

**FAILUREGRAPH-DB: GRAPH MACHINE LEARNING FOR PREDICTING
CASCADING DATABASE FAILURES FROM CROSS-LAYER DEPENDENCY
PATTERNS**

Nithin Reddy Gadicharla
Database Administrator, Independent Researcher
Celina, USA
nithin.gadicharla@ieee.org

Abstract

Modern database management systems operate as one component of a deeply layered software and hardware stack spanning client applications, query engines, concurrency-control systems, and physical data stores. Localized failure events in any individual layer can cascade across layer boundaries and cause a cascading failure across the entire system; these cross-layer dependencies are often not explicitly modeled in conventional per-component monitoring. In this paper, we present a conceptual framework, called FailureGraph-DB, for predicting cascading database failures from cross-layer dependency graphs, based on results from 22 studies, covering four related literatures: foundational graph representation learning, failure diagnosis in microservice architectures, graph neural network (GNN) approaches to cascading blackout prediction in power-grid infrastructure, and data-driven database self-management. It covers the evolution starting from graph convolution networks, inductive graph representation learning, graph attention mechanism, recurrent temporal encoders and personalized ranking algorithms, to fault diagnosis for microservices, nonlocal cascading blackout severity estimation, and database configuration tuning. Four recurring design patterns are identified: structural graph convolution for regular topologies, attention-weighted aggregation for varying dependency importance, temporal encoding for evolving telemetry, and ranking-based root-cause localization. A conceptual pipeline is proposed to build a cross-layer dependency graph, encode the temporal signal per node, propagate the risk across the graph with graph-based temporal signal layers, and rank the risk candidates using a personalized-ranking mechanism that utilizes nonlocal cascading-risk estimation methods borrowed from power-systems research. The synthesis also highlights some problems that are persistent, such as the scarcity of labeled incidents, the multi-layered ambiguity of causes, the non-stationarity of workloads, and the real-time constraints of inference, and outlines the paths for benchmark development and transfer learning in different scenarios of heterogeneous database deployments. It is presented not as an evaluated or deployed system, but as a literature-based conceptual synthesis intended to inform future empirical development.

Index Terms—graph neural networks, cascading failures, database systems, microservice architecture, anomaly detection, root cause analysis, graph convolutional networks, attention mechanisms, power-grid resilience, log-based diagnosis, dependency graphs, self-driving database management

I. INTRODUCTION

Today's database management systems are no longer standalone entities. These are nestled between client applications, query parsing and optimization engines, concurrency-control and transaction-management subsystems, and physical storage media. Any failure in any of these layers, including writing to a write-ahead log that is full or a lock-contention hotspot in the concurrency-control layer, can ripple both upward and downward through the stack, resulting in failure of the entire system. Traditional database monitoring approaches treat each layer — and frequently each individual metric within a layer — as a standalone signal that triggers threshold-based alerts. Such approaches are poorly suited to capturing the cross-layer, relational structure in which cascading failures do propagate [1].

Relational problems have come into the spotlight for graph machine learning, and in particular the family of graph neural network (GNN) architectures. In distributed microservice architectures, service call graphs and service execution traces have been used to locate the root cause of runtime faults [2]. In the field of electrical power-grid infrastructure, graph neural networks have been trained to predict severity and spatial extent of cascading blackouts given initial conditions in the grid, with a particular focus on the nonlocal nature of how a local disturbance can cause failures far away from its source. At the same time, the database systems community has worked on data-driven approaches to workload forecasting and configuration tuning that see the database as a learnable, adaptive system, instead of a fixed collection of manually tuned parameters.

This paper combines all three in one framework along with some basic work on graph representation learning [3] and on anomaly detection in logs and streaming telemetry to develop a conceptual model for predicting cascading database failures using cross-layer dependency patterns (FailureGraph-DB). Since none of the cited literature reviews considers a graph neural network specifically in the context of cascading database failures, the framework is presented here as a conceptual synthesis derived from the literature, and not as an empirically tested system. The rest of this paper will be structured as follows: In Section II, the range of the review is explained. In Section III, the basic architectures for graph and sequence representation learning are introduced. Graph-based diagnosis in microservice systems, cascading-failure prediction in power-grid infrastructure, log- and metric-based anomaly detection, and database self-management are explored, respectively, in Sections IV-VII. The conceptual FailureGraph-DB framework is given in Section VIII. Section IX presents a comparative analysis, Section X provides a discussion of open challenges, Section XI provides directions for future work, and Section XII concludes.

II. SCOPE OF THE REVIEW

The twenty-two sources in this review can be grouped into four general areas, which will organize the sections that follow. The first category comprises foundational graph and sequence representation-learning architectures that are not failure-prediction systems themselves, but that provide the mathematical core drawn upon by the subsequent papers [3]. The second type of research focuses on failure and anomaly diagnosis for microservice or distributed-service architectures using graph-structured or graph-adjacent learning approaches [4]. The third category comprises studies applying graph neural networks to cascading-failure prediction in power-grid infrastructure, a domain with a much longer empirical history of modeling cascading failures than

either microservices or databases. The fourth category comprises database-specific studies which, although not graph-based, demonstrate that database internals and workload dynamics can be modeled in a data-driven manner. The general anomaly detection foundations which span the four categories are addressed separately.

Table 1 summarizes this organizational scheme, which is intended to make explicit, prior to synthesis, that the review draws structural and methodological analogies across domains rather than reporting on a body of literature that already addresses database cascading failure directly [3].

