WHITEPAPER

An integrated approach to Enterprise Database Monitoring and Incident Management

A technical whitepaper by **Tony Davis**

Intended audience:
Development Team Leader, DevOps Manager, Operations Manager, Tech Architect

End-to-end
Database DevOps

redgate

Contents

Executive Summary

Whatever the software and hardware architecture of a database application, and however it is developed, it will only continue to function reliably if it is carefully monitored and maintained in production. Regardless of the intrinsic resilience of the application, unpredictable failures and other incidents can and will happen.

To achieve a high level of service, it is important to monitor and measure all the potential points of stress or failure, and to adopt an efficient way of alerting when the metrics suggest that that the service has failed, or is likely to fail, or that its performance has degraded. We then need an alerting and notification system that notifies only the right people of the problem, at the right time, and then gives them access to the relevant information and resources needed to fix it.

This whitepaper describes an optimized approach to Monitoring and Incident Management that will help maintain high service levels in Enterprise Database Systems and promotes a culture of team collaboration and continuous improvement. The proposed strategy uses a **tiered incident management** approach that engages the full range of skills available to the organization in dealing with day-to-day incidents and minimizes disruption to the overall work of the teams. It uses **specialized monitoring tools** to collect comprehensive metric and alert data for each type of service, and then integrates these tools with a **unified notification system** for managing and prioritizing the response.

Any urgent issue that poses an active or impending threat to the availability of the services, or to agreed *levels of service* in the applications and business systems, is immediately 'triaged' by the first line (Tier 1) response team. As required, incidents then escalate to the appropriate operational and technology specialists for diagnosis and analysis (Tier 2), and then to subject-matter experts for incident resolution, prevention and closure (Tier 3).

The whitepaper focuses on the role and requirements of a **specialized database monitoring tool**, such as Redgate Monitor, in this strategy. It discusses the range of monitoring data that it must collect, automatically, across all database server types and hosting platforms, and the intelligent alerting mechanism that will prevent 'flooding'. It discusses the need for an Estate-wide visual **dashboard** that will make all this data accessible to every tier of the incident response, enabling them to link symptoms to cause, and respond efficiently.

It concludes by highlighting the system's potential for further optimization, discussing how to incorporate tracking of Key Performance Indicators (KPIs) alongside database metrics, the need to **left shift** monitoring to includ development and test systems for earlier detection of issues, and for documented response procedures for an effective frontline response.

The requirements of enterprise data system monitoring

As businesses scale out their data systems to support higher data and transaction volumes, most often using cloud-based services, it adds diversity and complexity to the estate. This, in turn, adds complexity to the task of managing and monitoring the overall health of each component of that estate, and understanding exactly where a problem has occurred, why and what action is required.

Organizations need a system of monitoring that detects and reports issues in their data systems in a simple and accessible way; one that that makes it clear where the problem lies, helps identify the cause and enables an efficient team response to any incident. It must help teams deal quickly with active issues that are affecting the service, and then continuously improve them so they remain healthy and optimized over time.

To do this, the monitoring and alerting tool must allow the team to answer the following four questions as quickly and simply as possible.

1. Where is the problem and how is it affecting the service?

When a problem occurs that affects end user experience, or the efficient running of the business, the first role of an efficient enterprise monitoring system is to establish quickly, and unequivocally, which part of the system has caused the issue. Is it a database incident, or is it instead a problem with the analysis service, or a network issue such as a router failure?

The team responsible for maintaining the database system need to be confident that if an issue disrupts the service, the monitoring tool will let them know about it before their users or managers. Forewarned they then use the collected data to understand the nature of the issue, its scope and the extent of its impact. Is it widespread, or isolated a particular database, and perhaps to a certain table or set of tables used by the application?

Finally, they will make a diagnosis, informing the business what action is being taken and when 'normal service' might resume.

2. What is causing the problem?

The database monitoring tool must automatically collect diagnostic data to detect not only obvious problems, but also all the subtler signs of stress and failure in the system. It must **raise alerts** on active issues but also forewarn the team of impending issues, so that they can act before they cause service disruption and downtime, eliminating much of the need for heroics and firefighting.

