SHIELD-DBOps: An Admissibility-and-Verification Framework for Guardrail-Based Database Operations

Raghu Gollapudi
Fiserv Inc., Frisco, Texas, USA
reachraghu251@gmail.com (corresponding author)

Received: 22 April 2026 | Revised: 19 May 2026 and 25 May 2026 | Accepted: 27 May 2026

 DOI: https://doi.org/10.48084/etasr.19469

ABSTRACT

Enterprise production databases emit abundant telemetry, yet recurring database incidents are still often handled through disconnected alerts, scripts, Oracle-native point tools, and manual runbooks. The unresolved operational problem is not detection alone; it is the gap between detection and verified closure. In always-on financial and enterprise data environments, this gap is now operationally expensive because teams must control mixed Real Application Clusters (RAC), standby, and non-production estates under uptime, auditability, and staffing pressure. This paper presents SHIELD-DBOps (Standardized Handling of Incidents with Escalation Limits and Decision-guardrails), an external orchestration framework that standardizes four recurring incident classes through one supervisory control contract: TEMP/UNDO pressure, archive/FRA saturation, blocking sessions, and Data Guard apply lag. The technical contribution is a reusable admission-action-verification-handoff contract that separates database-operations policy from execution mechanics. Operational validation is reported on an 11-month ServiceNow incident series with class-level monthly counts and Mean Time To Resolve (MTTR). After production enablement in mid-May 2025, class-level MTTR remained below that of the transition month across all four classes, and incident counts declined overall. The results are descriptive and deployment-specific; phased rollout, mid-series policy tuning, and the absence of a control group preclude causal claims.

Keywords-self-healing database operations; database reliability engineering; Oracle operations; bounded automation; Data Guard; runbook orchestration

I. INTRODUCTION

Recurring Oracle operations incidents are often not hard to detect; they are hard to close safely and consistently after detection. TEMP pressure, Flash Recovery Area (FRA) growth, blocking chains, and lag usually arrive with recognizable signals; however, the response path in many enterprise Oracle environments is still split across Oracle Enterprise Manager (OEM) alerts, local scripts, ticket queues, and operator judgment. The current problem is that alert volume has grown faster than safe operational closure. An alert does not decide whether remediation is allowed, which executor may run, what signal proves recovery, or when automation must stop and escalate. In transaction-sensitive enterprise Oracle environments, that gap matters because delayed or inconsistent handling can amplify outage exposure, audit friction, and recovery risk [1-5].

The specific problem addressed here is narrower than "self-healing databases" [1-3]. In the reported Oracle enterprise environment, native Oracle tools and disciplined scripting already covered some of the work, but thresholds, exclusions, verification steps, and escalation paths were distributed across teams and shifts. That arrangement is fragile in current 24×7 database operations because the same incident class can be handled differently depending on shift, queue ownership, or local runbook interpretation. SHIELD-DBOps was developed to make four recurring infrastructure activities pass through one external, auditable decision path instead of several loosely coupled local responses.

The contribution is a deployed database reliability engineering pattern with a reusable operating grammar. SHIELD-DBOps does not propose another isolated runbook script; it defines a class-agnostic admissibility-and-verification control contract that makes permission to act, bounded execution, same-signal closure, and mandatory handoff explicit across heterogeneous incident classes. RMAN, DGMGRL, session control, and storage-growth actions remain tool-specific, but they operate under one shared policy contract that defines when automation is admissible, what action is permitted, what counts as successful clearance, and when control must return to human operators.

This paper makes four specific contributions. First, it formulates the operational problem as a missing control boundary between alert detection and verified closure. Second, it defines an admissibility-and-verification contract for recurring Oracle incidents, requiring persistent evidence before action, bounded authority during action, same-signal

verification before closure, and deterministic handoff when verification fails. Third, it shows how heterogeneous Oracle-native executors can be governed by one supervisory policy instead of disconnected runbooks. Fourth, it reports an 11-month production deployment using ServiceNow-derived incident counts, MTTR trends, escalation outcomes, and action-success evidence while limiting the claim to descriptive operational behavior rather than causal proof.

Therefore, the contribution is not a new Oracle remediation primitive, but a reusable database reliability engineering pattern for governing infrastructure remediation activities through consistent admission, bounded execution, same-signal verification, and explicit handoff. The value is in the control boundary: It turns the alert response from a collection of tactical scripts into a transferable operating model that can be inspected, tuned, and safely stopped.

