaly Detection Module

3.3.1 Noise-Diversity-Based Detection Framework

The core idea of the noise diversity strategy is to generate diversified synthetic noise instances as bounded admissible variations around the observed TLG, thereby establishing a reference envelope for anomaly detection. This framework designs three complementary types of noise to cover different dimensions of the TLG. Here, “noise” denotes controlled variations used to characterize tolerable lineage changes, rather than arbitrary corruptions:

1. **Structural Noise.** Simulates topological variations by randomly adding or removing edges while maintaining graph connectivity.
2. **Attribute Noise.** Perturbs node and edge attributes. For continuous attributes such as the stability metric s_v , Gaussian noise $\mathcal{N}(0, \sigma^2)$ is added, with σ adaptively set to 10% of the original attribute standard deviation. For discrete attributes, such as node type, values are randomly replaced with another value of the same category with probability $q = 0.05$.
3. **Temporal Noise.** Specifically designed for TLG evolution, modifying temporal edges E_t through delays or reordering.

Based on these strategies, N noise instances $\{\mathcal{G}_i^{\text{noise}}\}_{i=1}^N$ are generated, where each instance is interpreted as a bounded admissible variation of the original temporal lineage graph and $N = \lceil \log_2 |V| \rceil + 100$. Let the Laplacian eigenvalues of the k -th noise graph be $\lambda^{(k)} = (\lambda_1^{(k)}, \lambda_2^{(k)}, \dots, \lambda_N^{(k)})$, and those of the original graph be $\lambda^{(\ell)}$. The spectral diversity condition requires that the spectral distance between any two noise instances satisfies:

$$\min_{k \neq \ell} \|\lambda^{(k)} - \lambda^{(\ell)}\|_2 \geq \tau \cdot \sigma_\lambda \quad (5)$$

where σ_λ is the standard deviation of the original graph spectrum, and τ is a distinguishability threshold, empirically set to 0.3–0.5.

3.3.2 Anomaly Identification and Purification

Based on the generated noise instances, anomalies are identified through geometric analysis in feature space, where the noise instances define a bounded reference envelope of admissible lineage variations. Three types of feature vectors are first extracted:

1. Spectral Features: Laplacian eigenvalues of the graph.
2. Structural Features: Degree distribution, clustering coefficients, and path-length distribution.
3. Temporal Features: Change frequency and stability metrics.

The feature vector $\mathbf{f} = [\mathbf{f}_{\text{spec}}, \mathbf{f}_{\text{local}}, \mathbf{f}_{\text{temp}}] \in \mathbb{R}^{50}$ is filtered via mutual information, retaining only those features with $MI > 0.1$.

For a candidate subgraph $\mathcal{A}$, the minimum distance between its feature vector $\mathbf{f}_{\mathcal{A}}$ and the convex hull of noise instances $\text{conv}(\{\mathbf{f}_i^{\text{noise}}\})$ is computed, where the convex hull is used as a geometric approximation of the admissible-variation envelope.

$$d_{\mathcal{A}} = \min_{\mathbf{x} \in \text{conv}} \|\mathbf{f}_{\mathcal{A}} - \mathbf{x}\|_2 \quad (6)$$

To address the computational complexity of convex-hull operations in high-dimensional space, an approximation strategy is adopted. According to the Johnson–Lindenstrauss lemma, random projection is used to reduce the dimensionality from $d = 50$ to m , where $m = O(\epsilon^{-2} \log N)$, while preserving pairwise distances among N points within a factor of $1 \pm \epsilon$.

$$\mathbf{f}' = R \cdot \mathbf{f} \quad (7)$$

where $R \in \mathbb{R}^{m \times d}$ is a random projection matrix. The convex-hull distance $d'_{\mathcal{A}}$ is then computed in the reduced-dimensional space. The anomaly score is defined according to the quantile rank of $d'_{\mathcal{A}}$ within the distance distribution of all noise instances. To further enhance robustness, a quantile-based threshold is employed:

$$\text{score}(\mathcal{A}) = \text{QuantileRank}(d'_{\mathcal{A}}) \quad (8)$$

The anomaly threshold is set at the 95th percentile of the distance distribution; that is, subgraph $\mathcal{A}$ is deemed anomalous when $d'_{\mathcal{A}}$ exceeds the distances of 95% of noise instances. Finally, the method outputs the anomalous subgraph set $\{\mathcal{A}_j\}_{j=1}^K$ together with their corresponding anomaly scores $\{\text{score}(\mathcal{A}_j)\}_{j=1}^K$.

The purification process produces an anomaly mask matrix $M \in \{0, 1\}^{|V| \times |V|}$, where $M_{ij} = 1$ indicates that edge (i, j) belongs to an anomalous pattern. The purification process constructs a confidence-weighted propagation graph rather than physically removing edges from the stored TLG. For edges identified as anomalous, a confidence gating mechanism is applied: for each anomalous edge e , a confidence score $c_e \in [0, 1]$ is defined, where $c_e = 0$ denotes a fully anomalous edge that is ignored during influence propagation; $c_e = 1$ denotes a verified normal edge that is fully retained; and $c_e \in (0, 1)$ denotes partially reliable edges whose effects are proportionally attenuated. Within the influence propagation module, the effective edge weight is updated as

$$w'_e = c_e \cdot w_e \quad (9)$$

where w_e is the original edge weight. This soft adjustment preserves graph connectivity while reducing the contribution of anomalous paths to influence estimation.

The confidence score c_e is computed from the anomaly score as

$$c_e = \max(0, 1 - (\text{score} - 2.5\sigma)/3\sigma) \quad (10)$$

For visualization, the presentation layer may optionally display only edges with $c_e > 0.3$. However, all computations are conducted on the complete weighted graph to prevent information loss. This design enables downstream modules to flexibly decide whether and how to incorporate anomalous paths.

3.4 Change Impact Prediction Module

In enterprise data-governance practice, database schema modifications are routine operations; however, each modification may trigger a cascade of responses across downstream data pipelines, business queries, and analytical reports. Traditional impact-analysis methods typically rely on manual expertise or simple dependency tracing, which makes it difficult to quantify the degree of impact or to identify indirect propagation paths. The challenge becomes more pronounced when anomalous patterns exist within the lineage graph—for example, abnormal cyclic dependencies or excessively long chains may artificially inflate the perceived impact scope, leading to misjudgment. To address these issues, this module performs precise impact analysis using the purified graph $\mathcal{G}$ derived from the anomaly-detection stage. Here, purification denotes confidence-weighted suppression of suspicious edges during influence propagation rather than physical deletion from the stored TLG. By reducing the contribution of anomalous paths, the module delivers more reliable and accurate assessments of change impact.

This module models the impact propagation process by introducing the Information Bottleneck (IB) theory. Unlike traditional approaches based on path counting or simple decay functions, the IB framework constrains the maximum amount of information that can be transmitted between nodes from an information-theoretic perspective, thereby preventing overestimation of impact. This mechanism is particularly suitable for SQL lineage scenarios, as data transformations fundamentally constitute information-processing workflows. The information-processing capacity of each node—whether a table or a view—determines the upper bound of its downstream impact propagation. For example, aggregation queries compress upstream information and thereby limit the extent to which changes can propagate downstream, whereas simple projection operations retain information almost entirely, resulting in more direct impact transmission.

Change modeling and initial impact estimation parse database DDL (Data Definition Language) statements into graph operations and identify the set of directly affected nodes $\mathcal{A}_0$ through a multidimensional scoring mechanism, which serves as the initial seed for impact propagation. The information-bottleneck-based propagation then simulates the diffusion of impact across the lineage graph using an iterative message-passing mechanism. Each iteration simultaneously incorporates distance decay $\rho(d)$ and the information-capacity constraint $I(s; v)$, ensuring both convergence of impact values and the plausibility of the propagation process. Finally, impact quantification and risk assessment evaluate the overall impact from structural, semantic, and performance dimensions, producing risk levels and key-path reports that provide quantitative support for change-management decisions. The complete workflow outputs the affected-node list, impact-severity scores, risk-level categorization, and the top-(K) critical propagation paths, and further supports interactive what-if analysis.

3.4.1 Change Modeling and Initial Impact Assessment

Accurate modeling of database schema modifications forms the foundation of impact prediction. This component converts various DDL operations into structured graph-based representations of schema changes and evaluates their initial impact on directly adjacent nodes. The process involves not only parsing the syntactic structure of SQL statements but also interpreting their semantic implications, thereby providing precise initial conditions for subsequent impact propagation.

Change modeling maps database schema modifications to graph operations within the TLG. The input interface adopts a triple form $\langle \mathcal{G}', M, \Delta \rangle$, where $\Delta = (type, target, params)$ is a structured representation of the modification extracted by an extended SQL parser from DDL statements such as *ALTER TABLE* or *DROP COLUMN*. Here, $type \in \{SCHEMA, CONSTRAINT, INDEX\}$ denotes the category of the modification, $target$ specifies the affected database object, and $params$ contains the detailed modification parameters. The system defines an initial severity value $\omega_0 \in [0, 1]$ for common modification patterns, which is dynamically adjusted via Bayesian updating based on the observed real-world impact of historical changes.

