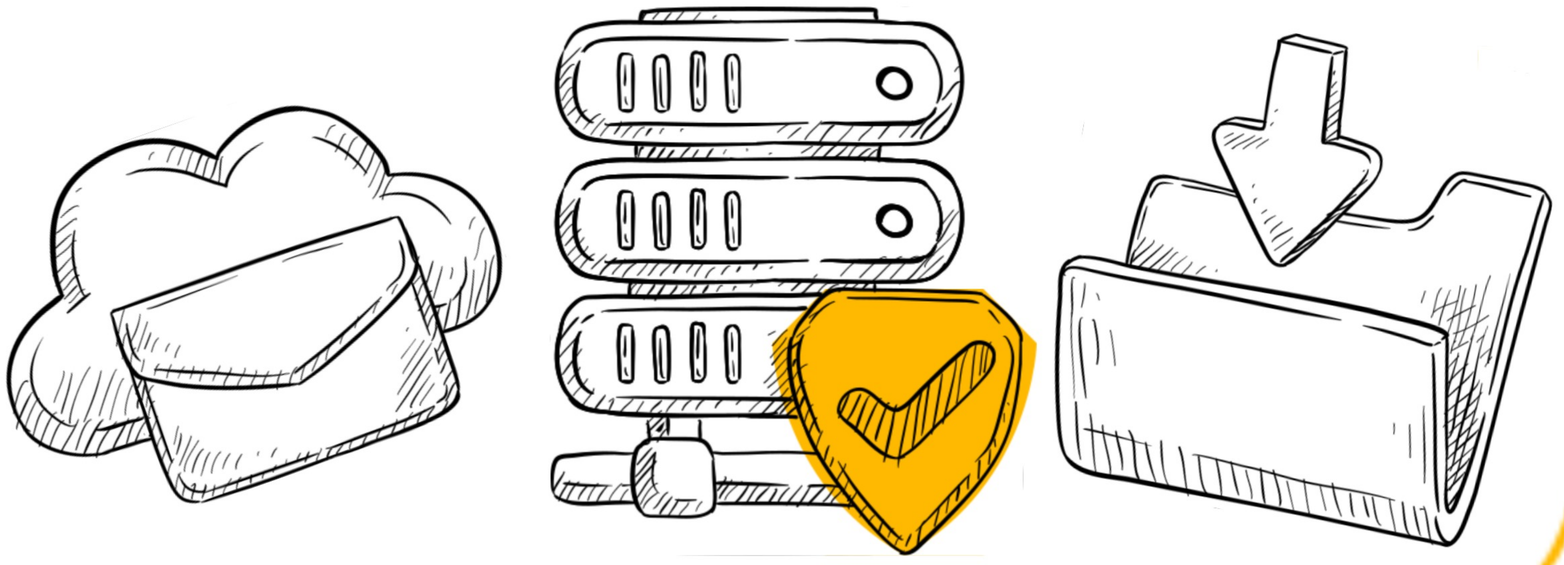
ManageEngine

HOW TO PREVENT THE DREADED DBA BURNOUT

24+ hours/week of firefighting vs 2 hours of strategy



AGENDA

What we'll cover in the next 60 minutes

ManageEngine

01 **Understand the DBA burnout reality**
Why 30%+ of the workweek vanishes into firefighting - and the visibility gap leadership doesn't see.

02 **Why traditional monitoring fails**
Hybrid complexity, tool sprawl, and issues that stay invisible until it's too late.

03 **The database observability framework**
Monitoring vs. observability, and the Find → Analyze → Fix model.

04 **Turn insights into action**
Faster detection, root-cause correlation, and cost-aware performance.

05 **How ManageEngine helps**
A practical path to preventing DBA burnout for good.

06 **Live demo**
Watch Alert → Root Cause → Resolution happen in minutes, on a real use case.

ManageEngine

01

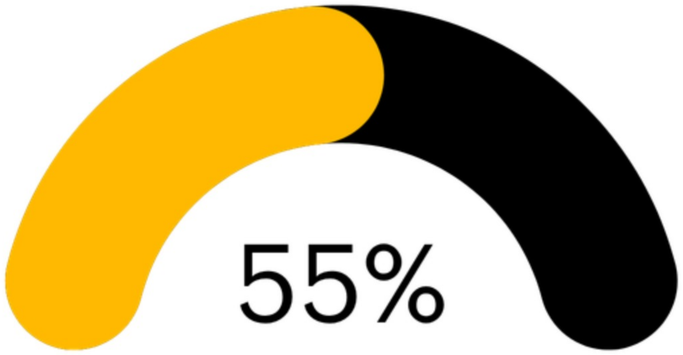
The DBA burnout reality

Why so much of the week disappears into reactive work and what it's costing your team