Table 1: Organization of Reviewed Literature by Category and Section

Category	Representative Sources	Section
Foundational graph and sequence representation learning	[6], [7], [9], [13], [20], [21]	III
Graph-based diagnosis in microservice and distributed systems	[2], [3], [5], [11], [12], [14], [16], [17]	IV
Graph neural networks for cascading failure in power-grid infrastructure	[4], [10], [19]	V
General anomaly detection foundations	[1], [8]	VI
Database self-management and workload-aware learning	[15], [18]	VII

III. FOUNDATIONS OF GRAPH AND SEQUENCE REPRESENTATION LEARNING

A graph $G = (V, E)$ consists of a node set V and an edge set E describing pairwise relationships among nodes. In a database dependency context, nodes may represent queries, transactions, locks, buffer-pool pages, or physical storage devices, while edges represent structural or causal relationships such as invocation, lock-wait, or data-flow dependencies, as illustrated conceptually in Figure 5 (Section VIII). Graph neural networks learn a vector representation for each node by iteratively aggregating information from its neighbors, a process commonly referred to as message passing and illustrated schematically in Figure 1 [3].

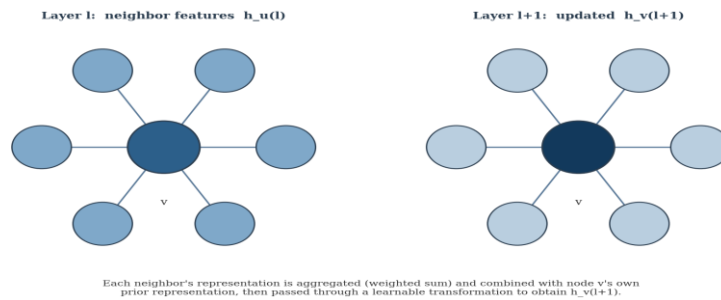


Figure 1. Graph convolutional message passing: neighbor features are aggregated and used to

update a node's representation.

The graph convolutional network (GCN) introduced by Kipf and Welling [4] formalizes this aggregation as a layer-wise propagation rule operating on the normalized graph adjacency matrix. Denoting the adjacency matrix augmented with self-loops as $\tilde{A}$, its degree matrix as $\tilde{D}$, the node-feature matrix at layer l as $H(l)$, and a learnable weight matrix as $W(l)$, the propagation rule is expressed as

$$H^{(l+1)} = \sigma(\tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2} H^{(l)} W^{(l)}) \quad (1)$$

where σ denotes a nonlinear activation function. This formulation is computationally efficient and well suited to graphs whose topology is largely fixed between training and inference, a property that holds approximately for power-grid transmission networks [4], and, potentially, for the relatively stable structural skeleton of a database's storage and concurrency-control hierarchy.

Because database and microservice topologies are not perfectly static, inductive representation learning is of particular relevance. GraphSAGE, proposed by Hamilton, Ying, and Leskovec [5], learns an aggregation function rather than a fixed set of per-node embeddings, allowing the trained model to generate representations for nodes unseen during training. For a node v with neighborhood $N(v)$, the update at layer l is expressed as

$$h_v^{(l+1)} = \sigma(W \cdot \text{CONCAT}(h_v^{(l)}, \text{AGG}\{h_u^{(l)} : u \in N(v)\})) \quad (2)$$

This inductive property is directly applicable to a framework intended to cover a variety of database deployments, each with a schema, index structure, and connection graph that may differ from one deployment to another, much as Guo et al. [5] construct trace-graph representations able to accommodate changing microservice call topologies.

The influence of neighboring nodes on a node's evolving state is not uniform. Veličković et al. [6] propose the graph attention network (GAT), which learns a normalized attention coefficient for each edge, allowing the model to weight the contribution of each neighbor according to the learned relevance of that edge. Conceptually, this is shown in Figure 2. The attention coefficient between node v and the neighbor j is calculated as

$$\alpha_{vj} = \text{softmax}_j(\text{LeakyReLU}(a^T [W h_v \parallel W h_j])) \quad (3)$$

and the updated representation of node v is obtained as a weighted combination of neighbor features [6],

$$h_v' = \sigma(\sum_{j \in N(v)} \alpha_{vj} W h_j) \quad (4)$$

where a and W are parameters to be learned. This mechanism is also directly useful for diagnosis: an attention coefficient that is large on one dependency and small on others identifies the dependency most likely responsible for an anomaly observed downstream, as exploited implicitly by causal root-cause analysis methods in microservice diagnosis [7] and by the multi-feature

fusion design of MTG_CD.

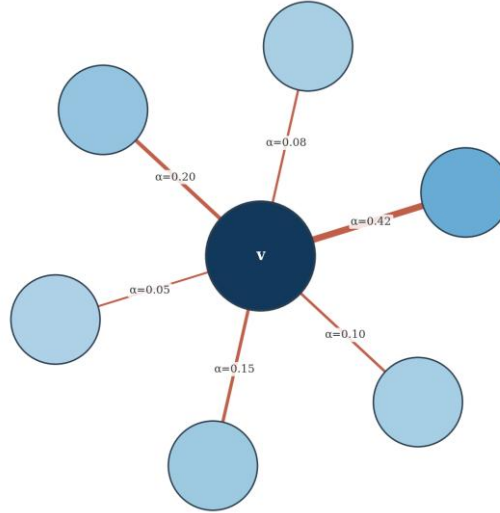


Figure 2. Graph attention mechanism: edge thickness denotes the learned attention coefficient assigned to each neighbor.

Telemetry on databases and microservices is inherently temporal: buffer-pool hit rates, queue depths, latency percentiles, and other values constantly change over time, and need to be encoded prior to being merged with the structural graph information. The long short-term memory (LSTM) architecture of Hochreiter and Schmidhuber [7] mitigates the vanishing-gradient problem of earlier recurrent networks through a gated cell state that regulates the flow of information over long sequences, governed by a forget gate, an input gate, and an output gate together with a candidate cell-state update. This form of temporal encoding is used in log-based anomaly detection and also forms part of the multi-scale microservice diagnosis built on a graph representation of the system.