The schema modification is modeled as an information source s , and each node in the lineage graph is modeled as an information-processing unit. The state of a node v_i is represented by the triplet (I_i, C_i, O_i) , corresponding to its input information, processing capacity, and output information, respectively. The dependency strength between nodes is estimated using Normalized Pointwise Mutual Information (NPMI):

$$NPMI(v_i, v_j) = \frac{\log \left(\frac{p(v_i, v_j)}{p(v_i)p(v_j)} \right)}{-\log p(v_i, v_j)} \quad (11)$$

where $p(v_i)$ denotes the marginal probability that node v_i is accessed within a given time window, and $p(v_i, v_j)$ denotes their joint probability of being co-accessed within the same query. Laplace smoothing is applied to avoid zero-probability issues. The NPMI value lies within $[-1, 1]$, making it easier to interpret and constrain. The upper bound of the information bottleneck is set as:

$$(s; v) = \max(0, NPMI(s, v)) \quad (12)$$

The information propagation network $\mathcal{N} = (\mathcal{G}', W, \rho)$ extends the original lineage graph by incorporating a transmission-weight matrix (W) and a decay function $\rho: \mathbb{R}^+ \rightarrow [0, 1]$. The decay function is defined as

$$\rho(d) = \exp(-\alpha \cdot d) \quad (13)$$

where α is inversely proportional to the spectral radius $\lambda_{\max}(\mathcal{L})$ of the graph, ensuring convergence of information propagation.

For table-structure modifications, *ALTER TABLE* statements are parsed into changes in node attribute vectors, $\Delta v = v' - v$, where the attribute vector $v \in \mathbb{R}^{10}$ encodes key information such as data types, constraints, and index status. Changes in column types correspond to variations in the third dimension of the attribute vector, with impact weights obtained via a type-compatibility matrix $\mathcal{C} \in [0, 1]^{n_{types} \times n_{types}}$. Additions or deletions of constraints are mapped to binary changes in dimensions 5–7 of the attribute vector, with each constraint type assigned a distinct influence coefficient.

When a node v_i undergoes a modification, the weight of its outgoing edge (v_i, v_j) is updated as

$$w'_{ij} = w_{ij} \cdot \max(0, (1 - \beta \cdot \|\Delta v_i\|_2)) \quad (14)$$

where $\beta \in [0, 1]$ is the decay coefficient.

Directly affected nodes are identified using a priority-based one-hop neighborhood screening procedure. The procedure maintains a priority queue Q , in which candidate nodes are ordered according to the product of the corresponding edge weight and node importance. For each one-hop neighbor $t \in N(s)$ of the change source node s , three types of impact scores are computed:

1. Structural impact $I_{struct}(t)$, reflecting the topological importance:

$$I_{struct}(t) = w_{st} \cdot \deg(t) / \sum_{u \in N(s)} \deg(u) \quad (15)$$

2. Semantic impact $I_{sem}(t)$, considering schema similarity and access frequency:

$$I_{sem}(t) = \text{sim}(\text{schema}_s, \text{schema}_t) \cdot \text{freq}(s, t) \quad (16)$$

3. Performance impact $I_{perf}(t)$, evaluating changes in query execution cost:

$$I_{perf}(t) = \text{cost}(t) \cdot P(\text{cache_miss} | \text{change}(s)) \quad (17)$$

The initial impact value is determined via a weighted combination:

$$I_0(t) = \alpha_1 I_{struct}(t) + \alpha_2 I_{sem}(t) + \alpha_3 I_{perf}(t) \quad (18)$$

where the weight vector $\alpha = [0.3, 0.5, 0.2]$ is determined based on expert knowledge and historical case analysis. Nodes with $\theta = 0.1$ are included in the affected-node set $\mathcal{A}_0$, serving as the initial seed for the propagation process.

3.4.2 Influence Propagation Based on the Information Bottleneck

At the k -th iteration, the influence value of node v is updated as

$$I^{(k+1)}(v) = \min(1, \rho(d_s v) \cdot I(s; v) \cdot \sum_{u \in N_{in}(v)} w_{uv} \cdot I^{(k)}(u)) \quad (19)$$

where $\tilde{w}_{uv}$ is the precomputed normalized weight

$$\tilde{w}_{uv} = w_{uv} / \sum_{z \in N_{out}(u)} w_{uz} \quad (20)$$

This formulation consists of four key components. The weighted aggregation of neighboring influences ensures that influence is allocated according to dependency strength; the normalization factor prevents nodes with high degrees from being disproportionately amplified; the distance attenuation term $\rho(d_{sv})$ guarantees an exponential decay of influence over longer distances; and the mutual-information upper bound $I(s; v)$ imposes a theoretical limit on the maximum amount of information transmissible from an information-theoretic perspective.

For multi-path propagation, the Noisy-OR model is employed to fuse information from multiple sources. Suppose node v has K upstream neighbors $u_1, \dots, u_K$, whose influence propagation probabilities are $I_{u_1}, \dots, I_{u_K}$, with corresponding edge weights $\tilde{w}_{u_1 v}, \dots, \tilde{w}_{u_K v}$. The Noisy-OR aggregation is given by

$$I_v = 1 - \prod_{u \in \mathcal{N}(v)} (1 - I_u \cdot \tilde{w}_{uv}) \quad (21)$$

where $I_u = I^{(k)}(u)$ denotes the influence received from neighbor (u) .

The propagation algorithm adopts an asynchronous message-passing mechanism, where each node maintains a message buffer B_v and a state flag $state_v \in \{IDLE, ACTIVE, CONVERGED\}$. Active nodes send influence messages $msg = (src, value, path)$ to their downstream neighbors, and the receiving nodes update their states according to the Noisy-OR aggregation strategy. The convergence of this iterative process is governed by the attenuated propagation matrix rather than by the row-normalized weight matrix alone. Since $\tilde{W}$ is row-normalized, its spectral radius is $r(\tilde{W}) = 1$. By incorporating the one-step attenuation factor $\rho(1) = e^{-\alpha}$, the convergence condition becomes:

$$r(\rho(1)\tilde{W}) = \rho(1)r(\tilde{W}) = e^{-\alpha} < 1 \quad (22)$$

which is automatically satisfied through the design of the decay factor α :

$$\alpha = 1/(1 + \lambda_{max}(L)) \quad (23)$$

where $\lambda_{max}(L)$ denotes the largest eigenvalue of the graph Laplacian matrix.

The temporal dimension is handled using a sliding-window mechanism, which accounts for the cumulative influence over the most recent w time snapshots:

$$I_{temp}(v) = \frac{\sum_{t \in W} \gamma(t) \cdot I_t(v)}{\sum_{t \in W} \gamma(t)} \quad (24)$$

where $\gamma(t)$ is the temporal decay weight:

$$\gamma(t) = \exp(-\lambda \cdot (T - t)/\tau) \quad (25)$$

and τ denotes the characteristic time scale. The overall change-impact propagation process, including the change source, temporal snapshots, distance decay, information-bottleneck constraint, and asynchronous message-passing mode, is illustrated in Fig. 3.

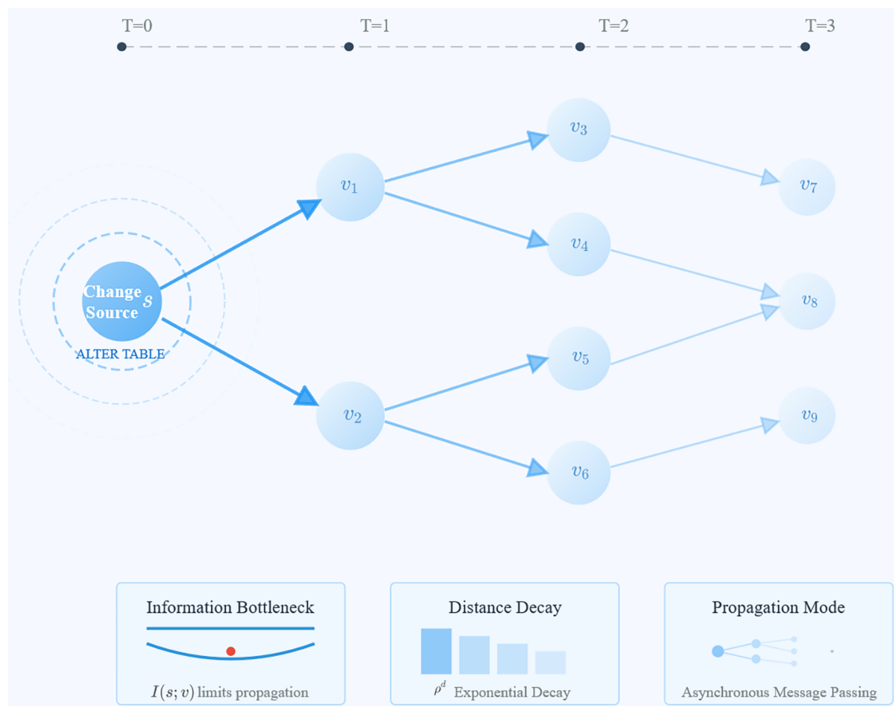


Figure 3: Change impact propagation model.

3.4.3 Influence Quantification and Risk Assessment

The degree of influence is quantified using a multidimensional scoring mechanism. The structural influence score S_{struct} measures the complexity of query rewriting and is computed based on edit distance and syntax-tree similarity:

$$S_{struct} = 1 - \frac{sim(AST, AST')}{|AST|} \quad (26)$$

where AST' denotes the query syntax tree adapted after the schema modification. When fields are removed or type inconsistencies arise, more rewriting operations are required, leading to a higher structural influence score.

The semantic influence score S_{sem} evaluates the extent of changes in business logic. By analyzing the semantic vectors of the query, the cosine similarity before and after the modification is computed:

$$S_{sem} = 1 - \cos(\mathbf{v}_{before}, \mathbf{v}_{after}) \quad (27)$$

and an additional weighting factor $w_{critical} = 2.0$ is assigned for queries involving critical business metrics.