ManageEngine

55% of your week is spent on DB issues

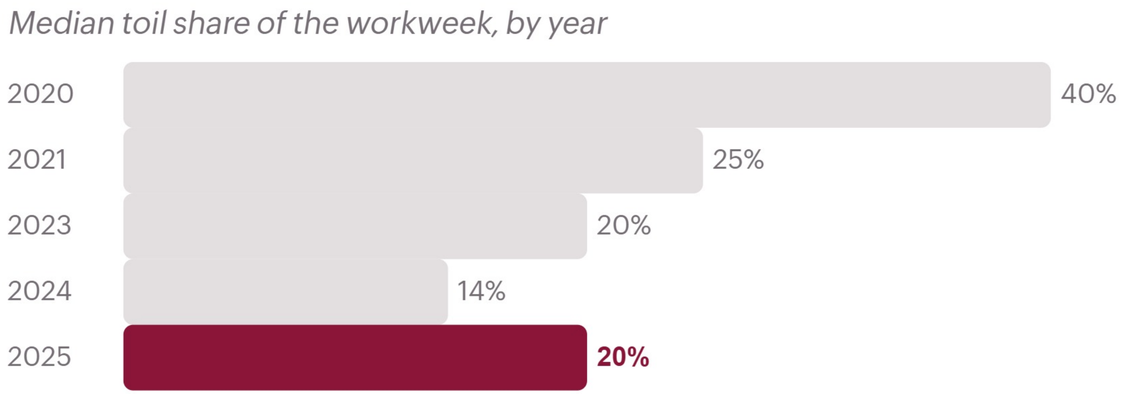
Across multiple independent industry studies, operations and database teams spend roughly 60% of every workweek — more than 24 hours — on reacting after something has already broken.



ManageEngine

Operational toil is rising, not falling

20% *median share of the workweek IT reliability teams spend on manual, repetitive, non-value-add work, rising for the first time in five years of tracking.*



The trend is moving the wrong way

Toil was declining, but in 2025 it reversed, and ops work rose from 25% to 30% of the week. AI hasn't fixed this reactive workload. When busywork eats this much time, strategic planning and tuning simply get pushed aside.

Source: Catchpoint, The SRE Report 2025 — 301 reliability professionals, Jul–Aug 2024.

ManageEngine

The burnout reality: What fills a DBA's week

A week that is dominated by reaction, not strategy.

Alert response & triage
Constantly monitoring and responding to incoming alerts across systems.

Incident troubleshooting & emergency fixes
Diagnosing and patching problems under pressure, in real time.

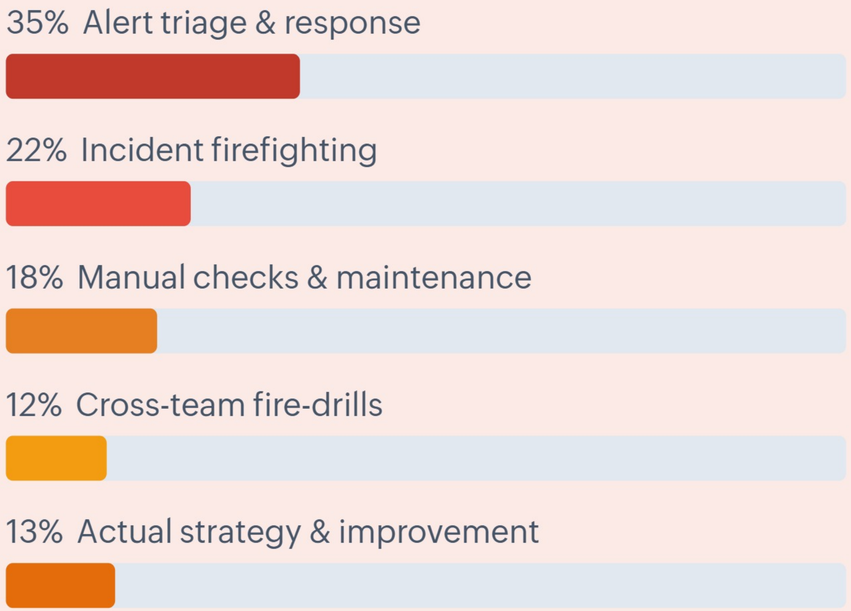
Performance complaints & escalations
Chasing down “why is it slow” tickets from frustrated users.