Prior work spans autonomic and self-healing systems [1-3], outage and diagnosis studies [4, 5], automated DBMS tuning and self-driving database operation [6-9], and database security/readiness and replication models [10]. Recent self-healing work emphasizes validation and safe autonomy [11, 12], while incident-management/AIOps studies focus on triage, failure management, and operational learning [13-15]. Against this backdrop, SHIELD-DBOps keeps Oracle-native executors but standardizes admissibility, closure evidence, and handoff.

II. FRAMEWORK MODEL AND OPERATIONAL GUARDRAILS

SHIELD-DBOps fixes the alert-to-closure gap by treating remediation as a policy-governed control loop rather than a script launch. Every admitted incident must pass five stages: evidence collection, class admission, bounded action, same-signal verification, and deterministic closure or handoff. The framework is governed by three executor-independent invariants: bounded admissibility, same-signal closure, and deterministic handoff. Figure 1 summarizes the control contract at the framework level.

SHIELD-DBOps is not proposed as a new database engine, optimizer, anomaly detector, or learned remediation policy. Its contribution is at the database-reliability control layer. It tells the operations environment what to fix, why the fix is safe enough to attempt, how far automation is allowed to go, and what evidence proves the condition has cleared. This is needed in current always-on estates because alert-only automation can accelerate action without proving closure. SHIELD-DBOps instead makes each action admissible, bounded, verified, and stoppable, reducing discretionary interpretation while keeping unsafe or unverifiable cases out of automation.

TABLE I. OPERATING DIFFERENCE FROM CONVENTIONAL DBA AUTOMATION

Conventional DBA automation	**SHIELD-DBOps operating fix**
Scattered scripts and class-specific runbooks	One supervisory contract for admission, bounded action, verification, and handoff
Alert or task completion may be treated as closure	Same-signal verification is required before closure
Fragmented evidence and local escalation practice	Auditable policy trail with deterministic handoff when verification fails

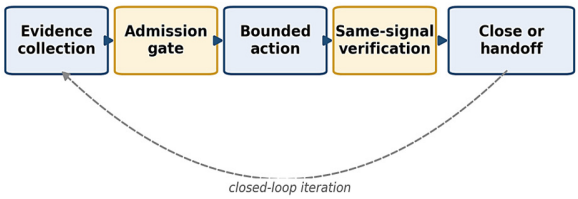

Fig. 1. SHIELD-DBOps closed-loop control path.

Viewed as a policy wrapper around Oracle-native operations, OEM alerts, and RMAN workflows, Data Guard commands and session-control actions are not treated as isolated automations. They are admitted only after persistence checks, bounded by explicit authority rules, and evaluated by a common post-action verification discipline. At runtime, the framework reads Oracle and infrastructure state, maps the observed condition to one of four supported classes or to no-action, checks persistence and exclusions, invokes a pre-mapped bounded action, and then verifies whether the same signal family has contracted. If verification fails, the system escalates rather than counting partial improvement as success. This preference for verified closure over forced automation is consistent with recent self-healing frameworks that treat safety and validation as explicit control stages [14, 15]. Operators can inspect which trigger fired, which exclusion blocked or permitted action, what actually ran, and which post-action signal caused closure or escalation.

The framework is narrower than the label "self-healing" may suggest. It does not attempt Structured Query Language (SQL) tuning, failover policy, capacity planning, or generalized anomaly response. It standardizes a set of repetitive Database Administrator (DBA) interventions whose triggers, authority boundaries, and stop conditions can be written down concretely before rollout. That limit is what makes the framework usable across production Real Application Clusters (RAC), standbys, and non-production systems without asserting that every database abnormality should be automated.

III. DEPLOYMENT METHOD

SHIELD-DBOps was deployed as an external control layer on designated automation hosts rather than as code embedded inside the Oracle engine. Policy lived in editable operational artifacts: trigger thresholds, exclusion lists, remediation mappings, and notification endpoints. That separation allowed policy tuning without changing database binaries or application logic. Deployment followed four phases. Shadow mode began on 10 February 2025 as read-only classification with no automated action; shadow classifications were logged to Splunk to examine class stability and false-positive patterns before actuation. Canary automation began on 22 March 2025 on three non-production clusters. Full production rollout began on 15 May 2025 across Primary Online Transaction Processing (OLTP) instances. A policy-tuning event on 5 August 2025 adjusted Data Guard restart thresholds to reduce Managed Recovery Process (MRP) flip-flopping during known network saturation windows and relaxed FRA backup triggers from 75% to 82%. Figure 2 summarizes the rollout sequence used to interpret the reported operational validation.

