Global Academic Journal of Economics and Business

Available online at https://www.gajrc.com
DOI: https://doi.org/10.36348/gajeb.2026.v08i03.018

OPEN ACCESS

ISSN: 2706-9001 (P)
ISSN: 2707-2584 (O)

Original Research Article

Machine Learning-Based Anomaly Detection for Hybrid Cloud Infrastructure and Mission-Critical Database Workloads

Tahseen Zafar[1]*
[1]Independent Researcher

***Corresponding Author**
Tahseen Zafar
Independent Researcher

Article History
Received: 13.04.2026
Accepted: 04.06.2026
Published: 06.06.2026

Abstract: Hybrid cloud has become the default operating model for enterprises that must combine public cloud elasticity, private cloud control, and locally governed data platforms. In this setting, mission-critical database workloads are often the most fragile layer because small deviations in wait events, query plans, replication delay, storage latency, or authentication behaviour can propagate into customer-facing outages and security exposure. This review synthesises research and industry evidence published between 2020 and 2025 on machine learning-based anomaly detection for hybrid cloud infrastructure and critical database operations. The aim is to define a practical evidence-informed framework that links telemetry collection, feature engineering, model selection, explainability, and guarded response. The review shows that no single model family is sufficient. Statistical baselines and rules remain valuable for stable service-level indicators, while tree-based, probabilistic, deep sequence, transformer, graph, and knowledge-based methods address non-linear workloads, cross-service dependencies, and root-cause reasoning. For database workloads, anomaly detection must integrate infrastructure metrics with database-native signals, including active sessions, wait classes, lock contention, query execution plans, buffer usage, redo or write-ahead logging behaviour, and replication health. The paper proposes an architecture in which observability, topology context, model governance, and operational runbooks are treated as one system rather than separate tools. It therefore argues that successful deployment should be measured not only by precision and recall, but also by mean time to detect, mean time to recover, false-alert cost, safety of automated actions, and compliance readiness. The review concludes that hybrid cloud anomaly detection is most reliable when machine learning is constrained by domain knowledge, continuous validation, human approval for high-risk interventions, and clear accountability across platform, security, and database engineering teams.

Keywords: Anomaly Detection, Hybrid Cloud, Database Workloads, Aiops, Observability, Machine Learning, Root-Cause Analysis, Operational Resilience.

1. INTRODUCTION

Hybrid cloud operations no longer represent an exceptional deployment pattern. Large organisations frequently retain regulated databases in private environments while distributing applications, analytics services, and user-facing platforms across several public cloud regions. This creates an operational surface in which failures

rarely remain local. A storage throttle in a private data centre can increase database wait time, which then appears as application latency in a cloud service, which then triggers retries that amplify load across gateways, queues, and connection pools. Research on cloud-native systems has therefore shifted from isolated server monitoring towards cross-service anomaly detection and failure root-cause analysis, especially in microservice-based applications where symptoms can travel through many dependent services before the true cause is visible (Soldani and Brogi, 2022; Pedroso *et al*., 2025).

Mission-critical databases make this problem sharper. Unlike stateless containers, databases hold durable business state, enforce transactional consistency, and often support strict recovery point and recovery time objectives. Performance anomalies in databases are not limited to CPU or memory signals. They include plan regressions, lock storms, index misuse, replication lag, checkpoint pressure, temporary-space exhaustion, deadlocks, privilege misuse, connection saturation, cache churn, and abnormal query mixes. Cloud database platforms increasingly expose built-in monitoring and automated tuning, reflecting the practical need to detect workload instability and performance regression without waiting for manual database administrator review (Amazon Web Services, 2024; Microsoft, 2025). Yet hybrid cloud environments still require independent detection strategies because many enterprises operate multiple database engines, several observability tools, and both managed and self-managed infrastructure.

Machine learning offers a compelling response because it can learn temporal, multivariate, and contextual behaviour from large telemetry streams. Studies on cloud components and large-scale cloud platforms indicate that recurrent networks, autoencoders, and other data-driven methods can identify early deviations across complex service stacks (Islam and Miranskyy, 2020; Islam *et al*., 2021). Recent work has also explored knowledge graphs, graph neural networks, and language-model-assisted reasoning to improve interpretability and root-cause explanation in cloud-native settings (Mitropoulou *et al*., 2024; Hsieh *et al*., 2024; Pedroso *et al.,* 2025). However, deployment in production is difficult. Training data can be imbalanced, anomalies are rare, labels are inconsistent, logs evolve after every release, and false positives can erode operator trust.

This review focuses on the intersection of hybrid cloud infrastructure and mission-critical database workloads. It treats anomaly detection as an operational capability, not merely as an algorithmic classification task. A useful detector must recognise abnormal behaviour, explain why it matters, route the signal to the right owner, and support safe remediation. The central argument is that high-quality anomaly detection depends on the joint design of telemetry, feature context, model governance, and incident workflow. Without this integration, sophisticated models may increase alert volume rather than improving resilience.