Cross-team coordination on recurring issues
Re-explaining the same incidents to ops, SREs, and leadership.

Manual backups, patching & maintenance
Routine upkeep that still demands hands-on attention.

BEFORE

Without observability



ManageEngine

Firefighting doesn't just cost hours

44%

of organizations had an **outage** in the past year tied directly to a suppressed or ignored alert

~70%

of SREs say **on-call stress** has contributed to burnout and attrition on their team

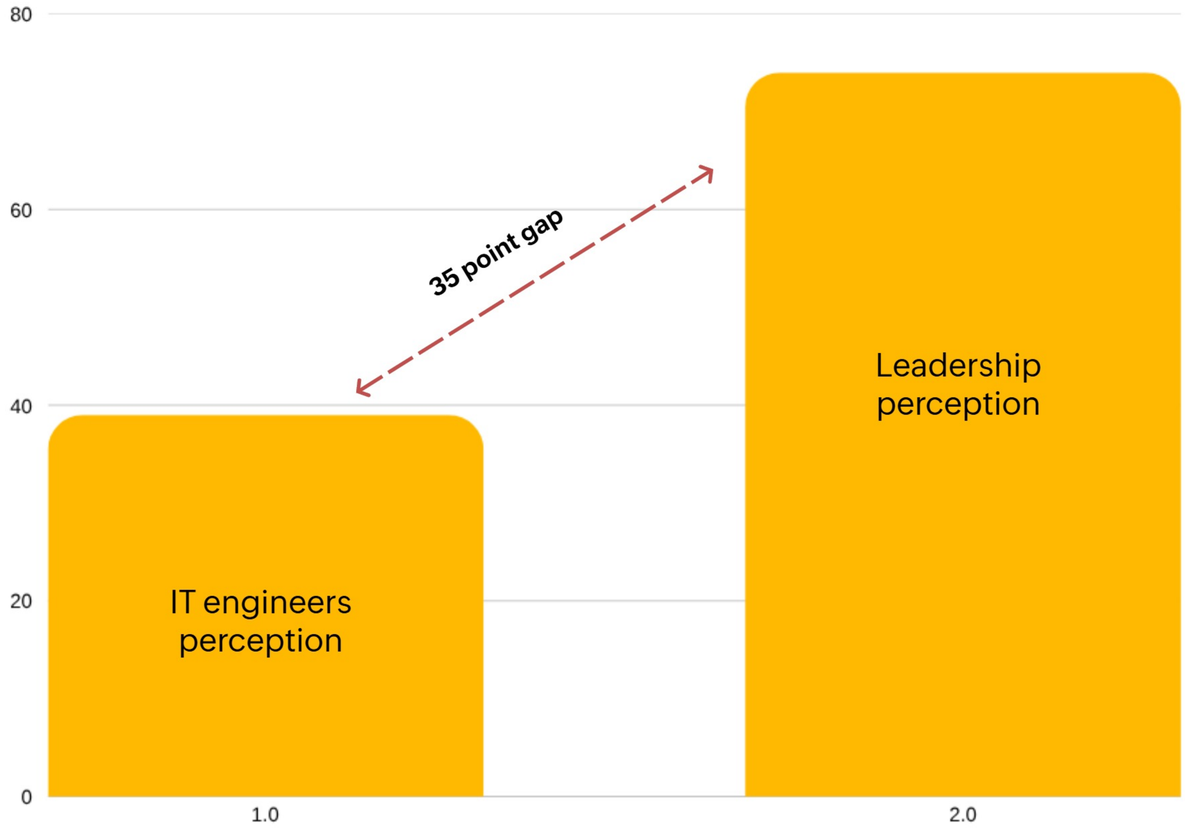
57%

of on-call teams say **fewer than 30% of the alerts they receive are actually actionable**

Sources: NeuBird AI, 2026 State of Production Reliability and AI Adoption Report (44%, 57%) · Catchpoint, The SRE Report 2025 (on-call burnout figure).

ManageEngine

Leadership thinks they see it all



A 35-point perception gap

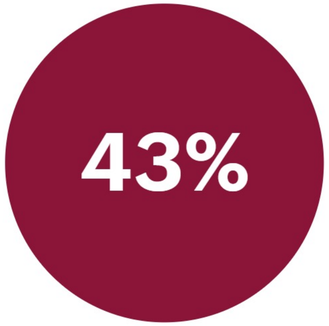
74% of executives believe their organization is already using AI to fix alerting and reliability problems — but only 39% of engineers agree. That gap becomes a **gap in budget, staffing, and timeline expectations** for the people closest to the problem.