By automatically storing and analyzing the collected data, a monitoring tool will make it much easier to spot 'what changed' to cause the problem. It can, for example, allow comparison of current behavior with normal usage, via a baseline. Using timelines, visual summaries and snapshots of activity, it can also provide the full context of any database incident. This will include when it was reported, and what processes ran around that time, including any potentially disruptive IT processes such as a deployment, backup job or data movement. This information makes it much simpler to link symptoms to the cause.

3. What action is required?

Having identified an issue that is affecting the service, or will do so soon, the monitoring system must then help any responder to make sense of all the diagnostic data and understand what action is required. Do they need to simply allocate more resource capacity (CPU, Memory, IO, disk space)? Or can they tackle the problem by tuning query processes to make them more efficient, adding indexes, or simply rescheduling resource-expensive processes to run at better times, where they avoid causing contention for users, or disrupting other business operations?

4. Which actions will have the biggest impact?

Increasingly, and especially as infrastructure and business applications move to the cloud, the primary role of a database monitoring tool is not so much in efficient management of the infrastructure, but in optimizing the workload.

The monitoring solution must identify potential bottlenecks in the database system and suggest actions, but also help the team prioritize those actions that will deliver most value to the users and organization. This generally means those actions that will deliver the biggest improvement in the **KPIs of the business applications**. To identify those actions, the database monitoring system must track these KPI-based metrics, alongside all the database diagnostics, to allow the team to *measure* the impact of database system improvements.

As more organizations shift to software as a service (SaaS), identifying high impact actions and improvements will include those that offer the biggest financial savings. Increasingly, organizations will need to left shift monitoring so that when a new feature is introduced, in development, they can *measure* cost of it. If it adds 10% CPU processing overhead, what does that translate to in terms of increased compute costs in the cloud? If you can tune that workload, you can point to significant costs savings, often tens of thousands of dollars a month.

Why do we need a tiered incident management strategy?

You may have constructed a database monitoring system that painstakingly collects and assembles diagnostic data from every available source. However, if that data is impenetrable, or even simply inaccessible, to anyone other than the technology specialists and subject matter experts then it will not result in an efficient response. When the investigation and resolution of day-to-day incidents that affect the business relies on the expertise and 'heroics' of a small number of operational or technology specialists, it slows down the response. It also causes considerable disruption to the strategic and business objectives of those teams.

This monitoring and incident response strategy does not scale to the management of enterprise data systems. So, how do we maintain a high level of service in such systems, while minimizing team disruption?

It requires a more efficient and structured way of dealing with incidents, when the diagnostic metrics suggest that the service has failed, or is likely to fail, or that its security is threatened, or its performance degraded. This whitepaper advocates a **tiered approach to incident management** that engages the full range of skills available to the organization. For example, frontline support staff handle initial identification and triage and deal with routine issues (Tier 1) then escalate incidents to technology specialists for diagnosis and analysis (Tier 2), and then to subject-matter experts for incident resolution, prevention and closure (Tier 3).

This tiered response strategy relies on a monitoring and alerting system that notifies the right people of a problem, at the right time, and provides them access to all the relevant information and resources they need, at each stage of the incident response. It must provide diagnostic information that is comprehensive and detailed, but also consistent across all tiers of the response, and **accessible and valuable** to all the teams involved.

It must allow Tier 1 responders to understand quickly what part of the system needs attention and to perform initial triage. It must allow subsequent tiers to perform increasingly detailed analysis to understand the cause of the problem, identify the required action, and then measure the impact. It must allow all tiers to contribute to a process of continuous optimization.

How do we implement such a strategy?

The goal: enterprise monitoring with optimized incident response

The rest of this whitepaper describes the requirements for a **tiered incident management** approach, and describes a monitoring, alerting and notification strategy that uses **specialized monitoring tools**, for each type of service. These tools filter urgent alerts from any of these systems into a **unified notification system** for managing the response to urgent issues reported for any service. This means any issue that poses an active or impending threat to the availability of the services, or to **agreed levels of service** in the associated applications and business systems. It explains how the metrics, alerts and analysis that a **specialized database monitoring tool** provides can support each stage of a tiered incident response.