2. Aim and Objectives

The aim of this review is to develop a consolidated, journal-ready perspective on how machine learning-based anomaly detection can be designed, evaluated, and governed for hybrid cloud infrastructure that supports mission-critical database workloads.

The objectives are fivefold. First, the review identifies the main telemetry sources required for infrastructure and database anomaly detection, including metrics, logs, traces, events, database counters, query-level evidence, and topology data. Second, it compares major model families used between 2020 and 2025, ranging from statistical baselines and unsupervised learning to deep sequence models, graph methods, and language-model-assisted diagnosis. Third, it explains how database-specific signals should be incorporated so that anomalies are not reduced to generic host-level deviations. Fourth, it proposes a deployment methodology that connects detection with triage, runbook selection, guarded automation, and auditability. Fifth, it defines evaluation criteria suitable for mission-critical settings, where the operational cost of a missed incident or unsafe automated response can exceed the statistical cost of misclassification.

The paper is written as a critical review for researchers, cloud architects, database reliability engineers, and security operations teams. Its contribution is not a new detector, but a synthesis framework that can guide empirical studies and production implementation. The scope includes public cloud, private cloud, edge-adjacent infrastructure, managed relational databases, self-managed relational engines, and distributed data stores used for transactional or near-real-time analytical workloads.

3. REVIEW METHODOLOGY

A structured narrative review methodology was adopted. Literature was selected from peer-reviewed journals, conference proceedings, standards documents, and high-value technical sources published from 2020 to 2025. Search concepts covered cloud anomaly detection, AIOps, log analysis, multivariate time-series anomaly detection,

database workload monitoring, root-cause analysis, observability, zero trust, service mesh security, and automated tuning. The review prioritised studies that addressed operational environments, multi-service systems, database performance, cloud telemetry, or deployment constraints rather than purely synthetic anomaly problems.

Evidence was interpreted through three inclusion principles. First, the source had to contribute to at least one of four layers: telemetry design, model selection, database workload diagnosis, or operational response. Second, it had to be relevant to hybrid or cloud-native systems, even when its empirical setting involved a large cloud provider, microservice benchmark, or time-series dataset rather than a named hybrid cloud. Third, the study had to provide transferable insight into production constraints such as evolving logs, high dimensionality, latency, human intervention, interpretability, resource overhead, or security governance.

The synthesis was organised using a concept-centric approach. Individual papers were grouped according to anomaly signal type, modelling strategy, and operational function. This allowed comparison between infrastructure-centric methods, log-based methods, graph-based root-cause approaches, and database-specific monitoring techniques. Technical vendor documents were used cautiously and only where they described current operational capabilities in managed platforms, such as database insights and automatic tuning. They were not treated as independent evidence of model superiority.

Quality appraisal considered methodological transparency, clarity of telemetry definitions, realism of workloads, availability of baseline comparisons, discussion of false positives, and relevance to production deployment. Studies based only on static benchmark accuracy were interpreted conservatively because time-series anomaly evaluation can be sensitive to labelling practice and scoring windows (Lai *et al*., 2021; Darban *et al*., 2024). The outcome of the review is therefore an integrative framework rather than a pooled quantitative result.

Table 1: Comparative review of machine learning approaches for hybrid cloud and database anomaly detection

Model family	Best-fit telemetry	Strength in hybrid cloud or database operations	Main limitation	Representative sources
Statistical baselines and robust thresholds	Stable metrics, SLO indicators, connection counts, storage latency	Transparent, inexpensive, and strong for capacity limits or seasonal patterns	Weak with non-linear interactions and changing workload mixes	Lai *et al*., (2021); Wu and Keogh (2021)
Classical unsupervised learning	Host metrics, VM metrics, database counters, workload fingerprints	Useful when labels are scarce; supports fast deployment and interpretable feature ranking	Requires careful scaling, drift control, and domain-specific thresholds	Islam and Miranskyy (2020); Jiang (2024)
Log-based AIOps models	Application logs, database alerts, authentication logs, orchestration events	Captures semantic failures that may not appear in numeric metrics	Template drift and noisy log volume can reduce reliability	Jiang *et al.,* (2021); Wang *et al.,* (2024)
Deep sequence and transformer models	Multivariate time series, traces, workload sequences, latency paths	Models temporal dependencies and subtle multivariate shifts	High calibration effort; lower transparency without explanation layers	Tuli *et al.,* (2022); Darban *et al*., (2024)
Graph and knowledge-based models	Service topology, dependencies, flow logs, database relationships	Supports propagation analysis and likely root-cause ranking	Topology quality and graph freshness are difficult in hybrid estates	Mitropoulou *et al*., (2024); Hsieh *et al*., (2024)
LLM-assisted and causal diagnosis	Incident narratives, runbooks, traces, logs, metrics, database evidence	Improves triage summaries and hypothesis generation when grounded in telemetry	Risk of unsupported recommendations if not constrained by verified evidence	Pedroso *et al.,* (2025); Lin *et al.,* (2024)