Fig. 2. Rollout sequence used for operational validation.

Scheduling remained simple. A centralized cron scheduler invoked the evaluation loop every 15 minutes from the control hosts. Each cycle collected fresh telemetry, evaluated class-specific rules, applied the mapped bounded action when permitted, and verified whether the unsafe state contracted before either closing the issue or paging the responsible channel.

The implementation combined Oracle dynamic performance views, Python 3.9 and shell wrappers, Ansible 2.14 playbooks, RMAN, Data Guard Manager Line (DGMGRL), and ServiceNow integration. Oracle telemetry was queried through cx_Oracle-backed wrappers over V$SESSION, V$TEMPSEG_USAGE, V$RECOVERY_FILE_DEST, V$DATAGUARD_STATS, and related DBA_* views; each cycle wrote a compact audit record containing class, trigger values, action ID, verification result, and escalation state. Representative bounded executors included 'ALTER TABLESPACE ... ADD TEMPFILE' for TEMP/UNDO, 'BACKUP ARCHIVELOG ALL DELETE INPUT' for FRA pressure, 'ALTER SYSTEM KILL SESSION' for non-excluded blockers, and a DGMGRL apply-off/apply-on restart of managed recovery for replication lag. These executor descriptions are included as transferability aids rather than release-ready playbooks. Runtime traces such as RMAN and DGMGRL outputs were used operationally, but the paper does not treat them as the reported dataset.

Class-specific policy uses three fields that matter operationally: trigger threshold, bounded action, and escalation condition. Exclusions are applied before actuation for protected accounts, sensitive execution windows, and system processes. Table II reports the operational rules used in the deployment.

TABLE II. CLASS-SPECIFIC TRIGGER, BOUNDED ACTION, AND ESCALATION CONDITION

Incident class	Trigger	Automated action	Escalate when
TEMP/UNDO pressure	SORT/UNDO > 85% for > 5 min	Add 5 GB tempfile or datafile	> 90% after 15 min or disk < 10%
Archive/FRA saturation	FRA utilization > 82%	RMAN archivelog backup and delete	FRA > 90% or ORA-/RMAN-error
Blocking sessions	Blocker > 300 s	ALTER SYSTEM KILL SESSION (non-excluded)	Count unchanged in 2 min or system process
Replication lag	Apply lag > 15 min	Restart managed recovery; verify contraction	MRP fails, ORA-16014, or lag grows

IV. OPERATIONAL VALIDATION

The operational validation of the framework is reported as monthly incident counts and monthly MTTR for the 11-month window from May 2025 through March 2026. This is a descriptive validation of a deployed framework in one enterprise Oracle environment. The reported series was derived only from ServiceNow incident records carrying a SHIELD-DBOps class tag and used ticket open and resolve timestamps as the sole MTTR clock. As in other operational outage studies, the series was used to characterize the observed behavior of the framework in production rather than to establish an isolated causal effect [4, 5]. Of the 95 monitored environments, approximately 19 were production RAC clusters, 14 were Active Data Guard standbys, 24 were pre-production databases, and 38 were development/UAT instances. Blocking-session burden was concentrated in production RAC tiers, while replication-lag burden was concentrated in standby environments. Monthly incident counts trend downward overall but not monotonically. TEMP/UNDO, Archive/FRA, blocking sessions, and replication lag all move downward across the series, but August and November still show renewed pressure.

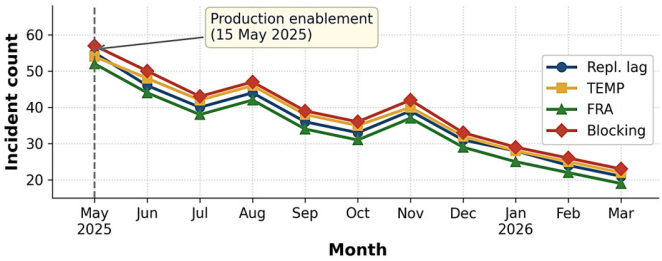

Fig. 3. Monthly incident counts by class, May 2025–March 2026.

Following production enablement, class-level MTTR was observed to be lower than the May level for every class.