An example architecture and workflow

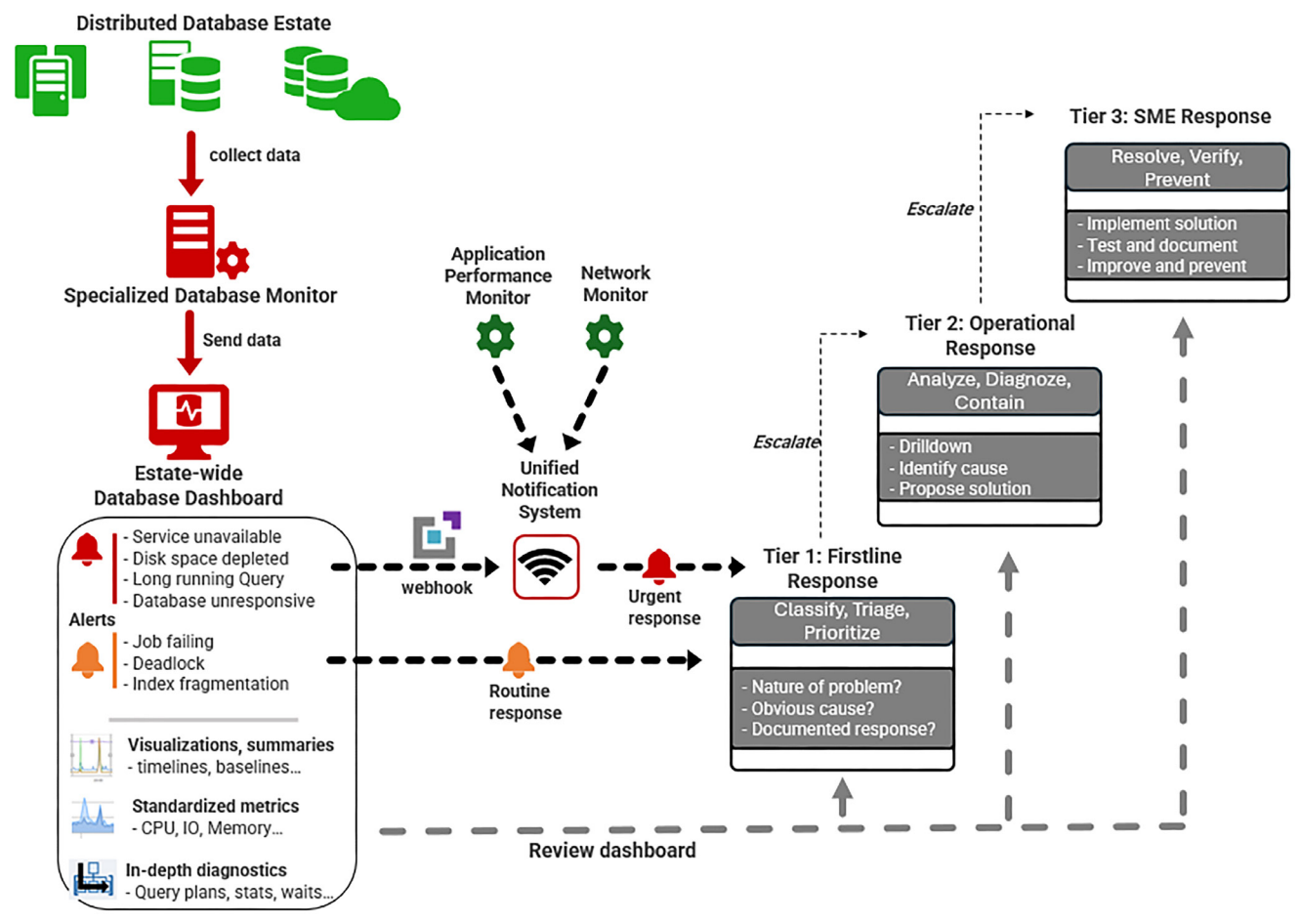
In the example architecture, shown below, a specialized database monitoring system automatically collects metrics for all the potential points of stress or failure in the database systems, whether hosted on local or remote servers or virtual machines, or on various cloud platforms. We enhance this system by adding custom metrics for the **Key Performance Indicators** (KPIs) that matter to the business and to end users.