4. Telemetry Foundations for Hybrid Cloud Anomaly Detection

The most important design decision in anomaly detection is not the algorithm but the observability substrate. Hybrid cloud workloads produce several telemetry classes. Infrastructure metrics describe CPU, memory, network throughput, disk latency, queue depth, container restarts, node pressure, and orchestration state. Application traces describe call paths and latency propagation. Logs provide event context, exceptions, authentication messages, configuration changes, and database warnings. Database telemetry adds engine-specific evidence such as wait events, active sessions, blocked processes, buffer cache behaviour, query plan changes, index usage, redo generation, vacuum or checkpoint activity, replication status, and storage commit latency.

Treating these sources separately creates blind spots. A database incident may begin as an index regression but appear first as pod latency, API timeout, or message queue growth. Conversely, a network partition may appear as database connection exhaustion without any change in query logic. Research on microservice failure analysis emphasises that symptoms and causes frequently occur in different services and different telemetry modalities (Soldani and Brogi, 2022; Yu *et al.,* 2023). For this reason, anomaly detection must correlate telemetry by service, database instance, tenant, deployment version, region, and time window.

Open observability standards and service-mesh guidance support this integrated view by encouraging consistent telemetry collection, secure service-to-service communication, and continuous monitoring across distributed applications (Chandramouli and Butcher, 2020; OpenTelemetry Authors, 2024). In hybrid cloud, consistency is especially important because telemetry may cross ownership boundaries. A private database team may manage wait-event data, while a cloud platform team manages container and network metrics, and a security team manages authentication and endpoint logs. The anomaly system must therefore maintain shared identifiers, including request IDs, trace IDs, database session IDs, workload tags, and change-ticket references.

Telemetry quality also determines model reliability. Noisy logs, missing values, sampling inconsistency, time-zone drift, and unannounced schema changes can create false anomalies. Self-adjusting observability research notes that collecting all logs can increase cost and degrade downstream AIOps tasks, making signal selection and adaptive sampling operational requirements rather than optional optimisations (Kabir *et al*., 2024). For mission-critical databases, telemetry retention must support incident reconstruction, capacity planning, and compliance audit. Short retention may be sufficient for dashboards but inadequate for seasonal baselining or forensic analysis.

A mature telemetry architecture should therefore include four safeguards. The first is completeness across infrastructure, application, and database layers. The second is semantic enrichment, including topology, ownership, deployment, workload class, and data criticality. The third is data quality monitoring, so missing telemetry becomes a detectable condition. The fourth is cost governance, because high-cardinality metrics and verbose logs can become financially and operationally unsustainable in large hybrid estates.

5. Machine Learning Approaches and Their Operational Suitability

Machine learning methods for anomaly detection can be grouped into five broad families. Statistical and forecasting methods establish normal seasonal behaviour and flag deviations. They are transparent and inexpensive but struggle with non-linear interactions. Classical machine learning, including isolation forests, one-class support vector machines, clustering, and tree ensembles, can handle higher-dimensional patterns and often provides useful baselines. Deep sequence methods, including recurrent networks, variational autoencoders, and transformer architectures, model complex temporal dependencies and multivariate telemetry, but they require careful calibration and can be difficult to explain (Tuli *et al*., 2022; Darban *et al*., 2024). Graph methods use service topology, dependency maps, and communication patterns to capture propagation and isolate likely causes. Knowledge-based and language-model-assisted approaches add operational context, runbook reasoning, and human-readable explanations (Mitropoulou *et al*., 2024; Pedroso *et al.,* 2025).

The best choice depends on the workload. Stable infrastructure indicators such as storage latency or connection pool saturation may be monitored effectively with seasonal baselines and robust thresholds. Database workloads with changing business calendars may require forecasting or adaptive baselines. Log streams with evolving templates may benefit from representation learning, but log parsing and tokenisation remain fragile when software versions change. Cross-system log detection is difficult because labels and event meaning differ between systems; meta-learning work has therefore focused on adaptation across target environments with limited labels (Wang *et al*., 2024).

For databases, unsupervised and semi-supervised approaches are often necessary because labelled incidents are scarce. A database team may have detailed records for major outages but very few labelled examples of subtle plan regressions or near-miss lock storms. Unsupervised database anomaly detection methods attempt to learn normal patterns in metrics and flag deviations without requiring large labelled datasets (Zhou, 2025). However, unsupervised output must be linked to operational meaning. A spike in active sessions is not automatically an incident; it may be normal during batch processing, month-end reporting, or promotional traffic. This is why context features are indispensable.