The performance influence score S_{perf} is derived from the cost estimation of the query optimizer. The execution-plan cost before the modification, C_{before} , is collected, and the post-modification cost C_{after} is simulated. The relative change is computed as

$$S_{perf} = \frac{|C_{after} - C_{before}|}{C_{before}} \quad (28)$$

When indexes become invalid or join conditions are altered, performance impacts may differ by several orders of magnitude.

The overall influence score is computed using a weighted geometric mean:

$$S_{total} = (S_{struct}^{\omega_1} \cdot S_{sem}^{\omega_2} \cdot S_{perf}^{\omega_3})^{\frac{1}{\omega_1 + \omega_2 + \omega_3}} \quad (29)$$

where the weight vector $\omega = [0.3, 0.5, 0.2]$ can be adjusted according to business priorities. The use of geometric averaging ensures that severe impact in any individual dimension is adequately reflected in the final score.

Risk levels are categorized based on the influence distribution of historical schema changes, where kernel density estimation is used to determine the corresponding quantiles: high risk ($S_{total} > Q_{0.9}$), medium risk ($Q_{0.7} < S_{total} \leq Q_{0.9}$), and low risk ($S_{total} \leq Q_{0.7}$). Considering the historical stability of nodes, the risk rating is down-weighted for nodes with frequent modifications to avoid excessive alerts:

$$R_{adjusted} = R_{base} \cdot \exp(-\lambda \cdot change_freq) \quad (30)$$

An improved Dijkstra algorithm is used to identify critical paths, where edge weights are defined as $-\log(1 - impact)$. Finding the “shortest path” corresponds to identifying the path with the maximum cumulative influence. The output includes the top-K critical paths along with their cumulative influence values, providing guidance for prioritizing optimization efforts.

The time complexity of this module is $O(|V| \cdot |E| \cdot k)$, where k denotes the number of iterations required for convergence. Through parallelized message passing and incremental computation optimizations, a single prediction on a lineage graph with millions of nodes can be completed within one second, meeting the real-time requirements of interactive analysis.

3.5 Theoretical Analysis

3.5.1 Applicability of Noise Diversity to Graph Structures

Let $\mathcal{G} = (\mathcal{V}, E)$ denote the original lineage graph, $\Omega_{\text{adm}}(\mathcal{G})$ denote the set of admissible lineage variations generated from the observed Temporal Lineage Graph $\mathcal{G}$ by the bounded structural, attribute, and temporal perturbation rules defined in Section 3.3.1. The generated noise graphs $\{\mathcal{G}_i^{\text{noise}}\}_{i=1}^N$ are finite samples from this admissible variation set. Let $\lambda^{(i)}$ denote the Laplacian spectrum of $G_{\text{noise}}^{(i)}$. The spectral diversity condition requires that

$$\min_{i \neq j} \|\lambda^{(i)} - \lambda^{(j)}\|_2 \geq \delta \sigma_\lambda \quad (31)$$

where σ_λ is the standard deviation of the spectrum of the observed graph and δ is the distinguishability threshold. This condition prevents the generated noise instances from collapsing into nearly identical graph perturbations and provides a non-degenerate reference set for admissible lineage evolution.

Let $\phi(\cdot) \in \mathbb{R}^d$ denote the implemented feature mapping constructed from the selected spectral, structural, and temporal descriptors in Section 3.3.2, where $d = 50$. Define the ideal admissible envelope and its finite-sample approximation as

$$\mathcal{C}^* = \text{conv} \{ \phi(H) : H \in \Omega_{\text{adm}}(\mathcal{G}) \}, \mathcal{C}_N = \text{conv} \left\{ \phi \left(G_{\text{noise}}^{(i)} \right) \right\}_{i=1}^N \quad (32)$$

The applicability of the finite noise set depends on its ability to approximate the admissible envelope. Let the empirical envelope satisfy the coverage condition

$$\sup_{\mathbf{z} \in \mathcal{C}^*} \text{dist}(\mathbf{z}, \mathcal{C}_N) \leq \varepsilon_{\text{cov}} \quad (33)$$

where ε_{cov} denotes the approximation error of the empirical reference envelope. The spectral diversity condition in Eq. (31) supports non-degenerate sampling, whereas Eq. (33) specifies the sufficient coverage condition required for anomaly discrimination.

For an admissible candidate subgraph A , $\phi(A) \in \mathcal{C}^*$, and therefore

$$\text{dist}(\phi(A), \mathcal{C}_N) \leq \varepsilon_{\text{cov}} \quad (34)$$

For an anomalous subgraph A_{an} , suppose that its feature representation has a sufficient margin from the admissible envelope:

$$\text{dist}(\phi(A_{\text{an}}), \mathcal{C}^*) \geq \gamma, \gamma > 2\varepsilon_{\text{cov}} \quad (35)$$

Then, by the triangle inequality,

$$\text{dist}(\phi(A_{\text{an}}), \mathcal{C}_N) \geq \gamma - \varepsilon_{\text{cov}} \quad (36)$$

Consequently, a decision threshold can be selected from

$$\varepsilon_{\text{cov}} < \tau < \gamma - \varepsilon_{\text{cov}} \quad (37)$$

so that admissible lineage variations and sufficiently separated anomalous subgraphs are distinguished by their distances to the empirical noise envelope.

To reduce the computational cost of convex-hull distance estimation, let $R \in \mathbb{R}^{m \times d}$ be the random projection matrix used in [Section 3.3.2](#). If R satisfies the subspace distance-preservation condition on the span of the candidate feature vector and the noise-reference feature vectors, then, with probability at least $1 - \delta_{RP}$,

$$(1 - \varepsilon_{RP}) \text{dist}(\phi(A), \mathcal{C}_N) \leq \text{dist}(R\phi(A), R\mathcal{C}_N) \leq (1 + \varepsilon_{RP}) \text{dist}(\phi(A), \mathcal{C}_N) \quad (38)$$

Thus, random projection preserves the distance-based separation between candidate subgraphs and the empirical reference envelope up to a controlled distortion.

For a normalized bounded reference statistic $h(G_{\text{noise}}^{(i)}) \in [0, 1]$, a conservative finite-sample sufficient condition for the stability of its empirical estimate is given by:

$$N \geq \frac{2}{\varepsilon_N^2} \ln \left(\frac{2}{\delta_N} \right) \quad (39)$$

For the enterprise graph with $|V| = 32,456$, the graph-size-aware rule in [Section 3.3.1](#) yields $N = 115$. With $\varepsilon_N = 0.26$ and $\delta_N = 0.05$, [Eq. \(39\)](#) gives $N \geq 110$. Therefore, the operational setting $N = 115$ satisfies the stated finite-sample sufficient condition. Here, ε_N is used for the finite-sample reference-statistic bound and is distinct from the random-projection distortion parameter ε_{RP} .

Therefore, under bounded perturbation, spectral diversity, empirical-envelope coverage, and feature-space margin conditions, the proposed structural, attribute, and temporal noise families form a non-degenerate reference envelope for admissible temporal lineage evolution. Candidate subgraphs that are sufficiently separated from this envelope can be identified as anomalous, and this separation is preserved with high probability after random projection.

3.5.2 Complexity Analysis of Temporal Lineage Queries

A time-indexed query algorithm is designed to efficiently retrieve the historical dependencies of nodes within a specified temporal interval. For a temporal lineage graph (TLG) containing n nodes over a time span of T , the query algorithm leverages a balanced binary search tree (BST) index. Let the temporal layers be an ordered sequence $\mathcal{T} = \{t_1 < t_2 < \dots < t_T\}$, with the BST height $\log T$. Locating the time interval $[t_1, t_2]$ requires $O(\log T)$ comparisons; subsequently, dependency collection traverses only k nodes (the number of upstream dependencies). Therefore, the overall time complexity is $O(\log T + k)$.

Any algorithm must access at least $\Omega(\log T)$ temporal layers to distinguish among T potential change points, in addition to an $\Omega(k)$ output cost. Entropy analysis indicates that a query requires at least $\log T$ bits of information. To establish this lower bound, we construct an adversarial scenario: consider a TLG in which dependency changes are hidden at some time point $t^* \in [1, T]$. For each query q_i by the algorithm, if $t^* \notin q_i$, the algorithm returns “no change”; if $t^* \in q_i$ and $|q_i| > T/2^i$, t^* is shifted to one half of q_i . In this case, the algorithm must perform at least $\log_2 T$ queries to locate t^* .

Therefore, any deterministic algorithm has a worst-case time complexity of $\Omega(\log T + k)$, consisting of an $\Omega(\log T)$ cost for locating the relevant temporal interval and an unavoidable $\Omega(k)$ cost for reporting the k dependencies. By Yao’s minimax principle, the $\Omega(\log T)$ search lower bound also applies to the expected running time of randomized algorithms. Combining this result with the independent output cost yields an expected lower bound of $\Omega(\log T + k)$.

3.5.3 Probabilistic Bounds on Change Influence Propagation

Based on information-theoretic and attenuated-random-walk analyses, let a change occur at node s , and let t be a downstream node at distance d . We impose two conservative constraints on the affected probability: a path-attenuation constraint determined by ρ^d , and an information-capacity constraint determined by $I(s; t)$. Their joint upper bound is:

$$P_{affected}(t | change(s)) \leq \min\{\rho^d, I(s; t)\} \quad (40)$$

where $\rho = e^{-\alpha} \in (0, 1)$ is the one-step attenuation factor, and $I(s; t) \in [0, 1]$ denotes the mutual information between nodes s and t .