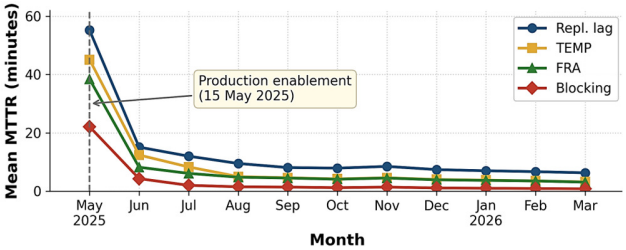

Fig. 4. Monthly MTTR by class, May 2025–March 2026.

Table III reports the monthly aggregated ServiceNow statistics in a compact, directly inspectable format. Each class cell reports ticket count and mean MTTR in minutes, preserving the core validation evidence while avoiding long wrapped text that obscures the trend.

TABLE III. MONTHLY AGGREGATED SERVICENOW TICKET STATISTICS BY CLASS (N/MEAN MTTR, MINUTES)

Month	Repl. lag n/mean	TEMP n/mean	FRA n/mean	Blocking n/mean
May 2025	55/55.2	54/45.1	52/38.4	57/22.1
Jun 2025	46/15.1	48/12.4	44/8.2	50/4.3
Jul 2025	40/12.0	42/8.3	38/6.1	43/2.0
Aug 2025	44/9.5	46/5.0	42/4.8	47/1.5
Sep 2025	36/8.1	38/4.6	34/4.5	39/1.4
Oct 2025	33/7.9	35/4.2	31/4.1	36/1.2
Nov 2025	39/8.5	40/4.6	37/4.5	42/1.4
Dec 2025	31/7.4	32/4.0	29/3.9	33/1.1
Jan 2026	28/7.0	28/3.8	25/3.7	29/1.0
Feb 2026	24/6.7	25/3.5	22/3.5	26/0.9
Mar 2026	21/6.3	22/3.2	19/3.1	23/0.8

The phased rollout sharpens interpretation. Shadow mode contributed only evidence collection, the canary affected three non-production clusters, and estate-wide production enablement began on 15 May. Quarterly 2024 operations summaries indicated that the same four incident classes were already persistent before automation, with overall monthly incident volume averaging roughly 54, 52, 57, and 55 across Q1 through Q4 2024; those summaries are retained only as background context because they were compiled outside the standardized monthly tagging pipeline. May 2025 is, therefore, a transition reference month rather than a clean pre-automation baseline. Descriptive monthly means are reported without statistical inference because phased rollout, operator learning, mid-series policy changes, and the absence of a control group preclude causal attribution. The narrower operational result is that admissibility, action authority, verification, and handoff became repeatable and auditable across four infrastructure activities previously dependent on local runbook interpretation.

Operational effectiveness was also assessed through the SHIELD-DBOps action-success and escalation profile. Success means closure after configured verification; escalation means verification failed or a handoff boundary was triggered. Correct handoff indicates that the post-incident review judged the escalation justified by policy rather than automation noise.

TABLE IV. OPERATIONAL ACTION-SUCCESS AND HANDOFF PROFILE

Class	Act.	Succ.	Esc.	Esc. rate	Correct handoff
TEMP/UNDO	410	395	15	3.7%	93.3%
Archive/FRA	373	355	18	4.8%	88.9%
Blocking	425	404	21	4.9%	90.5%
Repl. lag	397	362	35	8.8%	91.4%

These results support an operational-effectiveness claim rather than a causal-performance claim: SHIELD-DBOps made action eligibility, verification, and handoff visible and repeatable for recurring DBA activities.

To ground the reported validation in the dataset, Table V summarizes representative redacted ServiceNow field patterns used as evidence anchors across the four supported classes; these patterns are illustrative redactions rather than verbatim ticket excerpts.