Deep learning should be used with caution in mission-critical contexts. It can improve detection in high-dimensional telemetry, but benchmark success does not guarantee production value. Time-series anomaly detection research warns that evaluation can be misleading when labels are weak, scoring windows are inconsistent, or anomalies are obvious from preprocessing artefacts (Lai *et al*., 2021; Wu and Keogh, 2021). In operations, an accurate but opaque model may be less useful than a slightly less accurate model that explains which workload, dependency, query, or resource changed.

A practical model portfolio is therefore preferable to a single detector. Baseline thresholds can detect clear saturation. Forecasting models can handle regular seasonality. Tree-based unsupervised models can detect unusual combinations of resource indicators. Sequence autoencoders or transformers can model temporal signatures. Graph and causal models can reason about propagation. Language-model-assisted tools can summarise evidence, but their recommendations should be grounded in verified telemetry and approved runbooks rather than treated as autonomous truth. This layered approach reduces dependence on any single modelling assumption.

6. Database Workload Anomaly Patterns

Mission-critical databases have anomaly signatures that differ from generic cloud services. The first pattern is workload-shape drift. Query volume, transaction mix, read-write ratio, tenant distribution, and batch timing may change without infrastructure failure. If a detector only observes CPU, it may miss that the cause is a new query pattern or altered application behaviour. The second pattern is plan regression. A query may remain syntactically unchanged while the optimiser chooses a less efficient plan because of stale statistics, changed parameter values, or index modification. Automatic tuning platforms explicitly target plan regression and index recommendations because these issues can degrade performance while ordinary host metrics appear acceptable (Microsoft, 2025).

The third pattern is contention. Lock waits, latch waits, thread pool saturation, and connection storms often show non-linear behaviour. A small rise in concurrency can move a database from stable throughput into cascading waits. The fourth pattern is storage-path instability. Commit latency, checkpoint duration, write-ahead log pressure, redo shipping, or snapshot overhead can surface as query latency while the root cause lies in storage or replication. The fifth pattern is security-related abnormality. Repeated failed logins, unusual privileged commands, abnormal export queries, and access from unexpected networks may indicate compromise or insider misuse. Zero-trust architecture treats resource access decisions as continuous and contextual, a principle that aligns with anomaly detection for database access patterns (Rose *et al*., 2020).

The sixth pattern is hybrid dependency failure. Database calls may cross cloud and private network boundaries, where latency is affected by gateways, encryption, private links, domain-name resolution, firewalls, and routing policy. Hybrid incidents can therefore look like database faults even when the database engine is healthy. The detector should include network path metrics and application retry behaviour to avoid misclassification.

Feature engineering should preserve database semantics. Useful features include deltas in top SQL wait contribution, plan hash changes, buffer hit ratio shifts, lock wait distribution, temp-space growth, replication lag gradients, p95 and p99 commit latency, session state distribution, deadlock frequency, and anomaly scores for privileged command sequences. These features should be joined with deployment changes, schema migrations, maintenance windows, data growth, and business calendars. Without these joins, a model may treat expected month-end reporting as abnormal or miss a dangerous slow burn because each isolated signal remains below a static threshold.

7. Proposed Reference Architecture

The proposed architecture contains six layers. The first layer is telemetry ingestion. It collects cloud metrics, private infrastructure metrics, container events, service traces, security logs, and database-native telemetry. The second layer is context normalisation. It attaches service ownership, business criticality, region, environment, database engine, schema version, deployment identifier, and dependency topology. The third layer is feature generation. It computes rolling statistics, seasonal baselines, embeddings, workload fingerprints,

change indicators, and graph features. The fourth layer is the model portfolio. It combines rule-based checks, statistical baselines, unsupervised detectors, sequence models, graph-based localisation, and explanation services. The fifth layer is decision governance. It ranks severity, maps anomalies to business impact, selects runbooks, checks confidence, and determines whether an action should be automated or require approval. The sixth layer is feedback. Incident outcomes, operator decisions, false positives, and remediation success are returned to the model registry.