The influence propagation is modeled as a random walk on a Markov chain. Let $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$ denote the row-normalized transition matrix. Since $\mathbf{P}$ is row-stochastic, its spectral radius is $r(\mathbf{P}) = 1$. For brevity, let $\rho = \rho(1) = e^{-\alpha} \in (0, 1)$ denote the one-step attenuation factor. The effective propagation matrix is therefore $\rho\mathbf{P}$, whose spectral radius is smaller than one. The d -step attenuated transition from s to t satisfies

$$P_{s \rightarrow t}^{(d)} = \left[(\rho\mathbf{P})^d \right]_{st} = \rho^d \left[\mathbf{P}^d \right]_{st} \leq \rho^d \quad (41)$$

To explain information loss along a propagation path, let Y denote an intermediate node in the Markov chain $s \rightarrow Y \rightarrow t$. At the theoretical level, the underlying mutual-information interpretation of the information-capacity term obeys the Data Processing Inequality:

$$I(s; t) \leq \min\{I(s; Y), I(Y; t)\} \quad (42)$$

Accordingly, the operational influence score should remain nonnegative and should not exceed either the path-attenuation bound or the normalized information-capacity bound:

$$0 \leq B_{s,t}^{(d)} \leq \min\{\rho^d, I(s; t)\} \quad (43)$$

To simultaneously incorporate distance attenuation and information capacity while satisfying Eq. (43), the following multiplicative bounded influence score is adopted:

$$B_{s,t}^{(d)} = \rho^d I(s; t) \quad (44)$$

This multiplicative quantity is used as an operational influence score rather than as a universally calibrated probability upper bound. Because both ρ^d and $I(s; t)$ lie within $[0, 1]$, Eq. (44) automatically satisfies the boundedness requirement in Eq. (43). For the source node itself, the score equals one when $d = 0$ and $I(s; s) = 1$; as d increases, the score approaches zero; and when $I(s; t) = 0$, no information-related influence is assigned to node t . For temporal lineage graphs, the time-weighted aggregation defined in Eqs. (24) and (25) is applied, and the boundedness property remains valid.

4 Experiments and Analysis

This section evaluates the practical value of temporal lineage modeling, the behavior of the noise-diversity-based anomaly detector under available reference annotations, and change-impact prediction under bounded propagation. The evaluation adopts a task-specific multi-source design: temporal modeling, change-impact prediction, root-cause localization, and scalability results are reported on the enterprise corpus, whereas anomaly-detection results are reported on a joint anomaly-evaluation corpus.

4.1 Experimental Setup

The experimental evaluation adopts a task-specific multi-source design. The anonymized enterprise corpus consists of timestamped SQL query logs and schema-change records collected over an 18-month period. After preprocessing, the enterprise Temporal Lineage Graph contains 32,456 nodes and 185,743 edges across 18 snapshots. This corpus is used for temporal lineage modeling, change-impact prediction, root-cause localization, and scalability evaluation.

TPC-DS-derived temporal lineage instances with controllable structural, semantic, and temporal anomaly patterns are used as a labeled synthetic component of anomaly-detection evaluation. The joint anomaly-evaluation corpus combines the TPC-DS-derived labeled anomaly instances with the available enterprise anomaly-reference instances. A total of 324 real schema changes are used for change-impact evaluation, and the corresponding affected-entity reference sets are constructed through pre- and post-change snapshot replay together with manual verification.

4.2 Quantitative Value of Temporal Lineage Graphs

On enterprise data, the temporal lineage graph (TLG) provides interpretable time-slice analyses of evolution trajectories over an 18-month window. As summarized in Table 1, a total of 2743 fields underwent evolution during the observation period, with 46.9% corresponding to substantive changes in computational logic, 34.9% involving dependency reordering, and the remaining 18.2% reflecting data type modifications. Compared with static lineage, the TLG enables structured answers to “when a change occurred and what changed.” In root-cause localization tasks, this reduces the average localization time from 45 to 29 min, a 35.6% reduction. In a setting where the most recent 12 months of data are used to predict the next 3 months of evolution, the TLG achieves an Evolution Path Accuracy (EPA) of 85.7% and a 3-month rolling prediction accuracy of 64.2%, substantially outperforming the random baseline of 11.1%. Fig. 4 presents the distribution of edge-level change types over time as a heatmap. Semantic changes consistently exceed structural changes in proportion, explaining the limitations of traditional methods that rely solely on structural differences for historical replay and future extrapolation tasks.

Fig. 5 illustrates the distribution of node stability based on normalized entropy. Among the 32,456 nodes, approximately 30% are highly stable, primarily corresponding to dimension and base fact tables; over half of the nodes exhibit medium stability, mostly associated with business intermediate layers; and fewer than 20% have low stability, requiring focused monitoring within the change windows. The corresponding edge-level statistics are shown in Fig. 6. Among the 185,743 edges, stable (STABLE) edges account for 43.0%, semantic changes (SEMANTIC) for 34.0%, and structural changes (STRUCT) for 23.0%.

Table 1: Comparison of temporal modeling performance.

Metric	Static Method	TLG	Notes
Number of identified evolving fields	0	2743	18-month window
Evolution Path Accuracy (EPA)	0.00%	85.70%	Average Jaccard
Root-cause localization time	45 min	29 min	Ticket-level localization
3-month prediction accuracy	11.10%	64.20%	Extrapolated from previous 12 months

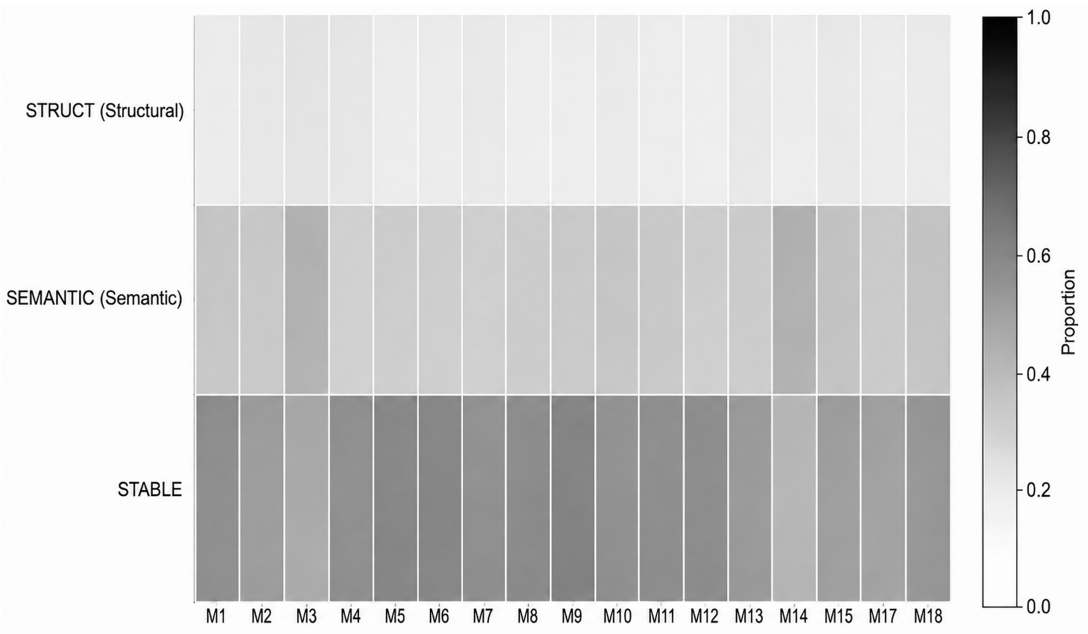


Figure 4: TLG evolution heatmap (time vs. change type proportions).

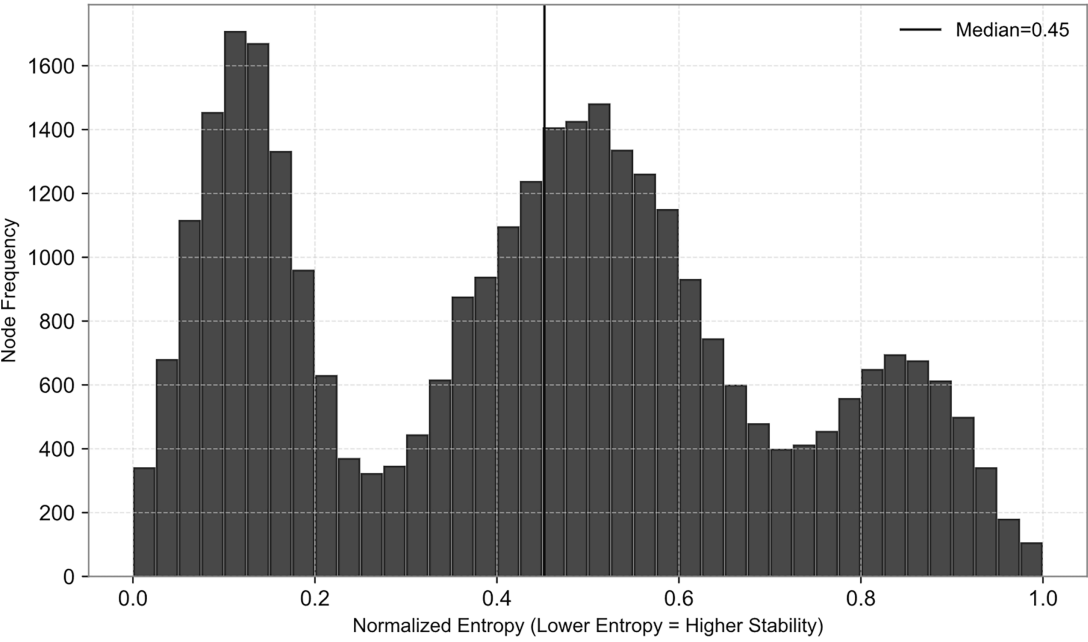


Figure 5: Distribution of node stability.