Source: NeuBird AI, 2026 State of Production Reliability and AI Adoption Report — 1,039 SRE, DevOps and IT operations professionals, surveyed Feb 2026.

ManageEngine

The human cost of constant firefighting

Stress compounds — and so does the risk to your team and your business.



of ITOps teams say they **spend too much** time responding to alerts

of organizations have **experienced outages** due to ignored or suppressed alerts

average cost per customer-impacting incident — up 43% year over year

ManageEngine

02

Why traditional monitoring fails

The structural reasons reactive tools keep teams trapped in the loop

ManageEngine

Hybrid cloud & multi-database complexity

On-premises systems
Legacy infrastructure still running mission-critical workloads

Private & Public Cloud
AWS, Azure, GCP — each with its own tools

Relational Databases
Oracle, SQL Server, PostgreSQL, MySQL

NoSQL & Document Stores
MongoDB, Cassandra, and beyond

Cloud-Native Managed Services
RDS, Azure SQL, Cloud SQL

Multi-Cloud Sprawl
Different consoles, different baselines

73% of organizations now **run a hybrid cloud model** — up 3 points year over year. Every one of those environments still needs a database strategy that spans both sides, on top of the engine sprawl above.

Source: Flexera, 2026 State of the Cloud Report — 753 cloud decision-makers surveyed.

ManageEngine

Death by a thousand dashboards

79% of enterprises now runs on more than one database platform— each demanding its own specialist tool. No one sees the whole picture.

29%

of enterprises run **5+ database platforms** simultaneously

Disconnected dashboards

A **different console** for every engine, every cloud, every layer — none of them talking to each other.

Noisy, duplicate alerts

The same incident triggers **several different alerts** in several different places simultaneously.

No single source of truth

When something breaks, the first 20 minutes go to figuring out which tool to even open.

ManageEngine

Common database issues hidden in the dark

Deadlocks & locking conflicts

Unpredictable latency spikes that can freeze entire workflows without warning.

Resource exhaustion

Sudden CPU, memory, or storage pressure that traditional thresholds catch too late.

Unoptimized queries

Slow execution and wasted resources hiding in plain sight.

Schema drift between environments

Silent breakage in automated workflows that surfaces only after deployment.

ManageEngine

Issues that go undetected — until they don't

Threshold alerts catch the obvious. They miss what builds quietly.

What threshold alerts catch

- Hard failures & outright crashes
- Obvious resource spikes past a fixed limit
- Alerts on metrics someone remembered to configure

What slips through

- Gradual query plan & index drift
- Lock contention building over hours, not seconds
- Schema changes with no immediate symptom
- Edge cases nobody set a threshold for

ManageEngine

03

The database observability framework

Monitoring tells you something's wrong. Observability tells you why.

ManageEngine

Monitoring vs. Observability

MONITORING	OBSERVABILITY
Tracks known metrics & fixed thresholds	Connects metrics, logs, queries & traces
Answers “is it up, green or red?”	Answers “why is it slow, and what's next?”
Reactive - alerts after impact	Proactive - surfaces drift before impact
One more alert to triage	One unified, explainable answer
Siloed per tool or database engine	Centralized across the entire DB fleet

ManageEngine

One fleet, One pane of glass

One view. Every engine. Every environment.

- On-prem databases
- Cloud-native DBs
- Multi-cloud infrastructure
- Containers
- Legacy systems

Unified Observability Layer

One dashboard - shared by DBAs, ITOps, SREs & leadership

ManageEngine

The Find → Analyze → Fix approach

A simple framework for turning chaos into a repeatable process.

STEP 1

Find

Surface issues the moment they emerge — across every database in your hybrid fleet, before users ever notice.

STEP 2

Analyze

Correlate metrics, queries, and alerts to pinpoint the true root cause — not just the symptom.

STEP 3

Fix

Resolve with confidence, automate remediation where you can, and stop the issue from coming back.

ManageEngine

Preemptive issue resolution: Before the alert fires

True observability catches three categories of drift before users ever feel them.

QUERY PLAN DRIFT

Without Observability

Optimizer silently swaps to a full table scan. Response time climbs 200ms/day. After 3 weeks — outage.

With Observability

Baseline deviation on I/O wait triggers anomaly. DBA re-runs ANALYZE at week 1. Zero user impact.