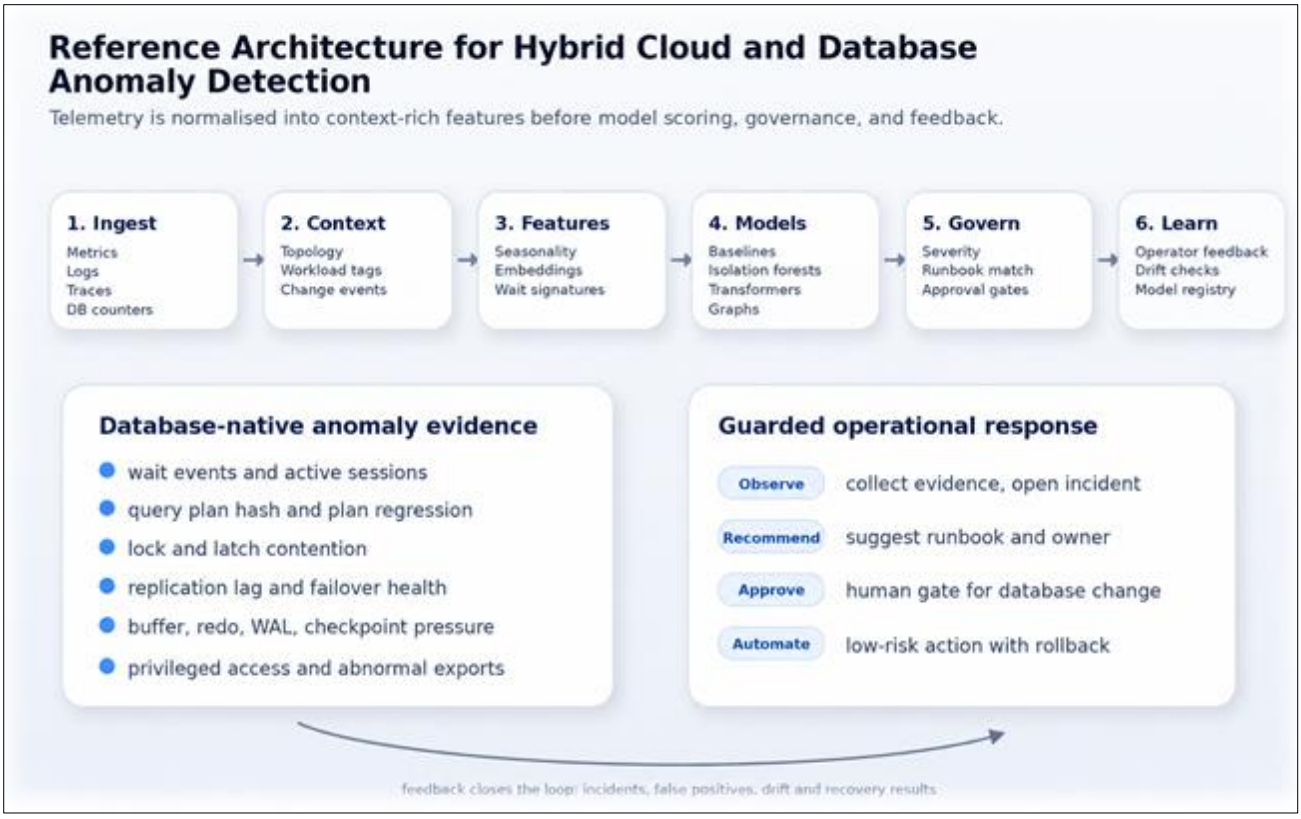

Figure 1: Reference architecture linking hybrid telemetry, database-native evidence, model governance and feedback

This architecture reflects evidence from cloud anomaly detection and self-healing research while separating detection from action. Cloud-native remediation can reduce recovery time, but automated actions in database environments can be dangerous if poorly scoped. Restarting an application pod may be low risk, whereas rebuilding an index, killing sessions, changing memory parameters, or promoting a replica can affect data integrity and availability. Therefore, response should be tiered. Low-risk responses may include evidence collection, ticket creation, trace sampling, scaling read replicas, or enabling enhanced diagnostics. Medium-risk responses may require on-call approval. High-risk actions, such as failover or destructive session termination, should require explicit safeguards and rollback paths.

The architecture also needs model governance. Each detector should have a documented objective, training window, input features, expected drift behaviour, threshold policy, and owner. Models should be tested against historical incidents and controlled chaos scenarios. New models should be introduced in shadow mode before alerting or action. This is consistent with reliability engineering practice: trust is built by showing that a detector adds signal without creating operational noise. Large-scale cloud studies show that production detectors must handle high dimensionality, changing workloads, and operational constraints that are not visible in small benchmark datasets (Islam *et al*., 2024).

Security must be built into the system. The anomaly platform will possess sensitive telemetry and may trigger privileged actions. It should enforce least privilege, isolate credentials, log all decisions, and protect model artefacts from tampering. Supply-chain and pipeline guidance for cloud-native applications highlights the importance of provenance, signed artefacts, and controlled deployment paths (Chandramouli *et al*., 2024). These principles apply equally to anomaly detectors and remediation scripts.

8. Evaluation Metrics and Evidence Standards

Evaluation must extend beyond model accuracy. Precision and recall remain useful, but they do not capture the business value of detecting a database incident early. For mission-critical workloads, evaluation should include mean time to detect, mean time to acknowledge, mean time to recover, false-alert burden, missed-incident severity, explanation quality, operator acceptance, and remediation safety. Cost metrics are also necessary because observability pipelines can be expensive, especially when traces and high-cardinality metrics are retained across several clouds.