TABLE V. REPRESENTATIVE REDACTED SERVICENOW EVIDENCE ANCHORS

Anchor	Redacted ServiceNow field pattern	Operational meaning
Incident summary	INC-[HASH] | PROD-RAC-[REDACTED] | TEMP usage 91%; class TEMP_UNDO admitted	TEMP/UNDO class tag and admission basis
Work-note sequence	2025-08-05T02:17Z | FRA 84.2%; RMAN cleanup invoked; reassess in 15 min	Archive/FRA action timing and follow-up state
Resolution note	2025-09-14T13:08Z | blocker SID/SERIAL cleared; blocked-session count contracted	Blocking-session closure evidence
Closure record	2025-11-21T04:42Z | apply lag below 15 min after MRP restart; state=Closed	Replication-lag closure timestamp supporting MTTR

V. DISCUSSION

The evidence supports a bounded experience-report claim. SHIELD-DBOps does not introduce a new Oracle command; it standardizes operational authority. The same admission, bounded-action, same-signal verification, and handoff rules govern TEMP/UNDO, FRA/archive, blocking-session, and Data Guard lag events. That policy wrapper coincided with lower class-level MTTR after rollout, but the result is best read as operational effectiveness rather than isolated causal proof because routing, authority clarity, and operator familiarity also changed during deployment.

The framework is useful only where triggers, actions, verification signals, and handoff lines are explicit. Blocking sessions remained the largest incident burden and the strongest leverage point; FRA and apply-lag behavior remained exposed to storage and network conditions outside framework control. Those boundaries limit the claim, but also make the framework safer: it standardizes repeatable infrastructure response without attempting unrestricted automation.

VI. OPERATIONAL LESSONS AND DEPLOYMENT LIMITS

For similar deployments, teams should instrument first, classify in shadow mode, tune thresholds against real incidents, and enable bounded actions only for stable classes. The transferable element is the control pattern, not the exact command string: stable trigger, bounded action, same-signal verification, and explicit handoff. Conservative rollout—shadow classification, canary automation, staged production expansion, and threshold retuning—kept automation auditable.

Residual risks remained. Before the August 2025 adjustment, managed-recovery restart could flip-flop during network saturation; blocking automation correctly refused excluded processes but still required review when a recurring batch workload shifted; one FRA remediation overlapped a backup window and needed manual assessment. SHIELD-DBOps also does not create SAN headroom, solve every archive gap, or guarantee schema-change rollback. Its role is to make database-layer response disciplined and stoppable across production RAC, standby, pre-production, and development tiers.

VII. TICKET EVIDENCE AND POLICY CONTRACT

Since this is a production deployment, claims are anchored in redacted ServiceNow fields: hashed incident identifiers, summaries, class tags, work notes, resolution notes, and closure timestamps. Splunk shadow logs, compact audit records, RMAN cleanup logs, and DGMGRL traces supported rollout review, but the monthly dataset remains ServiceNow-derived.

SHIELD-DBOps admitted only classes with a deterministic trigger, bounded action, objective verification signal, and clear handoff point. Across admitted classes, the trigger must persist, the post-action signal must contract before closure, and unresolved cases must exit automation through human handoff. Figure 5 and Table VI summarize this contract. TEMP/UNDO and Archive/FRA use conservative loops: standardized 5 GB file growth for TEMP/UNDO and RMAN backup/cleanup only after the FRA utilization boundary is crossed. Blocking and replication-lag loops require stricter runtime guardrails. Session termination is limited to non-excluded, non-system blockers; Data Guard restart is valid only when applying lag contracts after the managed-recovery action.

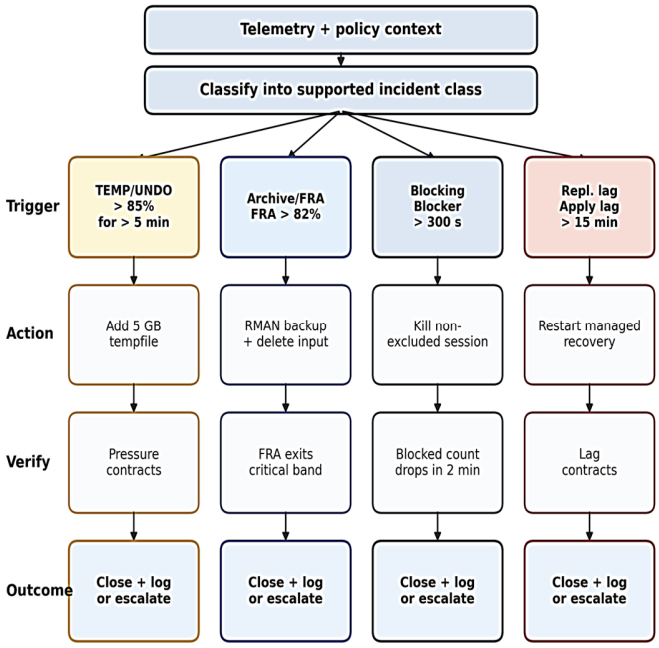

Fig. 5. Policy execution path and bounded actions used by the framework.