Where dependencies span an entire sequence rather than only its local neighborhood, the scaled dot-product attention mechanism of the Transformer architecture of Vaswani et al. [8] provides an alternative. For query, key, and value matrices Q , K , and V , where d_k denotes the dimensionality of the key vectors, attention is calculated as

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (5)$$

Finally, once a dependency graph has been constructed and node-level anomaly scores derived, a mechanism is required to rank candidate root causes. This is based on the idea of personalized PageRank [8] by Jeh and Widom, which derives an importance vector r by the following recurrence:

$$r = (1 - c)e + cP^T r \quad (6)$$

where e is a personalization (seed) vector, P is the transition matrix of the graph, and c is a damping factor. Seeding this random walk at nodes that have anomalous telemetry instead of a

generic root node returns a ranked list of potential root causes weighted by the combination of telemetry evidence and graph proximity, which is similar to the hybrid graph-and-log diagnosis method of Jia et al. [9] and the time-weighted control-flow graph mining of LogSed.

Table 2: Foundational Representation-Learning Architectures and Their Core Mechanisms

Architecture	Source	Core Mechanism	Primary Relevance to Cascading-Failure Modeling
Graph Convolutional Network (GCN)	[13]	Normalized spectral aggregation over fixed adjacency	Efficient propagation over relatively stable topology
GraphSAGE	[6]	Learned, inductive neighbor-sampling and aggregation	Generalization to unseen or evolving dependency graphs
Graph Attention Network (GAT)	[21]	Learned per-edge attention coefficients	Interpretable weighting of upstream dependency importance
Long Short-Term Memory (LSTM)	[7]	Gated recurrent cell state over sequences	Temporal encoding of streaming performance counters
Transformer / scaled dot-product attention	[20]	Global pairwise attention over a sequence	Long-range temporal dependency modeling
Personalized PageRank	[9]	Seeded random-walk recurrence	Root-cause ranking anchored at anomalous nodes

IV. GRAPH-BASED FAILURE DIAGNOSIS IN MICROSERVICE SYSTEMS

The most extensive development of graph-based failure diagnosis has been in microservice architectures, where the service call graph is structurally analogous to the cross-layer dependency graph proposed for database systems in Section VIII. Jia et al. [9] propose a hybrid graph model that combines a structural service-dependency graph with log text content to detect anomalies in distributed services, while another work from the same authors, LogSed, extracts a time-weighted control-flow graph directly from log sequences, where each transition is weighted according to its temporal proximity to an observed anomaly. Guo et al. extend this work by constructing trace graphs directly from distributed tracing data, which they show carries significant diagnostic value for architecture understanding and problem localization without relying on log text.

Later research has introduced explicit graph neural networks along with temporal feature extraction. To extract temporal-structural features from system monitoring metrics, Chen et al. [10] propose MTG_CD that combines multi-scale neural transformation and learnable graph-structure adjacency, and then fuses these features using a graph convolutional network for fault classification and diagnosis. This design is directly inspired by the temporal-then-structural fusion pattern described in Section III, where an LSTM-style temporal encoder first generates node features which are then propagated through a GCN. Unlike the purely structural graph approaches, Lin et al. adopt a causal-inference perspective, using neural Granger causal discovery to infer the direction of causal relationships between microservices rather than relying on

topological proximity or correlation alone. Song et al. consider a complementary practical question: no single fault-classification model is optimal under all operating conditions, and they propose an autonomous model-selection algorithm that dynamically chooses a fault-classification model according to the system's current operating state.

This body of work has been summarized in a more comprehensive survey of anomaly detection and failure root cause analysis for microservices-based cloud applications by Soldani and Brogi [10] from the perspective of data modality consumed (metrics, logs or traces) and of the diagnostic task (detection, localization or root cause explanation). As summarized in Table 3, this literature converges on a common three-part structure: a graph encoding dependency information, a recurrent or multi-scale transformation module for temporal telemetry, and a propagation module – graph convolution, attention, or causal-graph discovery – that yields either a fault-type classification or a ranking of candidate root causes.

Table 3: Graph-Based Failure and Anomaly Diagnosis Studies in Microservice Systems

Study	Graph / Data Representation	Learning Technique	Diagnostic Task
Jia et al. [11]	Hybrid service-dependency graph with log text	Graph model with heuristic scoring	Anomaly diagnosis in distributed services
Jia et al., LogSed [12]	Time-weighted control-flow graph mined from logs	Weighted graph traversal	Anomaly diagnosis via log mining
Guo et al. [5]	Trace graph from distributed tracing	Structural graph analysis	Architecture understanding and problem diagnosis
Chen et al., MTG_CD [2]	Learnable structural adjacency over services	Multi-scale transformation + GCN	Fault classification and diagnosis
Lin et al. [14]	Inferred causal graph among services	Neural Granger causal discovery	Root-cause analysis
Song et al. [17]	Model-selection over multiple classifiers	Autonomous adaptive model selection	Fault classification under varying conditions
Soldani & Brogi [16]	Survey (metrics, logs, traces)	Taxonomy synthesis	Anomaly detection and root-cause analysis, broadly

V. CASCADING FAILURE PREDICTION IN POWER-GRID INFRASTRUCTURE

Compared with software systems, the empirical track record in cascading failure prediction is quite long in power grids, due to the significant consequences of widespread blackouts and the need to quickly check contingency situations in the grid. However, as Gorka et al. [11] note, traditional power-flow-based simulation tools that evaluate cascading blackout risk are limited by their computational speed, and with the increasing variability of renewable energy sources, a vast number of possible initiating conditions emerges. They propose a statistically augmented graph neural network architecture comprising a coarse classification stage that eliminates scenarios deemed unlikely to cascade, a subsequent magnitude-estimation stage for scenarios passing that

classifier, and a nonlocal message-passing mechanism motivated by the observation that failures in transmission networks frequently propagate to distant rather than adjacent nodes.