LOCK CONTENTION BUILD-UP

Without Observability

Uncommitted transactions stack up over 4 hours. Connection pool exhausted at peak. P1 incident.

With Observability

Latch wait time crosses dynamic threshold. Blocking session flagged at hour 1. Killed before cascade.

PARSE STORM (HARD PARSES)

Without Observability

New microservice deploys unparameterized SQL. Query cache floods. CPU spikes to 95%.

With Observability

Hard-parse ratio anomaly detected within minutes of deploy. Dev team notified before cache saturation.

ManageEngine

04

Turning insights into action

Faster detection, sharper root-cause analysis, and cost-aware performance — all at once.

ManageEngine

Detect faster. Troubleshoot faster.

AI-driven baselines turn raw telemetry into answers.

faster issue diagnosis reported by teams using automated monitoring vs. manual approaches

AI-driven anomaly detection

Dynamic baselines learn what “normal” looks like for your environment — and flag only what's genuinely unusual.

Root cause, not just red alerts

Jump straight from “something's wrong” to the query, lock, or resource behind it.

Source: Forrester Research; New Relic 2024 Observability Forecast (n=1,700)

ManageEngine

Find → Analyze → Fix → Prevent

The framework from Section 03 has one more step. Every incident that gets fixed but not converted into a rule, a baseline, or a check is an incident you'll have again.

FIND
Detect the anomaly

ANALYZE
Pinpoint root cause

FIX
Resolve this instance

PREVENT
Stop the next one before it starts

The next two slides are the 4 signals to catch that make PREVENT actually happen, instead of staying a good intention.

ManageEngine

Consider this scenario ⚽

What database bottlenecks look like if you were broadcasting the World Cup Final:

The re-encoding nightmare

Instead of streaming the match, your system tries to rebuild the video feed from scratch for every single viewer who joins.

The delivery bottleneck

Your broadcast trucks are pushing out a perfect feed, but the local internet is frozen, causing data to back up at your door.

The confused director

Right as a player is about to shoot, the camera suddenly cuts away to a close-up of a fan eating nachos in row 24.

The jammed turnstile

One fan gets stuck at the stadium entrance gate, won't move out of the way, and blocks thousands of people behind them.

ManageEngine

Consider this scenario ⚽

What database bottlenecks look like if you were broadcasting the World Cup Final:

1. The re-encoding nightmare

Instead of streaming the match, your system tries to rebuild the video feed from scratch for every single viewer who joins.

2. The delivery bottleneck

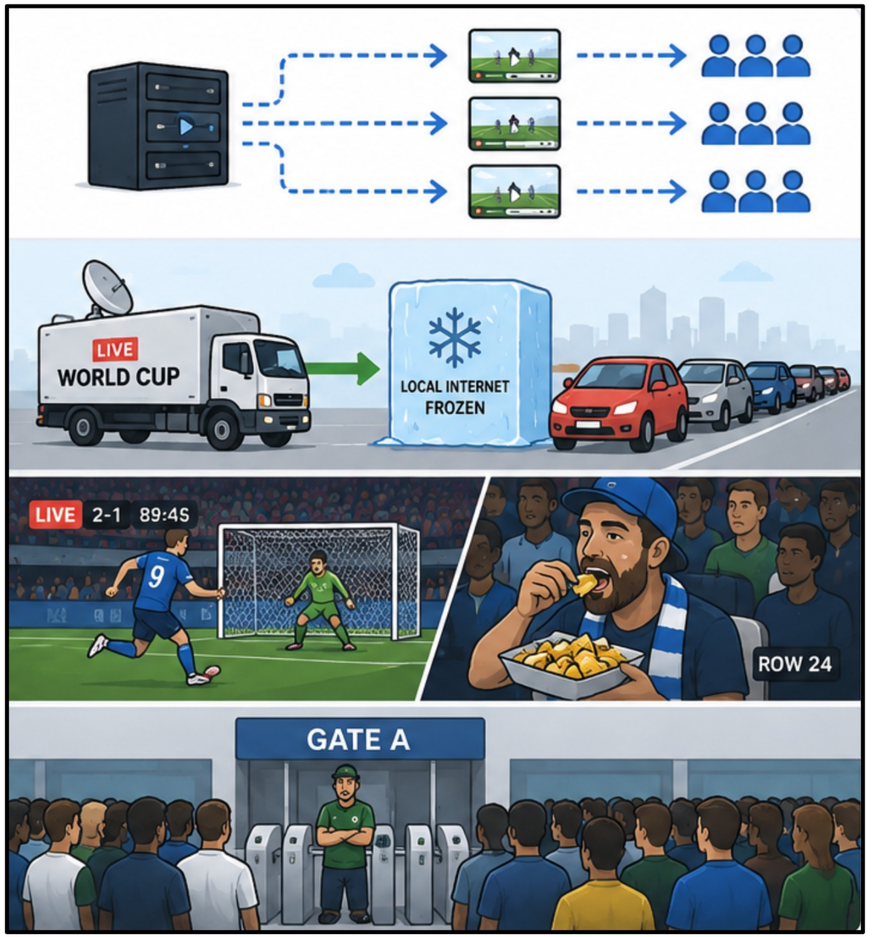
Your broadcast trucks are pushing out a perfect feed, but the local internet is frozen. Data to backs up at your door.