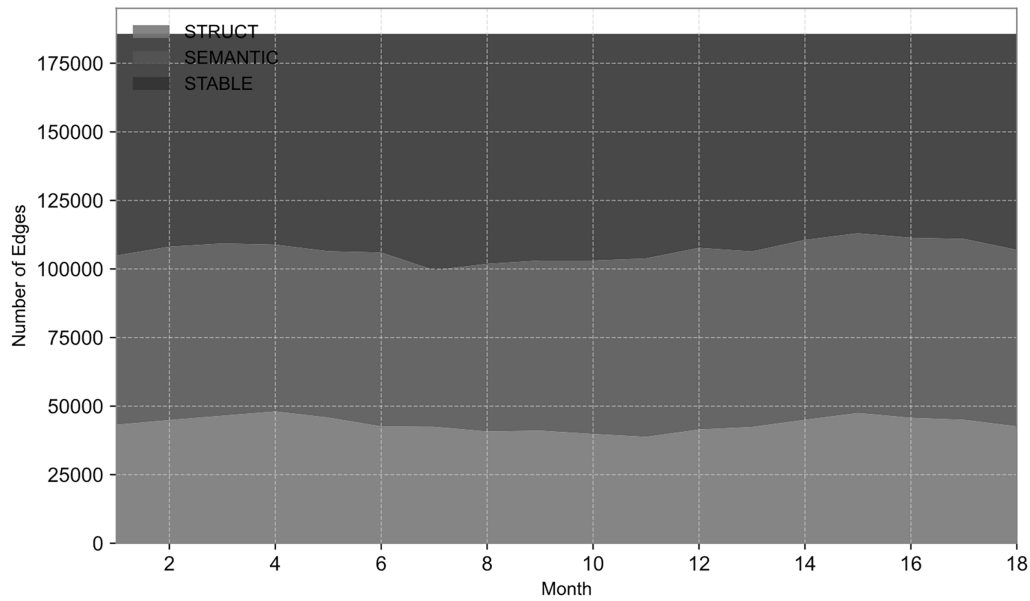


Figure 6: Stacked area chart of edge-level change types over time.

4.3 Effectiveness of Unsupervised Anomaly Detection

The anomaly-detection evaluation is conducted on the joint anomaly-evaluation corpus described in Section 4.1. In the spectral domain, the variance coverage among noise instances reaches 0.83, exceeding the empirical distinguishability threshold and providing sufficient diversity for reference-envelope construction. The anomaly decision boundary is determined using the empirical 95th percentile of the distance distribution generated from the reference noise instances. Fig. 7 presents the precision-recall curves of the proposed detector at the node, edge, and subgraph levels. The corresponding AUC-PR values are 0.720, 0.763, and 0.833, respectively.

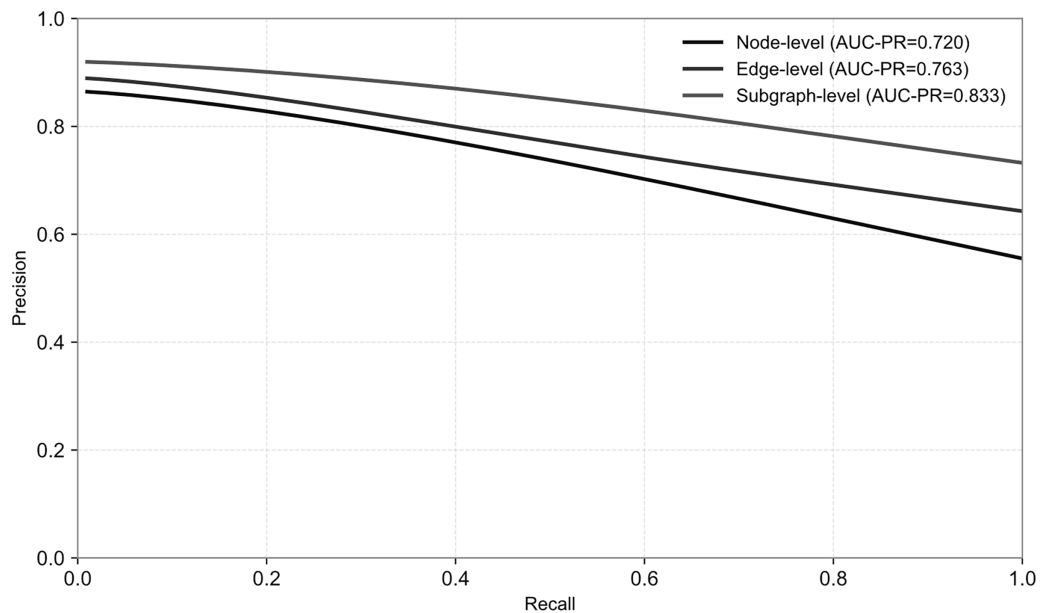


Figure 7: Anomaly detection pr curve (node/edge/subgraph).

On enterprise data, the method identified 1183 potential anomalies, of which 993 were confirmed as true anomalies through manual cross-validation. This yields an overall precision of 83.9%, recall of 88.7%, and F1-score of 0.863. Breakdowns by anomaly type—such as cyclic dependencies, abnormally long chains, dependency mutations, frequent changes, and type mismatches—are summarized in Table 2. Notably, the average length of anomalous long chains reaches 14.3 hops, substantially exceeding the 4.7-hop length of normal lineage paths. This observation highlights a potential risk of “overestimation at long distances” in subsequent influence propagation analyses.

Table 2: Performance of anomaly detection by category.

Anomaly Type	Detected	Confirmed	Precision	Recall	F1 Score
Cyclic Dependency	89	73	82.00%	87.30%	0.846
Excessive Chain Length	156	134	85.90%	91.20%	0.885
Dependency Mutation	387	312	80.60%	84.70%	0.826
Frequent Changes	234	198	84.60%	89.30%	0.869
Type Mismatch	317	276	87.10%	92.40%	0.897
Overall	1183	993	83.90%	88.70%	0.863

To evaluate the complementarity of the three noise types, systematic ablation experiments were conducted on the same evaluation set. The total number of true positives was normalized to 1120, corresponding to the ground-truth scale for an overall recall of 88.7%. Under this setting, the results for all noise strategies combined were: TP = 993, FP = 190, and FN = 127, consistent with the overall Precision, Recall, and F1. Performance comparisons across the ablation configurations are shown in Table 3. It can be observed that the “structure + temporal” combination significantly outperforms other pairwise combinations, and incorporating all three noise types further increases the F1-score to 0.863.

Table 3: Ablation study on noise strategies.

Noise Strategy	TP	FP	FN	Precision	Recall	F1 Score
Structure Only	626	220	494	73.99%	55.89%	0.637
Attribute Only	640	315	480	67.02%	57.14%	0.617
Temporal Only	760	253	360	75.02%	67.86%	0.713
Structure + Attribute	780	246	340	76.02%	69.64%	0.727
Structure + Temporal	920	202	200	82.00%	82.14%	0.821
Attribute + Temporal	880	220	240	80.00%	78.57%	0.793
All Combined	993	190	127	83.90%	88.70%	0.863

4.4 Change Influence Prediction and Bounded Propagation

A total of 324 real changes were observed over the 18-month period. Among the selected graph-based baselines, the proposed method achieves the highest reported Top-K ranking metrics while maintaining an online single-query latency of 1.2 s. Compared with path counting, PageRank, and Monte Carlo approaches, the method achieves $P@5 = 0.823$, $R@10 = 0.812$, and $NDCG@10 = 0.847$, while maintaining an online single-query latency of 1.2 s. Although path counting is faster, it ignores capacity bounds, resulting in a $P@5$ of only 0.487 and a tendency to overestimate the influence scope. PageRank is limited by its sensitivity to global structure and underperforms in lineage semantic contexts. Monte Carlo offers greater expressiveness

but requires 2.1 s per query, as summarized in Table 4. Fig. 8 shows the NDCG@K curves as K varies, indicating that the proposed method's advantage is most pronounced for $K \leq 10$, aligning well with practical requirements for localization and risk mitigation.

Table 4: Comparison of change impact prediction methods.