This nonlocal propagation behavior is studied directly by Jhun et al. [12] who propose an avalanche centrality measure, which is based on the classical model of cascading failure known as the Motter-Lai overload-redistribution model. Rather than exhaustively simulating all possible initiating disturbances, which is computationally costly, the authors train a graph neural network to learn avalanche centrality values on small synthetic networks and evaluate it on substantially larger synthetic networks and real grid topologies. This transferability result is of methodological importance for the framework proposed in this review, as it demonstrates the feasibility of training a GNN on the structural cascading-risk patterns of one topology and generalizing to structurally different yet relationally similar topologies, which is equivalent to training a model on the dependency graph of one deployment and applying it to another deployment with a different structure.

A longstanding challenge in this area, and one that is particularly relevant to the database environment, is the lack of available benchmark data that are standard and available to share. Real topologies and failure rates are often sensitive to proprietary concerns and/or security issues. To overcome this deficiency, Varbella et al. [12] propose the PowerGraph benchmark dataset, which consists of graph representations of power grid topology, along with simulated cascading-failure scenarios. As Table 4 and the comparative discussion in Section IX indicate, the availability of such a benchmark has enabled comparable and reproducible evaluation of competing GNN architectures — a capability currently absent from cascading database failure research.

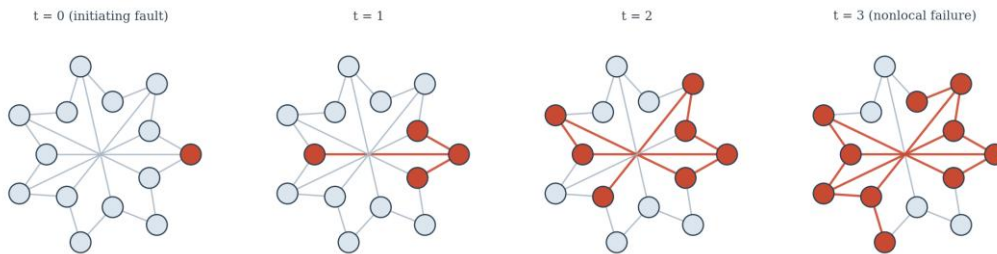


Figure 3. Conceptual illustration of nonlocal cascading failure propagation across successive time steps following a single initiating fault.

Table 4: Graph Neural Network Studies for Cascading Failure Prediction in Power-Grid Infrastructure

Study	Graph Domain	GNN Design Element	Key Contribution
Gorka et al. [4]	Transmission network topology	Two-stage classify-then-regress GNN with nonlocal message passing	Fast blackout-magnitude estimation
Jhun et al. [10]	Synthetic and real electrical grids	GNN trained on avalanche centrality	Transferable cascading-risk ranking across network sizes

Study	Graph Domain	GNN Design Element	Key Contribution
Varbella et al. [19]	Benchmark grid topologies	Standardized graph dataset	Reproducible evaluation resource (PowerGraph)

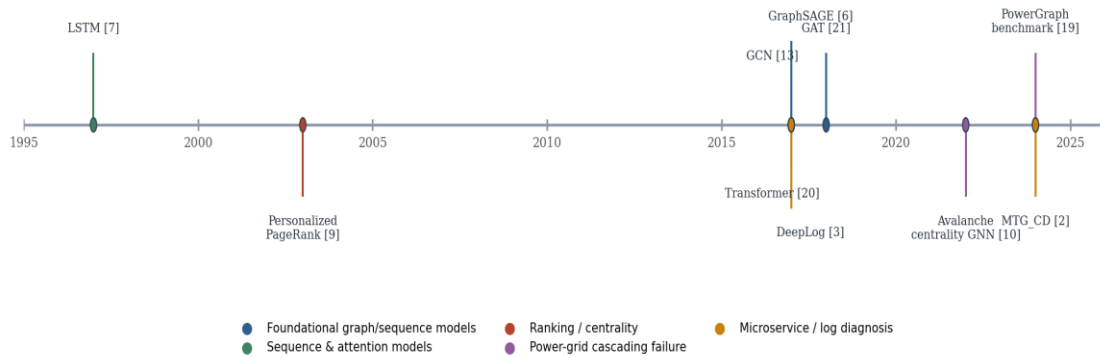


Figure 4. Approximate timeline of graph representation-learning and cascading-failure milestones synthesized in this review.

VI. FOUNDATIONS OF LOG- AND METRIC-BASED ANOMALY DETECTION

Both the microservice and power-grid literatures build upon a general foundation of anomaly detection research. Chandola et al. [13] have proposed a widely cited taxonomy which categorizes anomalies into three types: point anomalies, contextual anomalies, and collective anomalies and classifies detection methods into families including classification-based, clustering-based, nearest-neighbor-based, statistical, spectral, and information-theoretic approaches. This taxonomy provides the conceptual vocabulary that is used in Sections IV through VII to describe, in each individual study, what the anomalies are and how they are represented, and that is referenced again in Section X in the context of the class-imbalance and false-positive problems of cascading-failure prediction.

When the task of anomaly detection needs to be carried out over continuously incoming telemetry, instead of pre-collected data, scalability is always a concern. Huang and Kasiviswanathan address this concern directly with a streaming anomaly detection method that maintains a compact, continuously updated summary of high-dimensional streaming data using only a small randomized sub-matrix, substantially reducing memory requirements [13]. This design principle directly applies to database telemetry, which is typically generated continuously and in large volumes from many components being monitored concurrently, and is contrasted with the usual batch-oriented training procedures used with graph neural networks.

Du et al. [14] present a model named DeepLog which represents a sequence of log keys produced by a running system as a natural language sentence, and trains an LSTM type model (as described in Section III) for predicting the next log key in a sequence based on the recent history. The

discrepancies between predicted and actual log sequences are then considered anomalies and a complementary workflow-construction procedure is used to reconstruct the normal execution paths suggested by the log information. DeepLog is notable for showing that meaningful anomalies can be detected by modeling system behavior as a sequence of events over a discrete vocabulary, without recourse to the graph-explicit approaches described in Section IV, thereby anticipating the temporal-encoding stage of the FailureGraph-DB framework proposed in Section VIII.