Figure 2: Evaluation scorecard for mission-critical anomaly detection, balancing speed, safety, alert quality and learning

Ground truth is a persistent challenge. Incident tickets may record the final diagnosis but not all symptom timing. Labels may be delayed, incomplete, or biased towards major incidents. Near misses are rarely labelled. A reliable evaluation design should combine historical incidents, controlled fault injection, synthetic workload changes, replayed telemetry, and expert adjudication. It should report detection lead time relative to user impact, not only whether the anomaly was eventually detected. Cloud provider studies and workload characterisation research demonstrate the importance of production-scale data because real systems exhibit burstiness, tenant variability, and dependency effects that are difficult to reproduce in small laboratories (Shahrad *et al*., 2020; Islam *et al*., 2024).

For database workloads, benchmarks should include plan regression, lock contention, replica lag, storage latency, authentication anomalies, connection storms, and cross-region network degradation. The detector should be evaluated separately for each anomaly class because a model that performs well on CPU saturation may fail on query-plan drift. The evaluation should also measure action correctness. A recommendation is useful only if it routes the incident to the correct owner and proposes a safe next step.

Explainability deserves specific scoring. Operators need to know which signals changed, when they changed, which service or query was implicated, and why the model ranked the anomaly as severe. Graph and knowledge-based methods can help by connecting symptoms to topology and known failure modes (Mitropoulou *et al*., 2024; Pedroso *et al*., 2025). Explanations should be concise, evidence-backed, and reproducible. Overly fluent but ungrounded natural-language summaries can mislead incident teams, particularly under time pressure.

9. DISCUSSION

The literature indicates a clear direction: anomaly detection is moving from isolated thresholding towards integrated observability, machine learning, and root-cause reasoning. However, the operational frontier is not simply to deploy deeper models. It is to make models reliable under hybrid governance, mixed ownership, and database-critical risk. Hybrid cloud introduces heterogeneity in telemetry formats, identity models, network paths, and service ownership. Databases add transactional state, performance-plan complexity, and strict change-control requirements. These features make anomaly detection both essential and difficult.

Three design principles emerge. First, domain context should constrain machine learning. A model that knows the service topology, business calendar, deployment history, and database engine semantics will outperform a generic detector that sees only numeric metrics. Second, detection and response should be separated by governance. The model may classify an anomaly, but the action policy should decide whether to observe, escalate, recommend, or remediate. Third, evaluation should prioritise operational outcomes. A detector that slightly increases F1 score but doubles alert fatigue is not successful in a mission-critical environment.

There are also unresolved research gaps. Cross-cloud generalisation remains limited because telemetry schemas, managed-service abstractions, and tenant workloads differ widely. Label scarcity remains severe for rare but high-impact database incidents. Few studies evaluate detectors on combined infrastructure, trace, log, and database-native signals. Security anomalies and performance anomalies are also often studied separately, even though privilege misuse, data exfiltration, and denial-of-service behaviour can manifest as workload anomalies. Future studies should evaluate hybrid datasets that include both performance faults and adversarial behaviours.

Language models may improve triage by translating telemetry into incident narratives, but they must be grounded in evidence. Their most valuable role is likely summarisation, hypothesis ranking, and runbook retrieval, not unsupervised execution of high-risk database actions. Combining language models with Bayesian networks, knowledge graphs, and verified telemetry appears promising because it balances explanation with structure (Pedroso *et al*., 2025).

10. Practical Implications

For practitioners, the first implication is to begin with telemetry hygiene. Machine learning cannot compensate for missing database wait events, inconsistent trace identifiers, or untagged services. The second implication is to build a model portfolio rather than a single enterprise detector. Different anomaly classes require different methods. The third implication is to include database engineers in feature design. Host metrics alone cannot explain plan regression, lock contention, or replication delay. The fourth implication is to implement response guardrails before automation. Incident teams should decide in advance which actions can be automatic and which require approval.

Table 2: Implementation and evaluation framework for mission-critical database workloads

Implementation phase	Design requirement	Database-specific emphasis	Evaluation evidence	Operational control
Telemetry readiness	Common timestamps, identifiers, tags, retention and quality checks	Wait events, query plans, locks, replication, buffer/cache, redo/WAL	Missingness rate, clock skew, retention coverage	Data quality alerts and ownership catalogue
Feature engineering	Join metrics, logs, traces, topology and change data	Plan hash drift, top SQL contribution, session state, p99 commit latency	Feature stability, drift reports, expert review	Versioned feature store and lineage
Model deployment	Portfolio of baselines, unsupervised models, sequence models and graph methods	Separate models for plan regression, contention, lag, storage and access anomalies	Precision, recall, lead time, false-alert cost	Shadow mode, canary alerting, model cards
Triage and explanation	Evidence-backed severity and root-cause ranking	Link symptom to query, session, dependency, or engine resource	Explanation usefulness and owner accuracy	Runbook mapping and reviewer feedback

Implementation phase	Design requirement	Database-specific emphasis	Evaluation evidence	Operational control
Response governance	Tiered actions by risk and reversibility	No destructive database action without approval and rollback plan	MTTR, action success, rollback frequency	Approval gates, audit logs, least privilege
Continuous learning	Closed-loop feedback from incidents and false positives	Update baselines after releases and schema changes	Drift rate, retraining impact, operator trust	Post-incident model review board

For researchers, the main implication is the need for more realistic datasets. Benchmarks should include database internals, hybrid network dependencies, deployment events, and operator feedback. They should also measure early detection and safe diagnosis, not merely point-wise classification. For governance teams, anomaly detection should be treated as a controlled operational capability with audit trails, access controls, model documentation, and periodic review.