3. The confused director

Right as a player is about to shoot, the camera suddenly cuts away to a close-up of a fan eating nachos in row 24.

4. The jammed turnstile

One fan gets stuck at the stadium entrance gate, won't move out of the way, and blocks thousands of people behind them.



ManageEngine

DBO telemetry pillar: Signal → Triage → Remediation

Four critical signal patterns, what each means, and exactly what to do about it.

TELEMETRY PILLAR	CRITICAL SIGNAL	WHAT IT MEANS — TRIAGE	WHAT TO DO — GAME PLAN
Workload Fingerprint *Throughput*	**Hard Parse Spike + Flat Throughput (TPS)**	A recent deployment has introduced unparameterized queries, flooding the query cache and burning CPU on compilation instead of execution.	***Immediate:*** Rewrite dynamic SQL to use bind variables. Or enable force-parameterization at the DB engine level as a bridge fix.
Workload Fingerprint *Concurrency*	**Active Session Surge + Network I/O Waits**	DB processing fast, but the application cannot consume result sets quickly enough. DB buffers choking. ***Signals:*** ClientWrite / ASYNC_NETWORK_IO waits.	***Immediate:*** Check for N+1 query loop or giant unpaginated payloads. Scale the connection pool or cap max concurrent DB connections temporarily.

ManageEngine

DBO telemetry pillar: Signal → Triage → Remediation

Four critical signal patterns, what each means, and exactly what to do about it.

TELEMETRY PILLAR	CRITICAL SIGNAL	WHAT IT MEANS — TRIAGE	WHAT TO DO — GAME PLAN
The Engine Signals *Active Session History*	**High Disk I/O Wait + Sudden Plan Regression**	Query plan has broken. Optimizer forcing massive table scans instead of index scans. ***Likely cause:*** stale or dropped table statistics.	***Immediate:*** Run ANALYZE / REINDEX on the affected table. Or force a query plan hint to restore the known-good path without a full restart.
The Engine Signals *Wait States*	**High Lock/Latch Wait + Low CPU**	DB bottlenecked on concurrency. Threads idle waiting on resources. Typically caused by uncommitted application transactions or app-layer serialization.	***Immediate:*** Isolate root blocking session via system views. Safely terminate the blocking PID. Enforce stricter connection timeouts in the app config.

ManageEngine

Catching before it becomes an incident

The telemetry on the last two slides isn't just for triage after something breaks. The same signals, if watched continuously, let you fix the problem while it's still invisible to users.

Index bloat, caught at 60% density

A scheduled page check flags leaf-page density drifting below the ~70% baseline on a high-write table(index fragmentation). Automated defragmentation will rebuild in a maintenance window.

Connection creep, caught before the ceiling

Alerting at 70% of max_connections - not 95% - buys time to install a traffic management tool (a pool) *before* the server runs out of room and crashes the entire website.

Stale statistics, refreshed automatically

A scheduled ANALYZE after every bulk load keeps the planner working from current row counts, so a plan regression never gets the chance to happen in the first place.

ManageEngine

The metrics that matter: What to watch and why

Eight signals that tell you everything — before the phone rings.