Table 5: Foundational Anomaly Detection Studies for Logs and Streaming Telemetry

Study	Data Modality	Technique	Relevance to Database Telemetry
Chandola et al. [1]	General / cross-domain	Taxonomy of anomaly types and detection families	Conceptual vocabulary for anomaly characterization
Huang & Kasiviswanathan [8]	High-dimensional data streams	Randomized matrix sketching	Memory-efficient streaming anomaly scoring
Du et al., DeepLog [3]	System console logs	LSTM next-key prediction	Sequence-based precursor to graph-temporal fusion

VII. DATABASE SELF-MANAGEMENT AND WORKLOAD-AWARE LEARNING

Neither of the two database-specific studies reviewed here uses graph neural networks, but both demonstrate that database internals and workload dynamics are amenable to data-driven learned modeling — a prerequisite for extending graph-based failure prediction into the database domain. In order to alleviate the manual expertise traditionally needed to tune DBMS configuration, Van Aken et al. [14] proposed a system, OtterTune, that automatically tunes the configuration of a DBMS by leveraging machine learning models trained from a large number of DBMS tuning sessions observed from a wide variety of previous workloads. The underlying principle — that a large corpus of observed system states and outcomes can substitute for hand-crafted tuning heuristics — applies equally to the problem of learning which cross-layer dependency patterns historically precede cascading failure.

Ma et al. [15] take this data-driven approach one step further by suggesting a query-based approach for predicting the future workload arrival pattern in self-driving database management systems. Rather than predicting aggregate resource utilization directly, the method forecasts arrivals at the level of individual query templates, allowing the resulting forecasts to capture structural shifts in the mix of operations a system will encounter as well as changes in load intensity. This workload-forecasting capability is a natural complement to the FailureGraph-DB framework proposed in Section VIII, since it offers a means of distinguishing expected behavioral change driven by workload shifts from unexpected structural faults — a distinction difficult to draw from telemetry alone.

Table 6: Database Self-Management and Workload-Aware Learning Studies

Study	Focus	Technique	Complementary Role in FailureGraph-DB
Van Aken et al., OtterTune [18]	DBMS configuration tuning	ML models trained on historical tuning sessions	Establishes feasibility of learned DB internals modeling
Ma et al. [15]	Workload forecasting	Query-template-based forecasting	Contextual prior to discount anticipated load shifts

VIII. FAILUREGRAPH-DB: A CONCEPTUAL SYNTHESIS FRAMEWORK

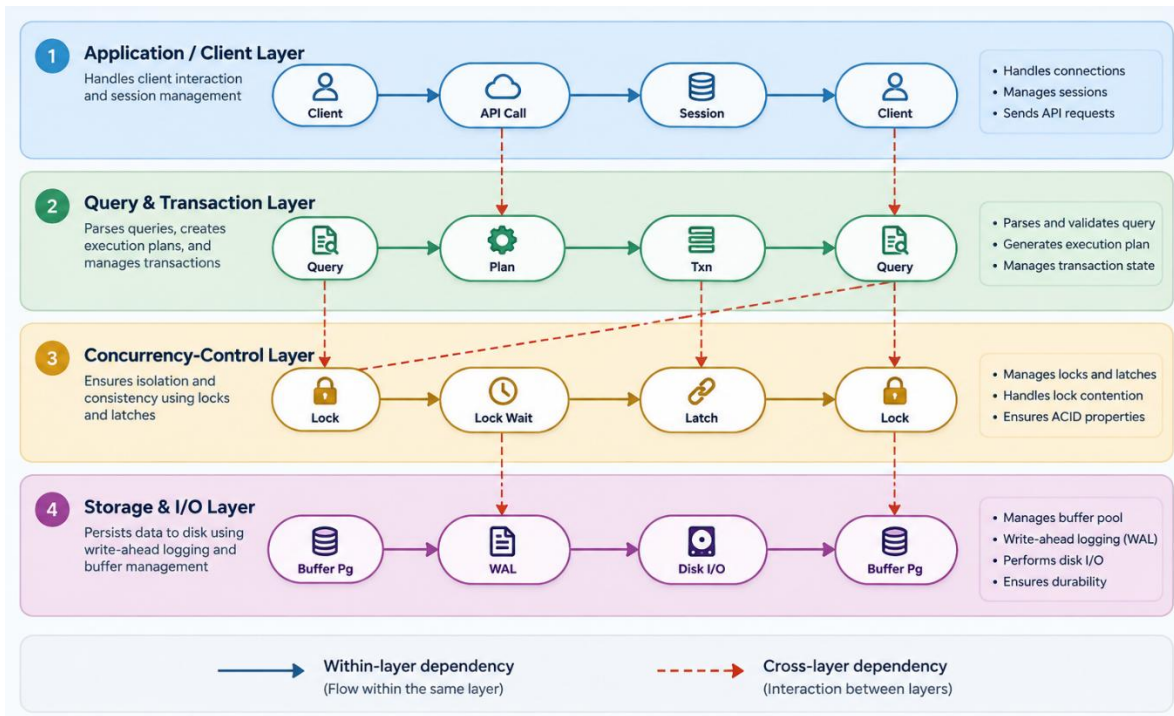


Figure 5. Conceptual cross-layer database dependency graph spanning the application, query, concurrency-control, and storage layers.

This section introduces FailureGraph-DB, a conceptual framework synthesizing the four literatures reviewed above. It is explicitly not an implemented or empirically validated system. The framework is organized as a six-stage conceptual pipeline – cross-layer telemetry collection, dependency graph construction, temporal encoding, graph propagation, risk ranking, and cascade-risk scoring – with a feedback loop that adjusts edge and node weights in the graph according to observed mitigation outcomes [15].