The specialist tool raises alerts and can send notifications when it detects problems in any of these metrics. For example, via a simple **webhook** integration, it can send urgent alerts notifications to the unified notification system (UNS), for immediate review by the frontline response team. It also gives every tier of the incident response access to comprehensive database metric and alert information, via an **estate-wide database dashboard**, for triage, analysis, diagnosis and resolution.

> **What is the unified notification system?**
> It can range from an incident response and on-call scheduling tool like PagerDuty or Opsgenie to a commercial Incident Management tool like ServiceNow or Jira. Minimally it should provide a central log of all urgent incidents, recording when they occurred, what systems and users were affected, who responded, and so on. It should provide a simple incident escalation process, allowing 'polling' of available or on-call personnel in the appropriate team until someone registers a response. It should provide an easy way to document an incident so that subsequent tiers can quickly review what remediation has been attempted and suggested next steps.

Similarly, the UNS will receive 'threat to service' notifications from the specialist monitors used for other components of the system, such as application monitors or network monitors, and each of these will provide similar specialist dashboards for that service.

This workflow makes it much easier for the "frontline" support team (Tier 1) to understand the nature of the problem. If the database becomes unavailable, is it a problem with the database service itself, or did a network router fail or is it a NAS hardware issue? They will categorize, assess and prioritize each event, based on its nature, severity, and impact and use their judgement as to the appropriate response, for that application. If required, they will **escalate** the alert, or series of alerts, as an **incident**. The incident notification will be sent to the appropriate Tier 2 team for an urgent response.



Alongside this, any database alerts that don't require immediate attention will be dealt with as part of a routine review process by the frontline support team, and on-duty operational staff. They will review the alert section of the specialist tool's monitoring dashboard, dealing with as many of them as possible, such as by using documented response procedures. Any that require action, but they can't resolve, they will escalate, as an incident requiring a routine response.

All response tiers have access to information provided by the monitoring dashboard, as required to fulfil their role in the response. This will include the full details of all the alerts associated with an incident, as well as timelines and graphs to understand its context and nature (Tier 1), resource usage metrics, query and process details that aid diagnosis (Tier 2), and the in-depth diagnostics that lead to resolution, closure and future prevention (Tier 3).

Advantages of the optimized incident response strategy

To instrument a complex database system such as SQL Server, Oracle or PostgreSQL, the most effective approach is to use a tool that is built specifically for this purpose. We then add simple integration and workflow to create an integrated system that can provide effective support for every stage of scalable, tiered incident response.

By having a single, specialized database monitoring tool provide consistent data across response tiers, we streamline the incident response. From the summaries and visualizations used by the frontline response team to the advanced analytics used by operations and other subject matter experts, the shared data ensures clear communication and minimizes confusion among teams. This encourages a culture of cooperation and continuous improvement, which over time will result in enhanced application instrumentation and documented response procedures, for earlier detection of issues and a more efficient frontline response.

By storing and analyzing the collected data, over time, the specialized monitoring tool increases its value and utility at every stage of the response. It can present trends and projections, allowing comparison of current behavior with normal usage, via a baseline. Using timelines, visual summaries and snapshots of activity, it provides the full context of any database incident. This will include when it was reported, and what processes ran around that time, including any potentially disruptive IT processes such as a deployment, backup job or data movement.

Adopt a service-oriented approach to monitoring

In a service-oriented approach to monitoring and maintaining a database-driven application, one gets agreement on what constitutes the service and then implement a strategy that can that maintain that service, accordingly, defining shortfalls in terms of the expected service-level.

To achieve this there must, firstly, be a shared understanding of the importance of an IT system to the organization. Once an application is ready for release, the most important task is to get agreement with the stakeholders about the service level that it requires. This is set out in a documented **service-level agreement** (SLA) that identifies and defines several important targets and objectives that will inform decisions about how the system itself, and the monitoring system for it, is built and maintained.

The **availability and recovery** objectives set out in the SLA inform the architecture plans for the production system and enable the operations and development teams to determine how the application and database will meet them. For example, should the architecture be on-premises or cloud-based? What is the plan for security, backups and disaster recovery? Is there a need for resilience architectures such as high-availability or load balancing?

The **key performance indicators** (KPIs) set out in the SLA will determine the measures and factors that will decide if the service provided by the application is meeting expectations. By measuring these KPIs as metrics, in the monitoring system, it can help you track performance of business processes, alongside the performance of the instances and databases that these processes access so that you can demonstrate improvements.