The following algorithm shows the control loop in pseudocode. Table VI summarizes the admissibility, verification, and handoff lines for each class.

```
collect x(t) and policy context e
c <- classify(x)   # supported class or no-
                   #action
if not admissible(c, x, e): log and wait
run bounded action A(c)
verify same signal family after action
if verification fails: escalate;
else close and log
```

TABLE VI. COMPACT POLICY CONTRACT BY INCIDENT CLASS

Class	Admissibility	Verification	Handoff
TEMP/UNDO	> 85% for > 5 min; volume has headroom	Pressure falls below emergency band after 5 GB add	> 90% after 15 min or disk < 10%
Archive/FRA	FRA > 82%; RMAN cleanup permitted	RMAN completes; FRA exits critical band	> 90% or ORA-/RMAN-failure
Blocking	> 300 s; not excluded; not system-owned	Blocked-session count drops in 2 min	System process or count unchanged
Repl. lag	Lag > 15 min; standby reachable; restart allowed	MRP restarts; apply lag contracts	MRP fails, ORA-16014, or lag grows

VIII. THREATS TO VALIDITY

This single-site validation has concrete confounders: phased rollout, August 2025 policy tuning, operator learning, and mixed production RAC, standby, pre-production, and development tiers. The curve should be read as operational behavior across rollout and tuned-policy phases, not one homogeneous treatment window.

Measurement also depends on ServiceNow tagging and ticket open/resolve discipline. Some MTTR compression may reflect faster paging or dispatch, not only faster database action; therefore, no formal inferential test is reported [4, 5].

Reproducibility is partial because raw ticket exports, traces, and playbooks cannot be released from a sensitive enterprise Oracle environment. Transferability is supported through trigger/action logic, rollout dates, escalation aggregates, redacted ticket-field patterns, and representative executor snippets, consistent with practice-oriented incident-management and self-healing reports [13-15].

Scope remains limited: SHIELD-DBOps does not solve SAN exhaustion, every archive-gap scenario, or schema-change rollback.

IX. CONCLUSION

This paper presented SHIELD-DBOps as a guardrail-based orchestration framework for four recurring Oracle operational incident classes: TEMP/UNDO pressure, archive/FRA saturation, blocking sessions, and Data Guard apply lag. Rather than proposing a universal autonomous database controller, the framework contributes a reusable operational control contract that requires persistent admission before action, bounded execution through approved Oracle-native mechanisms, same-signal verification before closure, and deterministic escalation when verification fails. The core contribution is architectural and operational: it standardizes how action authority is granted, how closure is proven, and how unsafe or unverifiable cases are handed back to human operators.

The 11-month ServiceNow-derived validation indicates that, after production enablement, class-level MTTR remained below the May transition reference across all four classes, incident counts trended downward overall, and most

escalations represented justified policy handoffs rather than automation noise. These observations support an operational-effectiveness claim. In the reported enterprise Oracle environment, SHIELD-DBOps improved the discipline of recurring DBA response by making admissibility rules, executor boundaries, closure evidence, and handoff conditions explicit, repeatable, and auditable across heterogeneous incident classes that previously depended more heavily on local scripts and shift-level runbook interpretation.

This study is intentionally bounded. It reflects one enterprise setting, a phased rollout, August 2025 policy tuning, operator learning effects, and redacted incident data that cannot be fully released from a sensitive production environment. Accordingly, the findings should be read as descriptive deployment evidence rather than isolated causal proof. Even with those limits, the field-level value remains clear: SHIELD-DBOps shows that Oracle-native remediation tools can be governed by a shared supervisory policy that is inspectable, tunable, and stoppable, allowing teams to reduce discretionary variation without claiming unrestricted automation.

For practitioners, the transferable lesson is not any single script or command sequence. It is the control pattern itself: define explicit admission criteria, restrict action authority, verify closure using the same signal family that triggered intervention, and escalate deterministically when verification fails. That pattern can be adapted to other database reliability engineering environments where the operational challenge is not merely detecting incidents, but closing them safely, consistently, and with auditable evidence.

DECLARATION OF COMPETING INTERESTS

The author is employed by Fiserv Inc., whose environment supplied the aggregated operational setting. The employer did not sponsor manuscript preparation or control the reported analysis.