The dependency graph itself is built on the four architectural layers shown in Figure 5: an application and client layer, a query and transaction layer, a concurrency-control layer and a storage and input-output layer. Each node corresponds to a concrete system entity – an active

query, a transaction, a lock, or a storage device — while each edge corresponds to a structural relationship such as a call sequence or a lock-wait chain. The cross-layer edges (dashed lines in Figure 5) represent statistical or historical correlations between anomalous states in different layers, and it is these co-occurrence relationships that the nonlocal message-passing mechanism — adapted from cascading blackout estimation — is intended to exploit [16].

To follow the temporal-encoding principle used in log-sequence anomaly detection [17] and multi-scale microservice diagnosis, each node is first given a temporal feature vector generated by LSTM encoder of its recent stream of performance counters. These temporal node features are then passed across the dependency graph using stacked graph convolutional and graph attention layers, and the attention coefficient of Equation 3 offers an intuitive clue as to which upstream dependency is most influential on the evolving risk state of a downstream node, directly supporting the causal-attribution goal of the microservice literature on Granger causal discovery.

Nodes whose propagated representations deviate significantly from historical norms — characterized using the anomaly taxonomy of Chandola et al. [18] and identified at streaming scale via sketching techniques of the kind proposed by Huang and Kasiviswanathan — serve as seeds for a personalized-ranking step. In the spirit of the hybrid graph-and-log ranking strategy of Jia et al., the ranked list of candidate root causes is obtained by a random walk that starts from these anomalous nodes, following the personalized PageRank formulation in Equation 6.

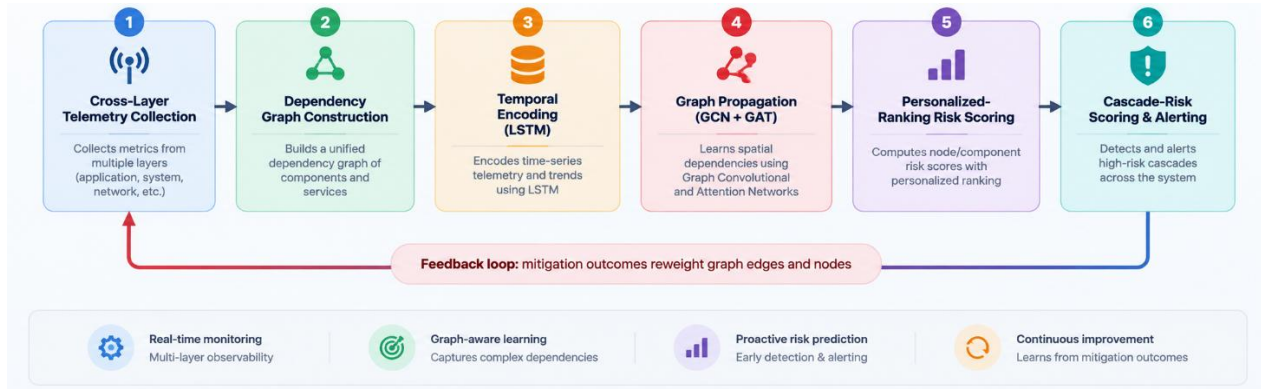


Figure 6. Conceptual FailureGraph-DB processing pipeline, from cross-layer telemetry collection to cascade risk scoring.

$$R_v = \lambda_1 \cdot \text{GNNrisk}_v + \lambda_2 \cdot \text{PPR}_v + \lambda_3 \cdot \text{AC}_v \quad (7)$$

where λ_1 , λ_2 , and λ_3 are weighting coefficients to be determined empirically once suitably labeled cascading-incident data are available. The composite score is intended to surface not only components exhibiting anomalous behavior, but also structurally critical components whose failure would be substantially more likely to cascade — the same motivation underlying centrality measures in the power-grid literature [19]. The framework also includes workload forecasting information and configuration-tuning history as contextual priors in the risk-scoring stage, which helps to account for expected, workload-induced variations in the score before considering it to be structurally risky.

IX. COMPARATIVE ANALYSIS OF METHOD FAMILIES

Table 7 and the heatmap in Figure 7 summarize qualitatively the relative emphasis placed on five method families – graph convolution, graph attention, recurrent/temporal modeling, ranking/centrality-based, and classical statistical/rule-based – across the four domains synthesized in this review [19]. In the field of microservice diagnosis and power-grid cascading failure, relatively strong attention is given to graph convolutional and attention-based methods, due to the existing well-defined call-graph or transmission-network topologies. In contrast, research on database systems is still heavily founded on classical statistical and recurrent techniques, while graph-explicit techniques are only marginally explored, and in particular the attention-based techniques—and their variants—remain under-researched, a motivation for the conceptual synthesis of FailureGraph-DB that is argued in this review.

A second axis of comparison is computational scalability – specifically, whether a method can operate within the latency and memory budgets of an online monitoring system. While the streaming, sketch-based approach used by Huang and Kasiviswanathan [20] is tailored towards memory and latency restrictions, the graph neural architectures surveyed in Sections IV and V are generally trained at batch scale and tested at the same scale. Gorka et al. motivate their work on the grounds that traditional power-flow simulation is too slow for real-time scenario screening, and the same latency pressure applies to database systems: a cascading-failure warning issued only after degradation is user-visible has substantially less operational value.

A third axis of comparison is benchmark availability. PowerGraph [21] has provided the power-systems GNN community with a common, reproducible evaluation resource, which is a common prerequisite for methodological advances in various application fields as pointed out by the general graph neural network survey of Wu et al. In the literature reviewed no direct comparison with a benchmark for cross-layer database cascading failure could be found, which has been identified as a priority for future empirical work in Section X.