Another key metric in a typical SLA is the **incident response time**, the time taken to resolve an incident. To meet these targets, the organization that maintains the database will need to plan how to integrate this service into their existing alerting and notification systems. They will also need to understand how to architect the incident response strategy, so it doesn't hit extensive delays due to unavailability of personnel or difficulty collecting or interpreting the diagnostic data.

Automate the task of monitoring the database service

Maintaining a high service level in a production data system requires constant vigilance. If a data system is online but isn't functioning well enough to service business requests, it is of little use to the organization.

There is a lot to check on and the team need to be warned immediately, with an alert, if there are signs of any impending trouble. For example:

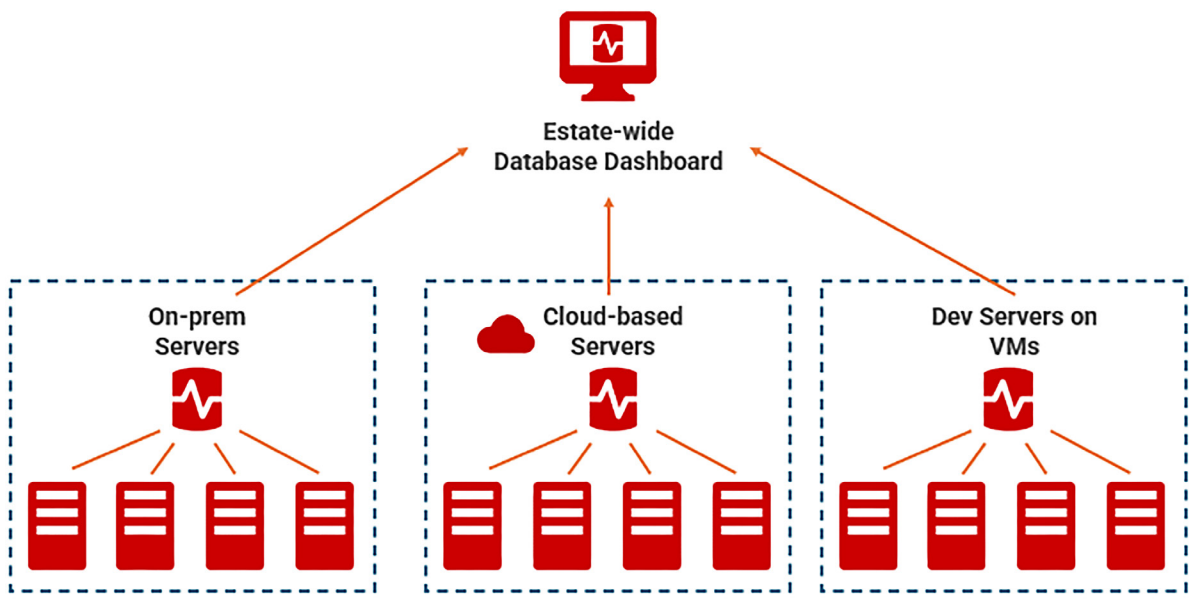
- Is there adequate disk space to allow for data and log growth?
- Are all backups and other scheduled jobs, including import and export of data, running smoothly?
- Is a database service suffering from any stress conditions or points of contention or blocking?
- Are there any alarming errors or warnings in the logs, such as deadlocks?
- Are there any signs of intrusion or unusual user activity patterns? Are all accesses legitimate?
- Are business applications suffering from the effects of slow-running queries or very-frequently used queries?

Automated monitoring

In the 'early days' of supporting a production database, a lot of the required checks were done manually. However, this does not scale if you need to check a large and diverse database estate, often distributed across a mixture of on-premises and Cloud-based platforms. Much of this time-consuming work must now, by necessity, be automated into the database monitoring system.

Your database monitoring might be scripted, or it might use a specialist tool, but it must be automated. It must also be distributed and scalable, so that it can collect comprehensive diagnostics from a 'federated' system and alert the team promptly when it spots signs of potential trouble from any source.

In the following example, we have a specialist database monitoring service automatically collecting monitoring and alert data for a range of database systems, hosted on different local and remote platforms, and making all the collected data available in a single estate-wide web dashboard.