Query Latency

> 2× baseline

Execution plan regression, lock wait, or resource starvation

Lock Wait Time

> 500ms

Uncommitted transactions or serialization bottleneck

Active Sessions

> 80% pool

Connection pool exhaustion or runaway query loop

Buffer Cache Hit Ratio

< 95%

Memory pressure forcing expensive disk reads

Replication Lag

> 30 seconds

Replica falling behind — read workloads may see stale data

TPS (Throughput)

Flat + high CPU

Parse storms or query plan bloat burning cycles

Hard Parse Ratio

> 10% of parses

Unparameterized SQL flooding the query cache

I/O Wait %

> 20% of DB time

Disk I/O bottleneck — index missing or plan broken

ManageEngine

GAMEPLAN

Alert on impact, not raw metrics

INSTEAD OF STATIC THRESHOLDS

- “CPU exceeded 80%”
- Cache hit ratio dropped
- Connection count spiked
- Replication lag detected

USE CONTEXT AGAINST A BASELINE

- Query latency crossing its own historical baseline
- Cache ratio vs. normal for this workload, not a flat number
- Connections vs. the baseline for this time of day
- Lag outside a scheduled batch-job window

A metric without a baseline is just a number. CPU at 80% can be healthy under a batch job, or a five-alarm fire on a quiet Sunday at 3am — the number alone can't tell you which.

ManageEngine

GAMEPLAN

Habits that make prevention stick

QUARTERLY HEALTH REVIEW

30 minutes, once a quarter

- Review incidents for recurring patterns
- Review slow-query trends for new entrants
- Check peak connection counts against the ceiling
- Test a backup restore, if not done in 90 days

POSTMORTEM, EVERY TIME

Complete within 48 hours

- Incident summary - severity, duration, users affected
- Timeline - first alert through service restored
- Five whys - work back to the systemic cause, not just the symptom
- Action items - with an owner and a due date, not just a note

ManageEngine

FINOPS BONUS

Cost + Performance: Two sides of the same query

Query-level observability doesn't just find slow queries — it finds expensive ones.

Query-level cost attribution

- Map each query to its compute cost — CPU cycles, I/O operations, and memory pressure.
- Identify the top 5 queries responsible for 80% of infrastructure cost.
- Give developers a cost profile alongside their execution plan.
- Surface when a schema change doubles the bill overnight

Idle resource detection

- Flag provisioned compute running at <10% utilization consistently.
- Detect dormant read replicas and standby nodes no query is reaching.
- Identify index bloat — indexes that exist but are never used.
- Right-size recommendations backed by 30-day workload telemetry.

ManageEngine

05

How ManageEngine Helps

A practical, unified path from reactive firefighting to proactive strategy.

ManageEngine

Database observability, built for real DBA workflows

ManageEngine unifies monitoring across your entire database fleet.

Unified Multi-Engine Visibility

Real-time visibility into relational, NoSQL, and cloud-native databases — across on-prem and multi-cloud — in one view.

AI-Driven Anomaly Detection

Dynamic baselines learn normal behavior and flag what's actually unusual, cutting through alert noise.

Root-Cause Pinpointing

Drill into the exact queries, locks, wait events, and replication issues behind a slowdown.

App-to-Database Correlation

Map application transactions straight down to the SQL behind them — ending the dev-vs-DBA finger-pointing.

Automated Remediation

Trigger self-healing workflows — like clearing temp files or restarting services — the moment a problem is detected.

Multi-Channel Alerting

Get actionable alerts via Slack, SMS, or email the instant a real issue emerges.

ManageEngine

One console for your entire database fleet

Oracle (RAC & Multitenant)