Method	Changes	P@5	P@10	R@10	NDCG@10	Online Query Time (s)
Path Counting	324	0.487	0.423	0.521	0.542	0.3
PageRank	324	0.671	0.598	0.642	0.693	0.8
Monte Carlo	324	0.734	0.689	0.712	0.758	2.1
Proposed Method	324	0.823	0.794	0.812	0.847	1.2

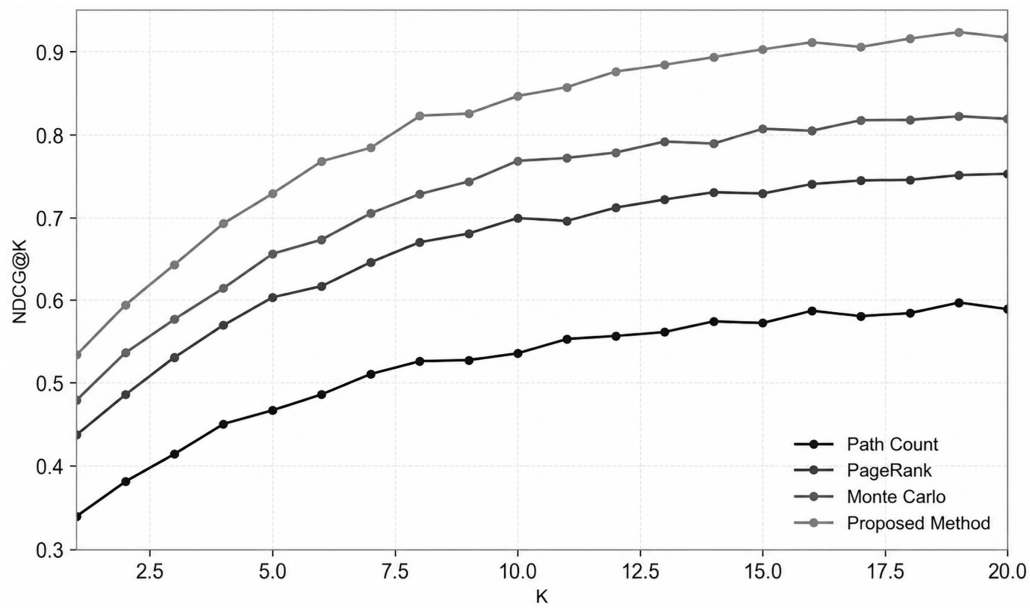


Figure 8: Influence prediction NDCG@K curve.

From the decomposition of propagation distances, the method achieves a one-hop accuracy of 91.3%, two-hop accuracy of 78.6%, and three hops or more of 62.4%, exhibiting a decreasing trend consistent with path decay and information upper bounds, as detailed in Table 5. Fig. 9 presents a scatter plot of the distribution of convergence iterations vs. spectral radius–decay relationships. Within the linear region, the decay coefficient α satisfies the convergence constraint $\alpha < 1/\rho$, with an average of 4.7 iterations to converge and a maximum of 8 iterations.

Table 5: Statistics of multi-hop impact identification.

Distance	Avg. Identified Nodes	Accuracy
1-hop	23.7	91.30%
2-hop	47.2	78.60%
≥ 3 -hop	59.4	62.40%

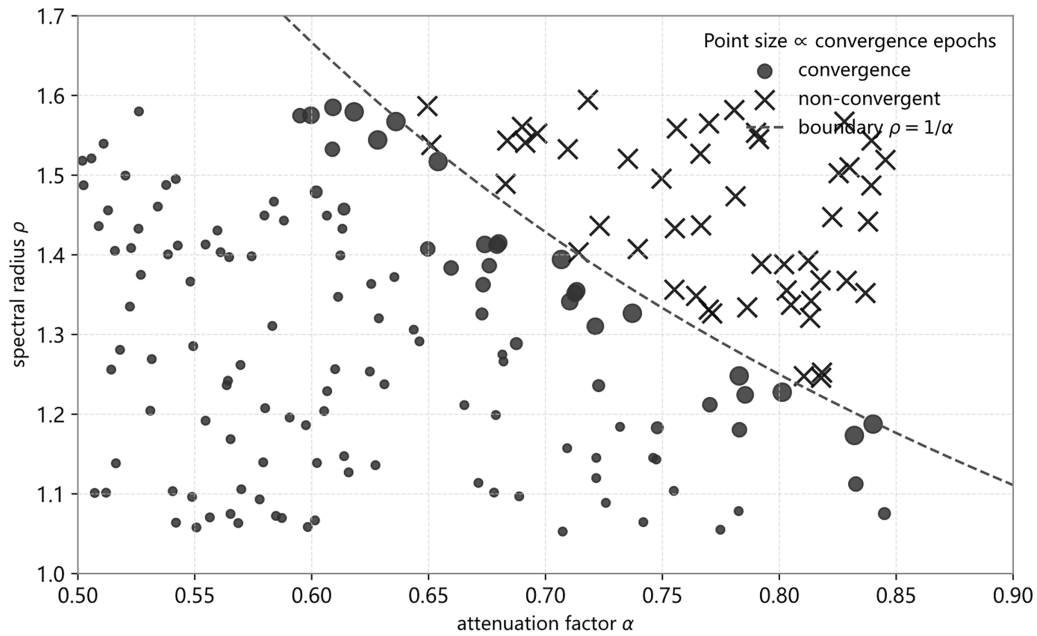


Figure 9: Scatter plot of spectral radius vs. attenuation, with point size indicating convergence rounds.

As shown in Table 6, the impact analysis by change type indicates that DROP COLUMN and TYPE-CHANGE operations exhibit higher destructiveness and deeper propagation depth. ALTER TABLE shows a moderate impact range, while ADD CONSTRAINT and CREATE INDEX primarily affect availability and performance, posing lower risks to structural correctness. Fig. 10 presents a visualization of the critical path for a DROP COLUMN operation. The path weights aggregated via the Noisy-OR model show significant decay within 3–4 hops, consistent with the theoretical expectations of bounded propagation.

Table 6: Impact scope and destructiveness across change types.

Change Type	Count	Avg. Affected Nodes	Impact Depth	Destructiveness Ratio
ALTER TABLE	147	47.3	3.2 hops	31%
DROP COLUMN	43	72.8	4.1 hops	86%
ADD CONSTRAINT	67	12.4	1.8 hops	12%
CREATE INDEX	39	5.7	1.2 hops	0%
TYPE CHANGE	28	38.6	2.9 hops	57%

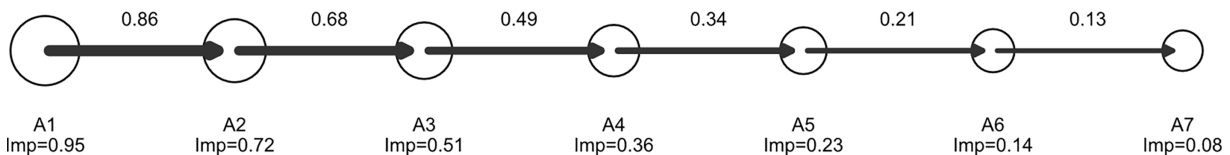


Figure 10: Key path visualization for DROP COLUMN change.

Interpretation of Structural and Semantic Changes. In the TLG, a structural change is assigned when the schema-level existence, field definition, or dependency topology of an object changes. For example, a DROP COLUMN operation removes a field from the schema and may invalidate downstream views, transformation

queries, and reports that directly reference the removed field. The critical path shown in Fig. 10 is therefore a structural-change case, in which the impact is triggered by the disappearance of a schema element and the resulting breakage of downstream references. A semantic change is assigned when the external table and field interfaces remain available but the SQL computation logic changes. In this case, downstream SQL statements may remain executable, but dashboards, reports, and models can produce results with a changed business interpretation. Thus, STRUCT captures schema or dependency breakage, whereas SEMANTIC captures logic evolution that may preserve syntactic executability while changing analytical meaning.

4.5 Scalability and Engineering Overhead

To evaluate the scalability of the method, the enterprise dataset was sampled at 1%, 10%, 50%, and 100%, and the end-to-end overhead and memory usage trends were recorded, as summarized in Table 7. In the offline stage, processing time scales approximately linearly with the number of edges. For the full dataset containing 32,456 nodes and 185,743 edges, lineage construction and anomaly detection require 127 and 37 s, respectively. Online single-impact queries remain at the second level. Fig. 11 presents a logarithmic-scale fit of construction and detection time vs. data size, with a coefficient of determination above 0.95, aligning with engineering expectations.

Table 7: Scalability and resource overhead.

Sampling Ratio	Nodes	Edges	Construction Time (s)	Detection Time (s)	Peak Memory (MB)
1%	325	1857	0.3	0.1	12
10%	3246	18,574	4.7	1.3	134
50%	16,228	92,872	43	14	1331
100%	32,456	185,743	127	37	3100

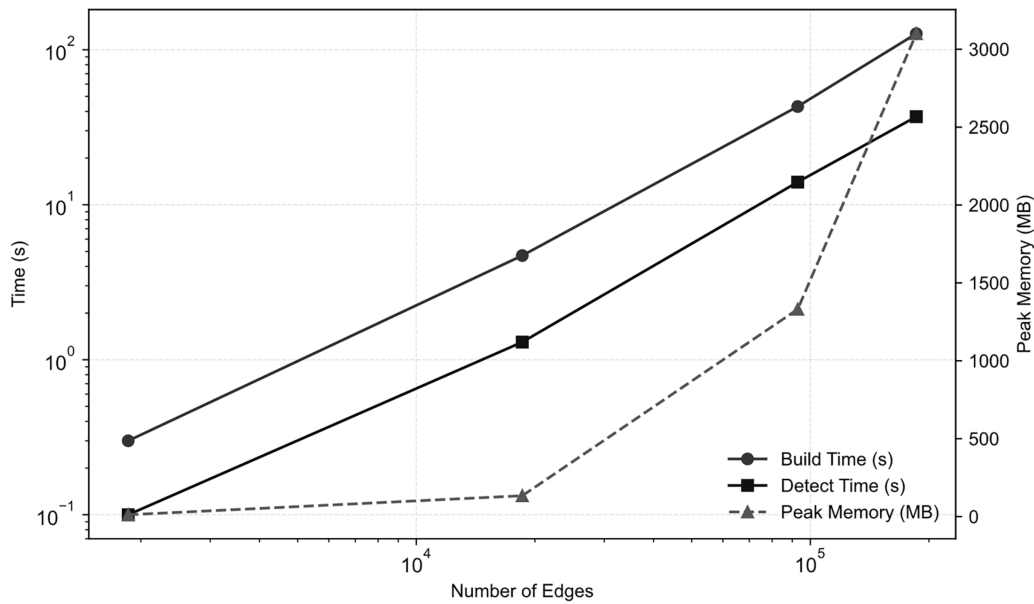


Figure 11: Scaling overhead curve.