In this way, an automated system of monitoring greatly reduces the time spent by operations teams, DBAs, developers and other SMEs in manually investigating problems and 'fighting fires'. The time saved allows these teams to manage better the many other aspects of their work, from implementing database architectures that support high-availability and resilience, to capacity planning, patch management, and user support.

Automated alerting

The monitoring system will automatically raise alerts when it detects potential incidents, and more generally when associated metrics deviate substantially from their expected value range. The automated alerting system must be highly configurable and use intelligent correlation, filtering and aggregation techniques to avoid "event flooding" (discussed in detail later).

Any alerts it raises should, wherever possible, include advice on the likely cause, and possible courses of action. Sometimes this is simple to do when there is a one-to-one correspondence between symptom and cause. For more complex problems, the automated system will send alert notifications when it detects abnormal patterns of behavior in a metrics, or group of metrics, but usually without any suggestion of the cause. However, by storing and analyzing the monitoring data over time, and documenting known issues according to their observed behavioral 'footprint' and patterns of alerts, we can enable frontline responders to spot even these trickier problems and respond using documented procedures.

A specialized database monitoring system makes this easier, by supplementing the general alerts for availability of a service with alerts for all dynamic parts of the database system. It can add to this a new range of alerts based on deviations from a baseline such as when database suddenly exceeds its normal growth rate.

Implement a tiered incident response strategy

The role of a basic monitoring system is to help detect any '**event**' that indicates a potential or actual problem to a service, and raise a timely '**alert**', which is a record of an event happening, getting worse (escalating), or getting better (deescalating) or ceasing. It can then send a '**notification**' to the appropriate personnel, actively engaging them to use the information in the alerts, and other indicators, to help to resolve the problem, and hopefully avoid it in future.

However, generating alerts and sending notifications is only useful if they enable a response that resolves the problem quickly and improves the service. Many monitoring systems fail to do this because they both distract the team with excessive alerts, and often fail to provide enough detail and context to work out what went wrong. An effective monitoring and alerting system will detect and record all possible causes of stress or failure in a service, but then also allow responders to quickly identify those events and alerts that represent an '**incident**' serious enough to require an urgent response, to avert an active, or a potential, service disruption.

Advantages of the tiered response strategy

When monitoring an Enterprise data system, the task of identifying, analyzing, and resolving incidents cannot fall one person, such as a DBA. It requires a team response, and effective communication and collaboration between the different personnel and roles responsible for each stage of the response. A *tiered incident response* strategy will engage the full range of skills available to the organization and help them avoid a situation where alerts from applications start to cause unnecessary distractions from the core activities of their specialist teams.

When notified of an urgent problem in one of the monitoring services, a first responder must be able to quickly establish the 'big picture' of what happened and decide that it needs to be checked or investigated, and by whom. By ensuring that the right member of staff responds to the incident, depending on where in the system the problem is located, it avoids 'hunting in packs' where responsibilities are unclear, people are distracted from their current tasks, and resources are wasted. Alongside this, any non-urgent alerts can be scanned by the response team and dealt with as time permits, in a process of continuous improvement.

This sort of structured, tiered response strategy provides a workflow system that clarifies who is responsible for dealing with an incident, at each of the stages of its lifecycle. It creates a clear 'audit trail' of who received, responded to, and documented each incident, making it much easier to provide an accurate post-incident analysis or review. The system can be refined and modified where necessary after each incident.

It allows rapid escalation of incidents that threaten the functions of the business systems, and a fast, efficient team response that deals with these incidents and improves the system to prevent recurrence. It also allows first-level responders to filter out routine alerts, and deal with as many as possible. Critical alerts will no longer be missed, or neglected, amid the noise of less important ones and it becomes much easier to ensure that there is always someone available to respond to critical issues, even during off-hours, different shifts, or time zones.

Supporting each stage of the Incident Management Lifecycle

Once a problem has been established with a particular system, such as a database, whether a serious incident or a routine alert, it is then the job of the monitoring and alerting system to allow the first responder to assess what was going on in that system when the incident occurred and then to provide access to the metrics, analysis and details required to support each subsequent stage of the tiered incident response, covering investigation and analysis, resolution, prevention and closure.