SQL Server

MySQL

PostgreSQL

MongoDB

SAP HANA

Cassandra & Couchbase

IBM DB2 & Informix

Redis

Sybase

AWS RDS & Aurora

Azure SQL & Google Cloud SQL

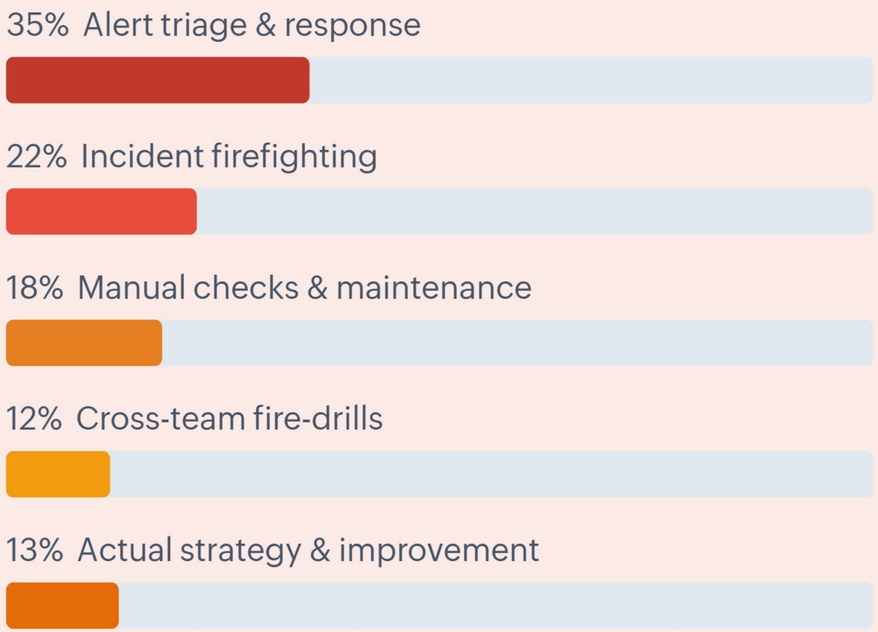
ManageEngine

The transformed week

The same 40 hours — completely redistributed.

BEFORE

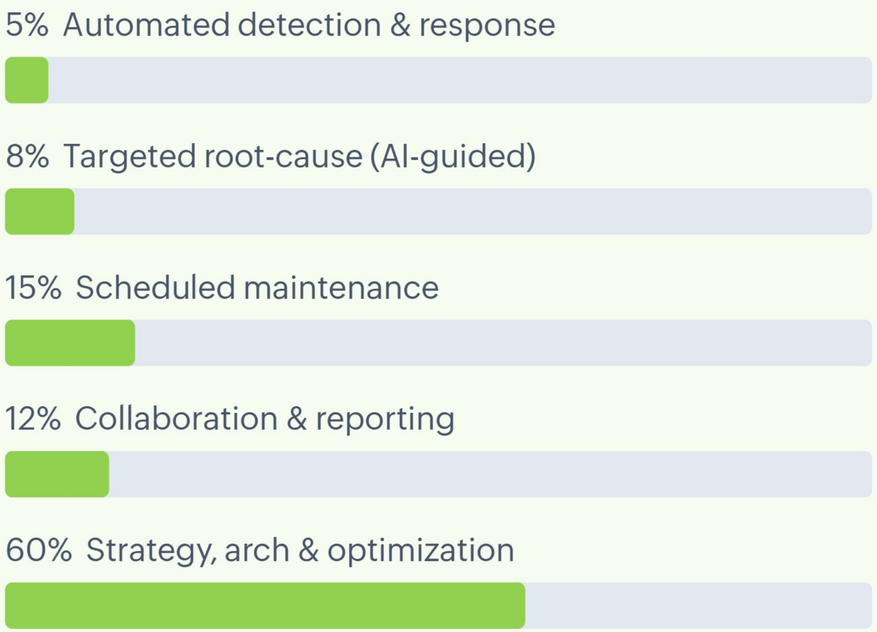
Without Observability



VS

AFTER

With ManageEngine



ManageEngine

Built for teams who'd rather strategize than firefight

50,000+

businesses worldwide rely on ManageEngine for infrastructure, application & database performance monitoring

What's included

- Broad database, cloud, and ERP coverage in one license
- Collaboration-ready: alerts route to the tools your team already uses
- Flexible on-premise or cloud deployment

3 versions to choose from

OpManager Nexus (On-premises)

OpManager Nexus (Cloud)

Applications Manager (On-premises)

ManageEngine

06

Live Demo

See it in action.

ManageEngine

Today's scenario

A production database starts slowing down checkout queries during a traffic spike — the kind of incident that normally kicks off a firefight.

1

Slow query detected

The anomaly is flagged automatically — no manual dashboard-watching required.

2

Root cause isolated

The exact query, session, and resource behind the slowdown, pinpointed.

3

Remedy automated

A guided remediation path — the same logic as the telemetry table earlier.

ManageEngine

RECAP: Key takeaways

Six shifts, one outcome: less firefighting, more strategy.

01

Burnout is structural, not personal

Reactive firefighting consuming 30%+ of the workweek is a visibility and tooling problem .

02

Traditional monitoring can't keep up

Hybrid, multi-database complexity has outgrown point tools.

03

Observability beats monitoring

Find → Analyze → Fix turns chaos into a repeatable process.

04

Insight only matters if it's actionable

Correlation and AI-driven detection close the loop fast.

05

Seeing is believing

A few minutes of demo beats hours of explanation.

06

The right platform changes the math

From reactive firefighting dominating your week to 2 focused hours of strategy — every week.

ManageEngine

Thank you for joining

Here's how to keep moving from firefighting to strategy.

Start a free trial
Download now

Book a 1:1 demo
Send us a mail at
eval-itom@manageengine.com

Implement strategies
using the slides & playbook

OpManager Nexus

Applications Manager