4.6 Discussion and Threats

An overview of failure patterns is summarized in Table 8. Sparse graphs with an average degree below 1.5 exhibit a marked decrease in detection F1 due to insufficient statistical features. Pattern instability during periods of high-frequency changes reduces the separability of temporal features. Cross-system dependencies are constrained by non-uniform metadata standards, and deep nesting introduces parsing complexity and exponential growth of propagation paths. Parameter sensitivity analyses and default settings are presented in Table 9 and Figs. 12 and 13. Variations in the time window within 5–10 days result in fluctuations of no more than 2% in performance; using 100–150 noise samples affects the F1-score by less than 3%; decay coefficients α in the range 0.5–0.8 reliably satisfy the convergence constraint. The number of noise instances primarily controls the coverage of the empirical reference envelope. The connection between the number of noise instances and random projection is indirect. According to the Johnson–Lindenstrauss relation, the required projection dimension satisfies $m = O(\epsilon_{\text{RP}}^{-2} \log N)$. Therefore, increasing the number of noise instances increases the required projection dimension only logarithmically. Accordingly, the default setting $N = 115$ balances empirical-envelope coverage and runtime, while its adequacy is further verified by the finite-sample sufficient condition in Section 3.5.1. Adopting the 95th-percentile threshold approximately 2.5σ for anomaly detection provides a robust balance between precision and recall.

Table 8: Typical failure scenarios and primary causes.

Scenario	Occurrences	Failure Rate	Primary Cause
Sparse Graph (avg. degree < 1.5)	127	38%	Insignificant statistical patterns; indistinct contrastive boundaries
High-Frequency Changes (>100/day)	23	48%	Unstable evolution patterns; degradation of temporal features
Cross-System Dependencies	89	57%	Inconsistent metadata formats; missing mappings
Deep Nesting (>5 levels)	34	27%	High parsing complexity; path explosion

Table 9: Sensitivity of key parameters and basis for selection.

Parameter	Default Value	Optimal Range	F1 Variation	Rationale for Selection
Time Window W	7 days	[5, 10]	$\pm 2\%$	Aligns with weekly business cycles
Number of Noise Instances N	115	[100, 150]	$\pm 3\%$	Trade-off between JL distance preservation and runtime
Decay Factor α	0.66	[0.5, 0.8]	$\pm 5\%$	Convergence constraint $\alpha < 1/\rho$
Anomaly Threshold	2.5σ	$[2\sigma, 3\sigma]$	$\pm 8\%$	Precision–recall balance and robustness

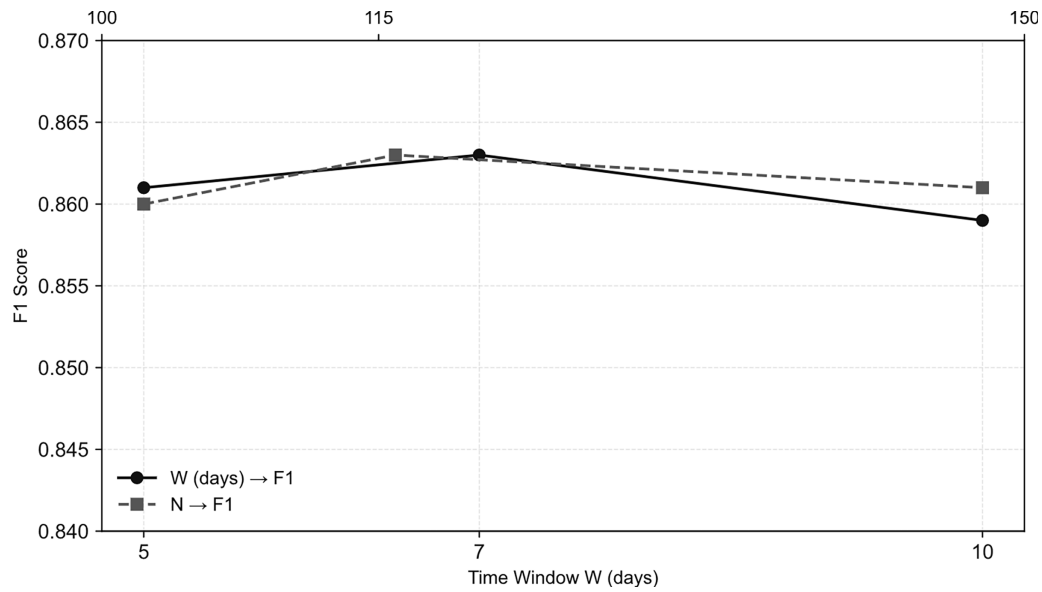


Figure 12: Parameter sensitivity: univariate analysis of W and N.

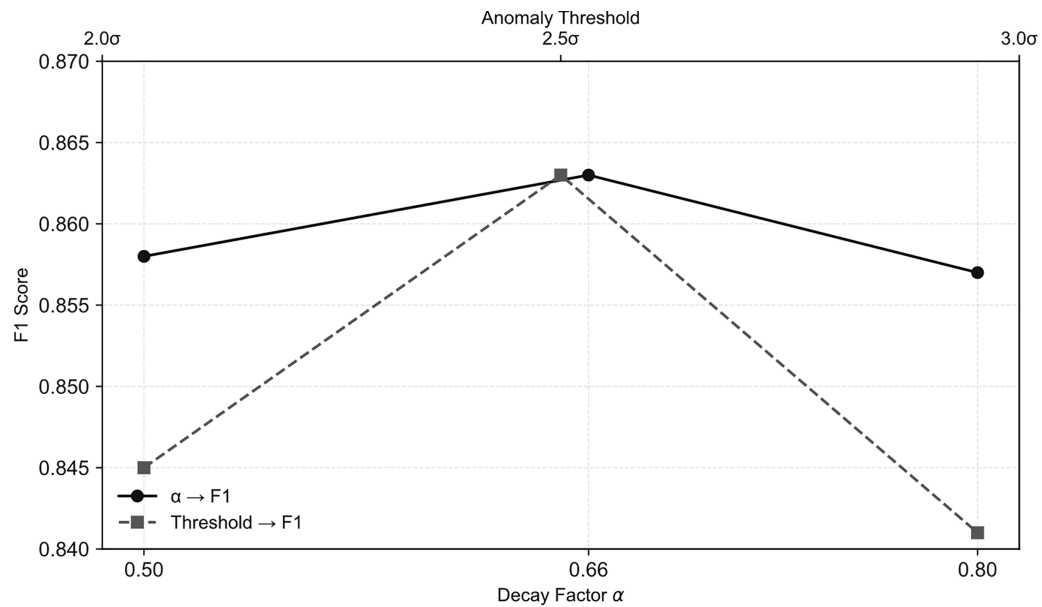


Figure 13: Parameter sensitivity: performance vs. α and threshold.

Inconsistent metadata and missing mappings may prevent reliable cross-system entity alignment, causing dependencies to be absent from the TLG or represented with reduced confidence. Future work will investigate canonical identifiers and confidence-aware cross-system mapping to mitigate this limitation.

Evaluation Scope and Future Validation. The current anomaly-detection evaluation focuses on the complementarity of the proposed structural, attribute, and temporal noise families. Future work will establish a unified temporal SQL-lineage benchmark with task-aligned inputs, common change-impact definitions, and consistent evaluation criteria. Such a benchmark will enable systematic comparison with adapted external

anomaly detectors, including LOF, DOMINANT, NetWalk, and StrGNN, as well as deterministic lineage-traversal baselines. Future validation will further examine the transferability of the proposed analytical layer across enterprise environments with different workload compositions and SQL-query complexities, and its integration with event-oriented lineage ecosystems such as OpenLineage, Marquez, and DataHub.

Future evaluations will also strengthen the reference-annotation protocol through independently sampled control subgraphs, multi-reviewer assessment, and blinded or semi-blinded verification where feasible, together with inter-annotator agreement statistics. These extensions will provide a more rigorous assessment of the generalizability and annotation reliability of the anomaly-detection and change-impact results.

5 Conclusion

This work addresses the core challenges of timeliness, accuracy, and interpretability in SQL lineage analysis for enterprise data governance. We propose a noise-aware, temporal SQL lineage graph (TLG) modeling and change influence prediction method. The approach systematically integrates three layers—temporal lineage modeling, unsupervised detection with noise diversity, and information-bottleneck-constrained propagation—to overcome the key limitations of traditional lineage analysis in historical tracing and change prediction tasks. The method was evaluated using logs from a real enterprise environment. Experimental results on the enterprise corpus show that, for historical root-cause localization, the average identification time is reduced from 45 to 29 min. For change-impact prediction, the method achieves $P@5 = 0.823$, $R@10 = 0.812$, and $NDCG@10 = 0.847$, achieving higher ranking metrics than the selected graph-based baselines, including PageRank, path counting, and Monte Carlo propagation. In unsupervised anomaly detection, the subgraph-level F1-score reaches 0.863 under the available reference annotations, indicating the effectiveness of the proposed noise-diversity configuration. Theoretically, the study analyzes the stability of convex-hull-distance-based anomaly decisions under noise diversity and random projection, and provides probabilistic bounds on the convergence of the propagation process. From an engineering perspective, the proposed TLG modeling and noise evaluation framework can be effectively integrated with existing data governance platforms, incurring low additional computational overhead and exhibiting good horizontal scalability.