There are many ways that an organization can manage an incident, and it is worth spelling out, first, what a typical **incident lifecycle** looks like, as it is the basis for a tiered response for a database-driven application. Most incident lifecycles involve seven distinct phases.

Phase 1. Incident Identification and Logging

The aim is to identify and log incidents as they occur. This may involve alerts from specialized monitoring systems, entries in system logs, user reports, or detection of anomalies through network traffic analysis. All incidents should be logged, centrally, to provide an auditable record of when it occurred, its severity, affected systems, and who responded.

Phase 2. Incident Triage and Prioritization

The incident is notified to the first level, or tier, of the incident response, where the response undertakes triage and prioritization, based on its severity and potential impact. If an immediate resolution isn't possible, the urgency of the incident is assessed and an estimate is made of the number of users affected, and the potential business impact, based on the Service-level agreement.

Where an incident needs an urgent response, the appropriate response team in the second level (Tier 2) is notified for Incident Investigation and Analysis. Any alerts that don't require immediate attention can be dealt with as part of a separate, routine review process by the on-duty operational staff. Any events that require investigation, to prevent them from causing a more serious issue later, can be raised as incidents.

Phase 3. Incident Investigation and Analysis

To determine a root cause, how and why it happened and how to fix it, the incident response team will need to gather additional information, undertake drill-down into the service, and often consult with relevant personnel.

The Tier 2 team will need to decide whether they have the means and guidelines to fix the problem or if it needs to be escalated to Tier 3, where the specialists are called in to assist.

Phase 4. Incident Containment and Resolution

Wherever the incident is dealt with, the next priority task is the 'containment' of the problem by resuming as much of the service as possible. This could involve strategies such as isolating affected systems, implementing workarounds, disabling specific services or even, yes, turning it off and on again.

Once the service is sufficiently revived comes the resolution phase, where the team then works to resolve the underlying issue, which may involve software patches, configuration changes, or hardware repairs.

Phase 5. Incident Communication

Where an incident escalates between tiers, a member of the team will be responsible for keeping the stakeholders informed about the progress of the incident, potential impacts, and expected resolution times. Any critical incidents, especially data breaches, must be brought to the attention of senior management.

Phase 6. Incident Documentation and Post-Incident Review

Once the incident is resolved, the experience gained needs to be made available for the handling of future incidents. A good start can be made by preserving and summarizing the detailed documentation that captured the incident details, actions taken, and resolution steps.