A fourth comparison is with regard to interpretability. In graph attention networks [21] and in MTG_CD there is a mechanism for distributing attention across edges, which provides a partial, edge-level explanation of the predictions made by these models; and the causal-discovery approach of Lin et al. aims for a stronger, directionally causal explanation. As Soldani and Brogi observe, interpretability is a less consistently supported requirement than raw detection accuracy in the microservice anomaly-detection literature; the same consideration applies in the database context, where trust in an automated cascading-failure warning will depend on the system's ability to identify which upstream dependency is implicated.

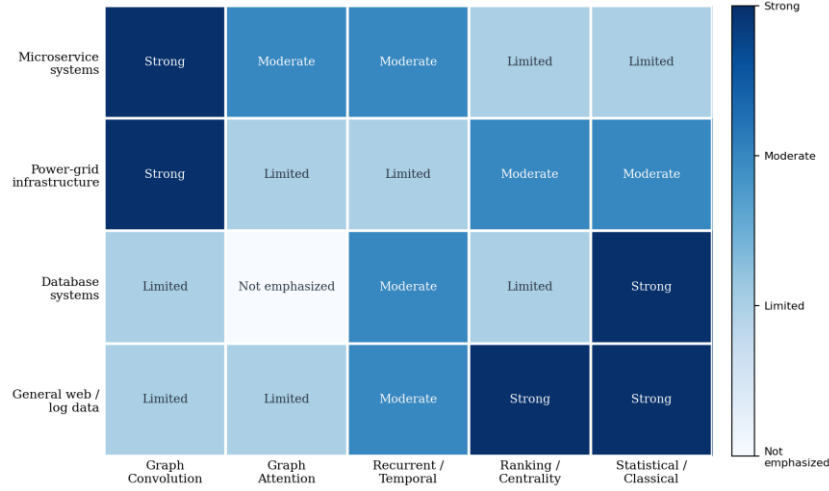


Figure 7. Qualitative heatmap of relative method-family emphasis across the four synthesized domains.

Table 7: Qualitative Comparative Emphasis of Method Families Across Domains

Domain	Graph Convolution	Graph Attention	Recurrent / Temporal	Ranking / Centrality	Statistical / Classical
Microservice systems	Strong	Moderate	Moderate	Limited	Limited
Power-grid infrastructure	Strong	Limited	Limited	Moderate	Moderate
Database systems	Limited	Not emphasized	Moderate	Limited	Strong
General web / log data	Limited	Limited	Moderate	Strong	Strong

X. CHALLENGES AND OPEN ISSUES

There are several recurring problems within the synthesized literature and that are likely to be applicable with similar or higher impact to cascading database failure prediction. First, labeled cascading-failure events are extremely sparse relative to the volume of normal-operation telemetry, creating a severe class-imbalance problem of the kind described in general terms by Chandola et al. [22] and only partially addressed by the two-stage classification-then-regression approach of Gorka et al. Second, causal ambiguity across layers is difficult to resolve: anomalies correlated across multiple layers do not necessarily imply a causal relationship. Although Granger causal discovery and time-weighted control-flow graph mining represent meaningful advances for microservice systems, no directly analogous method has yet been demonstrated across the heterogeneous layer types – application, query, concurrency control, and storage – that a database cascading-failure framework must span.

Third, workload and configuration non-stationarity makes it difficult to establish a stable baseline

of normal behavior, and consequently difficult to identify genuine deviations from it. Planned interventions such as those informed by workload forecasting [22] and automated configuration tuning produce legitimate changes in system behavior that a cascading-failure model must distinguish from unplanned structural degradation – a capability the composite risk score proposed in Section VIII is intended to provide but has not been empirically shown to deliver. Fourth, real-time inference constraints require that any graph-based model be evaluable at a latency compatible with operational response times. Streaming sketching of the kind demonstrated by Huang and Kasiviswanathan for anomaly scoring is feasible, but has not been shown to integrate with the deeper multi-layer graph propagation described in Section VIII. Finally, existing attention-based mechanisms provide only partial interpretability and do not by themselves establish operator trust, a limitation noted both in the anomaly-detection and root-cause-analysis survey of Soldani and Brogi and in the graph neural network survey of Wu et al.

XI. FUTURE DIRECTIONS

Four directions for future work follow from the challenges identified above. First, the power-systems community's experience with PowerGraph [22] suggests that a comparable effort to establish an open, standardized benchmark of instrumented database dependency graphs and labeled cascading-incident records would substantially accelerate progress in the database domain. Second, hybrid architectures built from combinations of the long-range sequence modeling power of Transformer-like attention and the explicit structural bias of graph attention networks should be explored for their ability to capture both slow and abrupt sequence-level drift in the same model. Third, the inductive generalization demonstrated by GraphSAGE and, in the power-grid setting, by the avalanche-centrality transfer results of Jhun et al. suggests that a model trained on the topology of one deployment may generalize to deployments with different topologies – a property of considerable practical value given the heterogeneity of real-world database installations. Finally, coupling workload forecasting more tightly with the graph-propagation stage described in Section VIII could allow a cascading-risk model to anticipate a high-risk window ahead of a forecast workload peak, rather than reacting only once early risk indicators have already appeared.

XII. CONCLUSION

In this paper, we have synthesized 22 papers covering foundational graph representation learning, failure diagnosis of microservices, cascading failure prediction in power-grid infrastructure using graph neural networks, and data-driven self-management of databases to build a conceptual framework called FailureGraph-DB for predicting cascading failures in databases from cross-layer dependency patterns. The review identified four cross-cutting motifs from those adjacent literatures: structural graph convolution, attention-weighted aggregation, temporal sequence encoding and personalized-ranking-based root-cause localization, and proposed a conceptual six-stage pipeline based on the identified motifs, thereby embedding them into a cross-layer database dependency graph. Persistent open challenges have also been identified, with corresponding directions for future work in benchmark development and transfer-learning. It is a framework that we propose as a literature-based starting point for future empirical research, not as a fully-fledged system that has been deployed in the wild; it is important to note that the main strength of the

proposed framework is that it draws on the structural analogies that have not yet been brought together in a single framework, in the context of cascading-failure phenomena in power-grid infrastructure, distributed microservice systems, and database internals.