ACKNOWLEDGMENT

Not applicable to this work.

DATA AVAILABILITY

Redacted ServiceNow aggregates are not public due to enterprise data-governance constraints. The manuscript reports monthly trends, escalation aggregates, rollout milestones, trigger/action logic, representative ticket fields, and transferable deployment details. Additional redacted excerpts may be shared upon reasonable request.

REFERENCES

[1] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, Jan. 2003, https://doi.org/10.1109/MC.2003.1160055.

[2] H. Psaier and S. Dustdar, "A survey on self-healing systems: approaches and systems," *Computing*, vol. 91, no. 1, pp. 43–73, Jan. 2011, https://doi.org/10.1007/s00607-010-0107-y.

[3] J. Chandra and P. Manhas, "Efficient and Scalable Self-Healing Databases Using Meta-Learning and Dependency-Driven Recovery." arXiv, July 18, 2025, https://doi.org/10.48550/arXiv.2507.13757.

[4] H. S. Gunawi *et al.*, "Why Does the Cloud Stop Computing?: Lessons from Hundreds of Service Outages," in *Proceedings of the Seventh ACM Symposium on Cloud Computing*, Oct. 2016, pp. 1–16, https://doi.org/10.1145/2987550.2987583.

[5] Y. Chen *et al.*, "Outage Prediction and Diagnosis for Cloud Service Systems," in *The World Wide Web Conference*, May 2019, pp. 2659–2665, https://doi.org/10.1145/3308558.3313501.

[6] D. Van Aken, A. Pavlo, G. J. Gordon, and B. Zhang, "Automatic Database Management System Tuning Through Large-scale Machine Learning," in *Proceedings of the 2017 ACM International Conference on Management of Data*, May 2017, pp. 1009–1024, https://doi.org/10.1145/3035918.3064029.

[7] G. Li *et al.*, "openGauss: an autonomous database system," *Proceedings of the VLDB Endowment*, vol. 14, no. 12, pp. 3028–3042, July 2021, https://doi.org/10.14778/3476311.3476380.

[8] D. Van Aken *et al.*, "An inquiry into machine learning-based automatic configuration tuning services on real-world database management systems," *Proceedings of the VLDB Endowment*, vol. 14, no. 7, pp. 1241–1253, Mar. 2021, https://doi.org/10.14778/3450980.3450992.

[9] A. Pavlo *et al.*, "Make your database system dream of electric sheep: towards self-driving operation," *Proceedings of the VLDB Endowment*, vol. 14, no. 12, pp. 3211–3221, July 2021, https://doi.org/10.14778/3476311.3476411.

[10] A. Albugmi, "Digital Forensics Readiness Framework (DFRF) to Secure Database Systems," *Engineering, Technology & Applied Science Research*, vol. 14, no. 2, pp. 13732–13740, Apr. 2024, https://doi.org/10.48084/etasr.7116.

[11] C. Tan *et al.*, "A Knowledge-driven Self-healing Dual-loop and Validation for Autonomous Networks," in *Proceedings of the 8th Asia-Pacific Workshop on Networking*, Aug. 2024, pp. 185–186, https://doi.org/10.1145/3663408.3665811.

[12] G. White, L. L. Custode, and O. O'Brien, "SASH: Safe Autonomous Self-Healing," in *Service-Oriented Computing – ICSOC 2022 Workshops*, vol. 13821, J. Troya, R. Mirandola, E. Navarro, A. Delgado, S. Segura, G. Ortiz, C. Pautasso, C. Zirpins, P. Fernández, and A. Ruiz-Cortés, Eds. Springer Nature Switzerland, 2023, pp. 142–153.

[13] Z. Chen et al., "Towards intelligent incident management: why we need it and how we make it," in Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Nov. 2020, pp. 1487–1497, https://doi.org/10.1145/3368089.3417055.

[14] P. Notaro, J. Cardoso, and M. Gerndt, "A Survey of AIOps Methods for Failure Management," *ACM Transactions on Intelligent Systems and Technology*, vol. 12, no. 6, pp. 1–45, Dec. 2021, https://doi.org/10.1145/3483424.

[15] Y. Dang, Q. Lin, and P. Huang, "AIOps: Real-World Challenges and Research Innovations," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, May 2019, pp. 4–5, https://doi.org/10.1109/ICSE-Companion.2019.00023.