11. Limitations

This review is limited by the maturity of published evidence. Many studies use cloud or microservice datasets that do not expose full database internals. Vendor documentation describes useful platform capabilities but cannot establish independent comparative performance. Preprint studies provide timely insights but may change after review. The synthesis is therefore interpretive rather than exhaustive. It is also intentionally focused on operational anomaly detection, not fraud analytics, generic intrusion detection, or application-level business anomaly detection.

12. Research Agenda

A research agenda for this field should move in four directions. The first is benchmark construction. Public datasets should represent hybrid reality: managed and self-managed databases, private and public network paths, routine maintenance, failed releases, noisy neighbours, credential misuse, and recovery actions. Dataset documentation should describe telemetry sampling, clock alignment, missingness, injected or naturally occurring incidents, and the business meaning of labels. This would reduce the risk that models learn artefacts rather than operational causes.

The second direction is cross-modal fusion. Current systems often analyse metrics, logs, traces, and database evidence in separate pipelines. Future detectors should learn relationships between query fingerprints, trace spans, lock graphs, storage latency, and deployment changes. Graph models are suitable for this task because they can represent service calls, database dependencies, ownership, and fault propagation in one structure (Hsieh *et al*., 2024; Zhu *et al*., 2024). Nevertheless, graph outputs must remain interpretable enough for on-call staff.

The third direction is resilient learning under drift. Hybrid estates change continuously through scaling, schema evolution, new application releases, security patches, and cloud service updates. Models should detect drift, request retraining, and preserve old baselines for audit. Shadow deployment, canary alerting, and rollback of model versions should become normal practices, much like software release controls.

The fourth direction is safe action learning. Reinforcement learning and adaptive policy engines may eventually optimise response selection, but database operations require stronger safeguards than ordinary infrastructure healing. Future work should formalise action risk, dependency blast radius, approval thresholds, and verification checks. Anomaly detection should therefore mature into a closed-loop reliability discipline in which models observe, explain, recommend, act within limits, and learn from reviewed outcomes.

Finally, collaborative evaluation is needed. Cloud engineers, database administrators, application owners, and security analysts should score explanations jointly because each role sees a different part of the incident chain. A detector that satisfies a data scientist may still fail an operator if it cannot distinguish harmless batch pressure from customer-impacting degradation. Conversely, a rule preferred by an operator may miss novel dependencies introduced by platform change. Shared evaluation workshops, reproducible incident replays, and post-incident model reviews can turn anomaly detection from an isolated analytics project into a learning system. This cultural layer is essential in mission-critical environments where accountability, compliance, and service trust are as important as algorithmic novelty. It also supports fair comparison between vendors, open-source tools, and in-house models without hiding operational risk behind headline accuracy and transparent continuous improvement.

13. CONCLUSION

Machine learning-based anomaly detection can materially improve the resilience of hybrid cloud infrastructure and mission-critical database workloads, but only when it is embedded within a disciplined operational architecture. The evidence from 2020 to 2025 supports a layered approach that combines telemetry quality, database-aware feature engineering, multiple model families, explainable root-cause reasoning, and guarded response. The most effective systems are not those that replace human expertise, but those that place timely, contextual, and trustworthy evidence in front of the right engineering team before service impact escalates. Future progress will depend on production-scale hybrid datasets, stronger database-specific benchmarks, integrated security and performance analysis, and governance models that make automated detection both effective and safe.