ACKNOWLEDGMENT

This review was prepared as an independent literature synthesis. No external sponsorship is associated with this manuscript.

REFERENCES

1. Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys*, 41(3), Article 15, 1-58. <https://doi.org/10.1145/1541880.1541882>
2. Chen, J., Zhang, R., Chen, P., Ren, J., Wu, Z., Wang, Y., Li, X., & Xiong, L. (2024). MTG_CD: Multi-scale learnable transformation graph for fault classification and diagnosis in microservices. *Journal of Cloud Computing*, 13(1), Article 103. <https://doi.org/10.1186/s13677-024-00666-0>
3. Du, M., Li, F., Zheng, G., & Srikanth, V. (2017). DeepLog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (pp. 1285-1298). Association for Computing Machinery. <https://doi.org/10.1145/3133956.3134015>
4. Gorka, J., Hsu, T., Li, W., Maximov, Y., & Roald, L. (2024). Cascading blackout severity prediction with statistically-augmented graph neural networks (arXiv:2403.15363). *arXiv*. <https://doi.org/10.48550/arXiv.2403.15363>
5. Guo, X., Peng, X., Wang, H., Li, W., Jiang, H., Ding, D., Xie, T., & Su, L. (2020). Graph-based trace analysis for microservice architecture understanding and problem diagnosis. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (pp. 1387-1397). Association for Computing Machinery. <https://doi.org/10.1145/3368089.3417066>
6. Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive representation learning on large graphs (arXiv:1706.02216). *arXiv*. <https://doi.org/10.48550/arXiv.1706.02216>
7. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735-1780. <https://doi.org/10.1162/neco.1997.9.8.1735>
8. Huang, H., & Kasiviswanathan, S. P. (2015). Streaming anomaly detection using randomized matrix sketching. *Proceedings of the VLDB Endowment*, 9(3), 192-203. <https://doi.org/10.14778/2850583.2850593>
9. Jeh, G., & Widom, J. (2003). Scaling personalized web search. In *Proceedings of the 12th International Conference on World Wide Web* (pp. 271-279). Association for Computing Machinery. <https://doi.org/10.1145/775152.775191>
10. Jhun, B., Choi, H., Lee, Y., Lee, J., Kim, C. H., & Kahng, B. (2022). Prediction and mitigation of nonlocal cascading failures using graph neural networks (arXiv:2208.00133). *arXiv*. <https://doi.org/10.48550/arXiv.2208.00133>
11. Jia, T., Chen, P., Yang, L., Li, Y., Meng, F., & Xu, J. (2017). An approach for anomaly diagnosis based on hybrid graph model with logs for distributed services. In *2017 IEEE*

- International Conference on Web Services (pp. 25–32). IEEE.
<https://doi.org/10.1109/ICWS.2017.12>
12. Jia, T., Yang, L., Chen, P., Li, Y., Meng, F., & Xu, J. (2017). LogSed: Anomaly diagnosis through mining time-weighted control flow graph in logs. In 2017 IEEE 10th International Conference on Cloud Computing (pp. 447–455). IEEE.
<https://doi.org/10.1109/CLOUD.2017.64>
 13. Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks (arXiv:1609.02907). arXiv. <https://doi.org/10.48550/arXiv.1609.02907>
 14. Lin, C.-M., Chang, C., Wang, W.-Y., Wang, K.-D., & Peng, W.-C. (2024). Root cause analysis in microservice using neural Granger causal discovery. Proceedings of the AAAI Conference on Artificial Intelligence, 38(1), 206–213.
<https://doi.org/10.48550/arXiv.2402.01140>
 15. Ma, L., Van Aken, D., Hefny, A., Mezerhane, G., Pavlo, A., & Gordon, G. J. (2018). Query-based workload forecasting for self-driving database management systems. In Proceedings of the 2018 International Conference on Management of Data (pp. 631–645). Association for Computing Machinery. <https://doi.org/10.1145/3183713.3196908>
 16. Soldani, J., & Brogi, A. (2022). Anomaly detection and failure root cause analysis in (micro)service-based cloud applications: A survey. ACM Computing Surveys, 55(3), Article 59, 1–39. <https://doi.org/10.1145/3501297>
 17. Song, Y., Xin, R., Chen, P., Zhang, R., Chen, J., & Zhao, Z. (2024). Autonomous selection of the fault classification models for diagnosing microservice applications. Future Generation Computer Systems, 153, 326–339. <https://doi.org/10.1016/j.future.2023.12.005>
 18. Van Aken, D., Pavlo, A., Gordon, G. J., & Zhang, B. (2017). Automatic database management system tuning through large-scale machine learning. In Proceedings of the 2017 ACM International Conference on Management of Data (pp. 1009–1024). Association for Computing Machinery. <https://doi.org/10.1145/3035918.3064029>
 19. Varbella, A., Amara, K., Gjorgiev, B., El-Assady, M., & Sansavini, G. (2024). PowerGraph: A power grid benchmark dataset for graph neural networks (arXiv:2402.02827). arXiv. <https://doi.org/10.48550/arXiv.2402.02827>
 20. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need (arXiv:1706.03762). arXiv. <https://doi.org/10.48550/arXiv.1706.03762>
 21. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph attention networks (arXiv:1710.10903). arXiv. <https://doi.org/10.48550/arXiv.1710.10903>
 22. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2021). A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems, 32(1), 4–24. <https://doi.org/10.1109/TNNLS.2020.2978386>