The effectiveness of the noise-diversity detector depends on whether the structural, attribute, and temporal noise strategies can cover common benign variations in the target environment. When real-world lineage variations deviate significantly from these predefined noise families, the detector may over-filter normal subgraphs or under-detect subtle anomalies. In summary, the proposed temporal SQL lineage analysis framework achieves a well-balanced trade-off among interpretability, robustness, and engineering scalability, offering a practical and efficient technical pathway for historical tracing and change prediction in enterprise-level data governance. Future work will investigate data-driven calibration of noise-generation strategies across multiple enterprise environments, more comprehensive annotation protocols, and broader comparisons with external anomaly-detection and lineage-management systems.

Acknowledgement: None.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: The authors confirm contribution to the paper as follows: Conceptualization, Chao Liu and Hong Mu; methodology, Min Wang and Enliang Wang; software, Enliang Wang; validation, Min Wang, Enliang Wang and Xuejian Zhao; formal analysis, Xuejian Zhao; investigation, Chao Liu; resources, Hong Mu; data curation, Hong Mu and Enliang Wang; writing—original draft preparation, Min Wang and Enliang Wang; writing—review and editing, Min Wang, Enliang Wang and Xuejian Zhao; visualization, Enliang Wang; supervision, Xuejian Zhao; project administration, Chao Liu and Xuejian Zhao. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: Data not available due to commercial restrictions. Due to the nature of this research, participants of this study did not agree for their data to be shared publicly, so supporting data is not available.

Ethics Approval: Not applicable.

Conflicts of Interest: Chao Liu and Hong Mu are employees of Nanjing Digital Data Technology Co., Ltd., from which the enterprise dataset used in this study was obtained. The authors declare no conflicts of interest .

Abbreviations

The following abbreviations are used in this manuscript:

SQL	Structured Query Language
TLG	Temporal Lineage Graph
AST	Abstract Syntax Tree
DDL	Data Definition Language
IB	Information Bottleneck
GIB	Graph Information Bottleneck
NPMI	Normalized Pointwise Mutual Information
BFS	Breadth-First Search
BST	Balanced Binary Search Tree
JL	Johnson–Lindenstrauss
MD5	Message-Digest Algorithm 5
TPC-DS	Transaction Processing Performance Council Decision Support Benchmark
GCN	Graph Convolutional Network
RNN	Recurrent Neural Network
TGN	Temporal Graph Network
TGAT	Temporal Graph Attention Network
LOF	Local Outlier Factor
ROC	Receiver Operating Characteristic
PR	Precision–Recall
AUC	Area Under the Curve
F1	F1-score
P@K	Precision at K
R@K	Recall at K
NDCG@K	Normalized Discounted Cumulative Gain at K
TCS	Temporal Consistency Score
EPA	Evolution Path Accuracy
TP	True Positive
FP	False Positive
FN	False Negative
NVMe	Non-Volatile Memory Express
GB	Gigabyte
MB	Megabyte

References

1. Abraham R, Schneider J, vom Brocke J. Data governance: a conceptual framework, structured review, and research agenda. *Int J Inf Manag.* 2019;49(S1):424–38. doi:10.1016/j.ijinfomgt.2019.07.008.
2. Buneman P, Khanna S, Wang-Chiew T. Why and where: a characterization of data provenance. In: Van den Bussche J, Vianu V, editors. *Database theory—ICDT 2001. Lecture Notes in Computer Science.* Vol. 1973. Berlin/Heidelberg, Germany: Springer; 2001. p. 316–30. doi:10.1007/3-540-44503-X_20.

3. Green TJ, Karvounarakis G, Tannen V. Provenance semirings. In: Proceedings of the 26th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems; 2007 Jun 11–13; Beijing, China. New York, NY, USA: ACM; 2007. p. 31–40. doi:10.1145/1265530.1265535.
4. Jacques-Silva G, Kalyvianaki E, Cohn-Gordon K, Meguid A, Nguyen H, Ben-David D, et al. Unified lineage system: tracking data provenance at scale. In: Proceedings of the Companion of the 2025 International Conference on Management of Data; 2025 Jun 22–27; Berlin, Germany. New York, NY, USA: ACM; 2025. p. 457–70. doi:10.1145/3722212.3724458.
5. Psallidas F, Wu E. Smoke: fine-grained lineage at interactive speed. *Proc VLDB Endow*. 2018;11(6):719–32. doi:10.14778/3184470.3184475.
6. OpenLineage. Object model [Internet]. [cited 2026 Jul 3]. Available from: <https://openlineage.io/docs/spec/object-model/>.
7. Marquez Project. Marquez [Internet]. [cited 2026 Jul 3]. Available from: <https://marquezproject.ai/docs/api/marquez/>.
8. DataHub Project Authors. Lineage impact analysis [Internet]. [cited 2026 Jul 3]. Available from: <https://docs.datahub.com/docs/act-on-metadata/impact-analysis/>.
9. Pareja A, Domeniconi G, Chen J, Ma T, Suzumura T, Kanezashi H, et al. EvolveGCN: evolving graph convolutional networks for dynamic graphs. *Proc AAAI Conf Artif Intell*. 2020;34(4):5363–70. doi:10.1609/aaai.v34i04.5984.
10. Huang S, Poursafaei F, Danovitch J, Fey M, Hu W, Rossi E, et al. Temporal graph benchmark for machine learning on temporal graphs. In: Proceedings of the 37th International Conference on Neural Information Processing Systems; 2023 Dec 10–16; New Orleans, LA, USA. p. 2056–73.
11. Gastinger J, Huang S, Galkin M, Loghmani E, Parviz A, Poursafaei F, et al. TGB 2.0: a benchmark for learning on temporal knowledge graphs and heterogeneous graphs. In: Proceedings of the 38th International Conference on Neural Information Processing Systems; 2024 Dec 10–15; Vancouver, BC, Canada. p. 140199–229.
12. Zheng Y, Yi L, Wei Z. A survey of dynamic graph neural networks. *Front Comput Sci*. 2025;19(6):196323. doi:10.1007/s11704-024-3853-2.
13. Yu L, Sun L, Du B, Lv W. Towards better dynamic graph learning: new architecture and unified library. In: Proceedings of the 37th International Conference on Neural Information Processing Systems; 2023 Dec 10–16; New Orleans, LA, USA. p. 67686–700.
14. Breunig MM, Kriegel HP, Ng RT, Sander J. LOF: identifying density-based local outliers. *SIGMOD Rec*. 2000;29(2):93–104. doi:10.1145/342009.335388.
15. Ding K, Li J, Bhanushali R, Liu H. Deep anomaly detection on attributed networks. In: Proceedings of the 2019 SIAM International Conference on Data Mining; 2019 May 2–4; Calgary, AB, Canada. Philadelphia, PA, USA: SIAM; 2019. p. 594–602. doi:10.1137/1.9781611975673.67.
16. Ma X, Wu J, Xue S, Yang J, Zhou C, Sheng QZ, et al. A comprehensive survey on graph anomaly detection with deep learning. *IEEE Trans Knowl Data Eng*. 2023;35(12):12012–38. doi:10.1109/TKDE.2021.3118815.
17. Ekle OA, Eberle W. Anomaly detection in dynamic graphs: a comprehensive survey. *ACM Trans Knowl Discov Data*. 2024;18(8):192. doi:10.1145/3669906.
18. Cai L, Chen Z, Luo C, Gui J, Ni J, Li D, et al. Structural temporal graph neural networks for anomaly detection in dynamic graphs. In: Proceedings of the 30th ACM International Conference on Information and Knowledge Management; 2021 Nov 1–5; Virtual Event. New York, NY, USA: ACM; 2021. p. 3747–56. doi:10.1145/3459637.3481955.
19. Brahmia Z, Grandi F, Oliboni B. A literature review on schema evolution in databases. *Comput Open*. 2024;2(3):2430001. doi:10.1142/S2972370124300012.
20. Curino C, Moon HJ, Deutsch A, Zaniolo C. Automating the database schema evolution process. *VLDB J*. 2013;22(1):73–98. doi:10.1007/s00778-012-0302-x.
21. Hu T, Wang T, Zhou Q. Online schema evolution is (almost) free for snapshot databases. *Proc VLDB Endow*. 2022;16(2):140–53. doi:10.14778/3565816.3565818.

22. Kempe D, Kleinberg J, Tardos E. Maximizing the spread of influence through a social network. In: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2003 Aug 24–27; Washington, DC, USA. New York, NY, USA: ACM; 2003. p. 137–46. doi:10.1145/956750.956769.
23. Chen W, Wang Y, Yang S. Efficient influence maximization in social networks. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2009 Jun 28–Jul 1; Paris, France. New York, NY, USA: ACM; 2009. p. 199–208. doi:10.1145/1557019.1557047.
24. Wang C, Chen W, Wang Y. Scalable influence maximization for independent cascade model in large-scale social networks. *Data Min Knowl Discov*. 2012;25(3):545–76. doi:10.1007/s10618-012-0262-1.
25. Tishby N, Pereira FC, Bialek W. The information bottleneck method. In: Proceedings of the 37th Annual Allerton Conference on Communication, Control, and Computing; 1999 Sep 22–24; Monticello, IL, USA. Urbana, IL, USA: University of Illinois; 1999. p. 368–77.
26. Wu T, Ren H, Li P, Leskovec J. Graph information bottleneck. In: Proceedings of the 34th International Conference on Neural Information Processing Systems; 2020 Dec 6–12; Vancouver, BC, Canada. Red Hook, NY, USA: Curran Associates, Inc.; 2020. p. 20437–48.