REFERENCES

- Amazon Web Services. (2024). Amazon CloudWatch Database Insights. AWS What's New.
- Chandramouli, R., & Butcher, Z. (2020). Building secure microservices-based applications using service-mesh architecture. NIST Special Publication 800-204A. https://doi.org/10.6028/NIST.SP.800-204A
- Chandramouli, R., Kautz, F., & Torres-Arias, S. (2024). Strategies for the integration of software supply chain security in DevSecOps CI/CD pipelines. NIST Special Publication 800-204D. https://doi.org/10.6028/NIST.SP.800-204D
- Darban, Z. Z., Webb, G. I., Pan, S., Aggarwal, C. C., & Salehi, M. (2024). Deep learning for time series anomaly detection: A survey. ACM Computing Surveys.
- Ding, R., Zhang, C., Wang, L., Xu, Y., Ma, M., Wu, X., Zhang, M., Chen, Q., Gao, X., Gao, X., Fan, H., Rajmohan, S., Lin, Q., & Zhang, D. (2023). TraceDiag: Adaptive, interpretable, and efficient root cause analysis on large-scale microservice systems. Proceedings of ESEC/FSE 2023.
- Hsieh, K., Wong, M., Segarra, S., Mani, S. K., Eberl, T., Panasyuk, A., Netravali, R., Chandra, R., & Kandula, S. (2024). NetVigil: Robust and low-cost anomaly detection for east-west data center security. Proceedings of NSDI 2024.
- Islam, M. S., & Miranskyy, A. (2020). Anomaly detection in cloud components. IEEE International Conference on Software Maintenance and Evolution.
- Islam, M. S., Pourmajidi, W., Zhang, L., Steinbacher, J., Erwin, T., & Miranskyy, A. (2021). Anomaly detection in a large-scale cloud platform. IEEE conference proceedings.
- Islam, M. S., Rakha, M. S., Pourmajidi, W., Sivaloganathan, J., Steinbacher, J., & Miranskyy, A. (2024). Anomaly detection in large-scale cloud systems: An industry case and dataset. arXiv preprint.
- Jiang, G. (2024). Artificial intelligence-based adaptive anomaly detection technology for IaaS cloud virtual machines. Journal of Engineering and Applied Science, 71, 102.
- Jiang, Z., Li, T., Zhang, Z., Ge, J., You, J., & Li, L. (2021). A survey on log research of AIOps: Methods and trends. Mobile Networks and Applications, 26, 2353-2364.
- Kabir, M. A., Brown, C., Cohen, S., & Ghanbari, H. (2024). Self-adjusting log observability for cloud-native applications. IEEE International Conference on Cloud Computing.
- Lai, K. H., Zha, D., Wang, G., Xu, J., Zhao, Y., Kumar, D., Chen, Y., Zumkhawaka, P., Wan, M., Martinez, D., & Hu, X. (2021). Revisiting time series outlier detection: Definitions and benchmarks. NeurIPS Datasets and Benchmarks Track.
- 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. arXiv preprint.
- Mahgoub, A., Medoff, A. M., Kumar, R., Mitra, S., Klimovic, A., Chaterji, S., & Bagchi, S. (2020). OPTIMUSCLOUD: Heterogeneous configuration optimization for distributed databases in the cloud. USENIX Annual Technical Conference 2020, 189-203.
- Microsoft. (2025). Automatic database tuning. Microsoft Learn.
- Mitropoulou, K., Kokkinos, P., Soumplis, P., & Varvarigos, E. (2024). Anomaly detection in cloud computing using knowledge graph embedding and machine learning mechanisms. Journal of Grid Computing, 22, Article 6.
- Notaro, P., Cardoso, J., & Gerndt, M. (2020). A systematic mapping study in AIOps. arXiv preprint.
- OpenTelemetry Authors. (2024). OpenTelemetry specification: Logs, metrics, traces and semantic conventions.
- Pedroso, D. F., Almeida, L., Pulcinelli, L. E. G., Aisawa, W. A. A., Dutra, I., & Bruschi, S. M. (2025). Anomaly detection and root cause analysis in cloud-native environments using large language models and Bayesian networks. IEEE Access, 13, 77550-77564.
- Rose, S., Borchert, O., Mitchell, S., & Connelly, S. (2020). Zero trust architecture. NIST Special Publication 800-207. https://doi.org/10.6028/NIST.SP.800-207
- Shahrad, M., Fonseca, R., Goiri, I., Chaudhry, G., & Bianchini, R. (2020). Serverless in the wild: Characterizing and optimizing the serverless workload at a large cloud provider. USENIX Annual Technical Conference 2020.

- 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 53.
- The Kubernetes Authors. (2024). Configure liveness, readiness and startup probes. Kubernetes Documentation.
- Tuli, S., Casale, G., & Jennings, N. R. (2022). TranAD: Deep transformer networks for anomaly detection in multivariate time series data. Proceedings of the VLDB Endowment, 15(6), 1201-1214.
- Wang, Y., Mantyla, M. V., Nyyssola, J., Ping, K., & Wang, L. (2024). Cross-system software log-based anomaly detection using meta-learning. arXiv preprint.
- Wu, R., & Keogh, E. (2021). Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress. arXiv preprint.
- Yu, G., Chen, P., Li, Y., Chen, H., Li, X., & Zheng, Z. (2023). Nezha: Interpretable fine-grained root causes analysis for microservices on multi-modal observability data. Proceedings of ESEC/FSE 2023, 553-565.
- Zhou, W., Sun, P., Zhou, X., Zang, Q., Xu, J., Zhang, T., Li, G., & Wu, F. (2025). DBAIOps: A reasoning LLM-enhanced database operation and maintenance system using knowledge graphs. arXiv preprint.
- Zhu, Y., Wang, J., Li, B., Tang, X., Li, H., Zhang, N., & Zhao, Y. (2024). Root cause localization for microservice systems in cloud-edge collaborative environments. arXiv preprint.