Phase 7. Incident Closure and Prevention

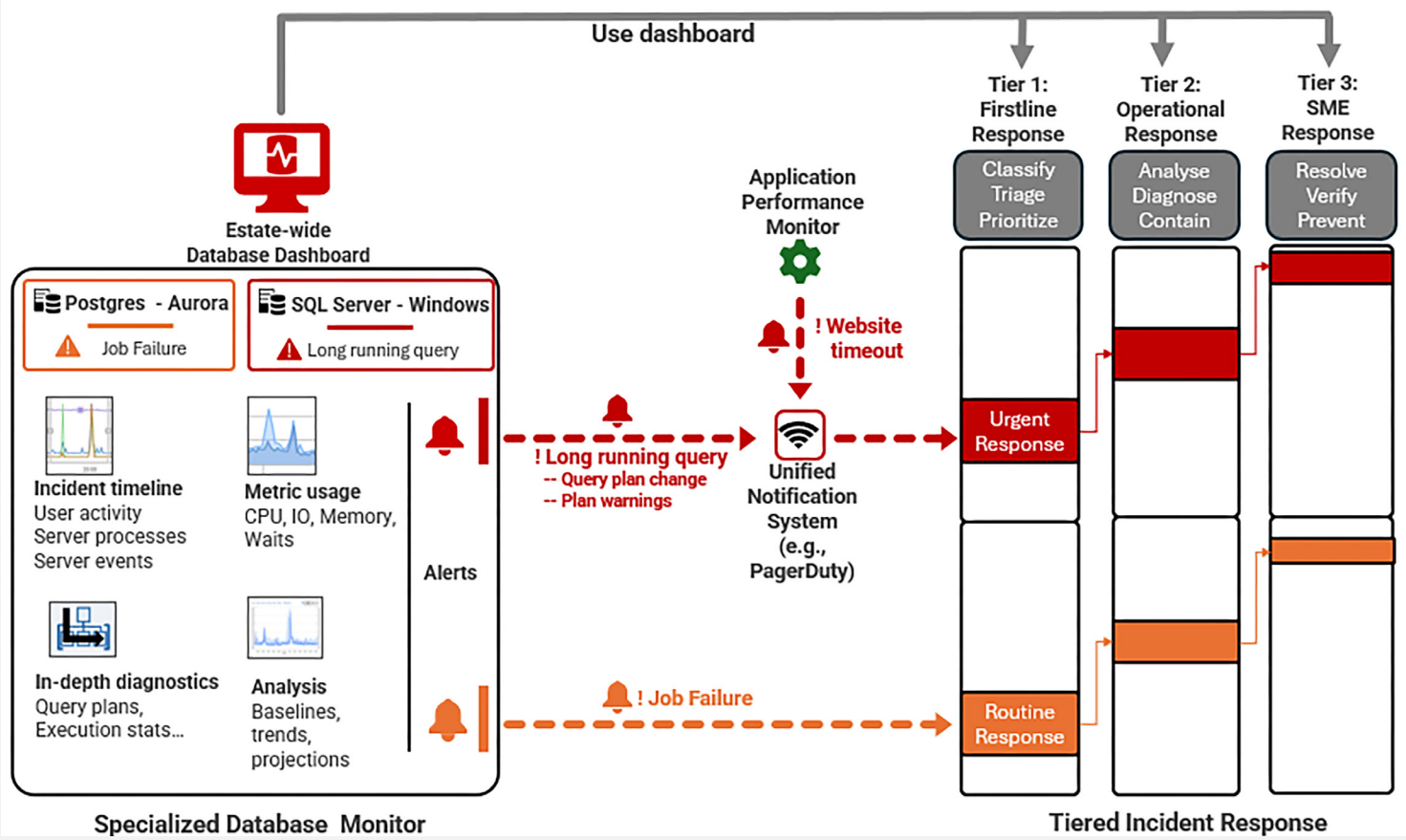
Once the incident is resolved and documented, the incident is closed. The team can conduct a post-incident review to identify lessons learned and opportunities for improving incident prevention and response strategies. For any major incidents, this will also involve management staff.

The role of each tier in an incident response

This section will describe broadly the task and goals at each tier, the information and tools that are required by the response team to fulfil them, and the escalation process between tiers.

The following diagram loosely breaks down the role of each into **Classify, Triage, Prioritize** (Tier 1), **Analyse, Diagnose, Contain** (Tier 2) and **Resolve, Verify, Prevent** (Tier 3). However, the specific tiers and their roles may need to vary depending on the organization's size, structure, and the importance of its database systems.

Whatever the complexity or scale of the system, or nature and severity of the problem, there must be a clear escalation path that ensures that the right people are notified at the appropriate time to address database problems effectively. It might be company policy that certain issues, such as security or major hardware issues, should auto-escalate to the appropriate expert. Conversely, for some routine issues all seven stages of the incident lifecycle might be managed by first line (Tier 1) responders. When an incident is escalated, ownership of the problem passes to the next tier, although collaboration across tiers may continue until the issue is resolved.



Tier 1 – Firstline response: triage

Incident management, triage, service recovery, resolution of routine alerts

The team that handles Tier 1 events are usually responsible for *Incident Management Support*. This involves maintaining the tools, processes, skills and rules for an effective and efficient handling of events, and where necessary escalating them to incidents.

The Tier 1 response will involve staff who will not have in-depth technical expertise but will possess the broader skill of **triage**. As with a medical first-responder, they are the superheroes of resuscitation and appropriate first response.

When notified of an urgent alert, or series of alerts, their first task is *Incident Logging and Categorization*, which is essential to quickly record the nature of the event, the extent and severity of its impact. Their first goal is to locate the likely source of the problem. For example, if Tier 1 receive an alert notification that the process that delivers the end of quarter business review has failed, they will review the notification details, alongside dashboards for the relevant services to establish quickly whether it was a database failure, a problem with the analysis service, or a network problem, such as a router failure.

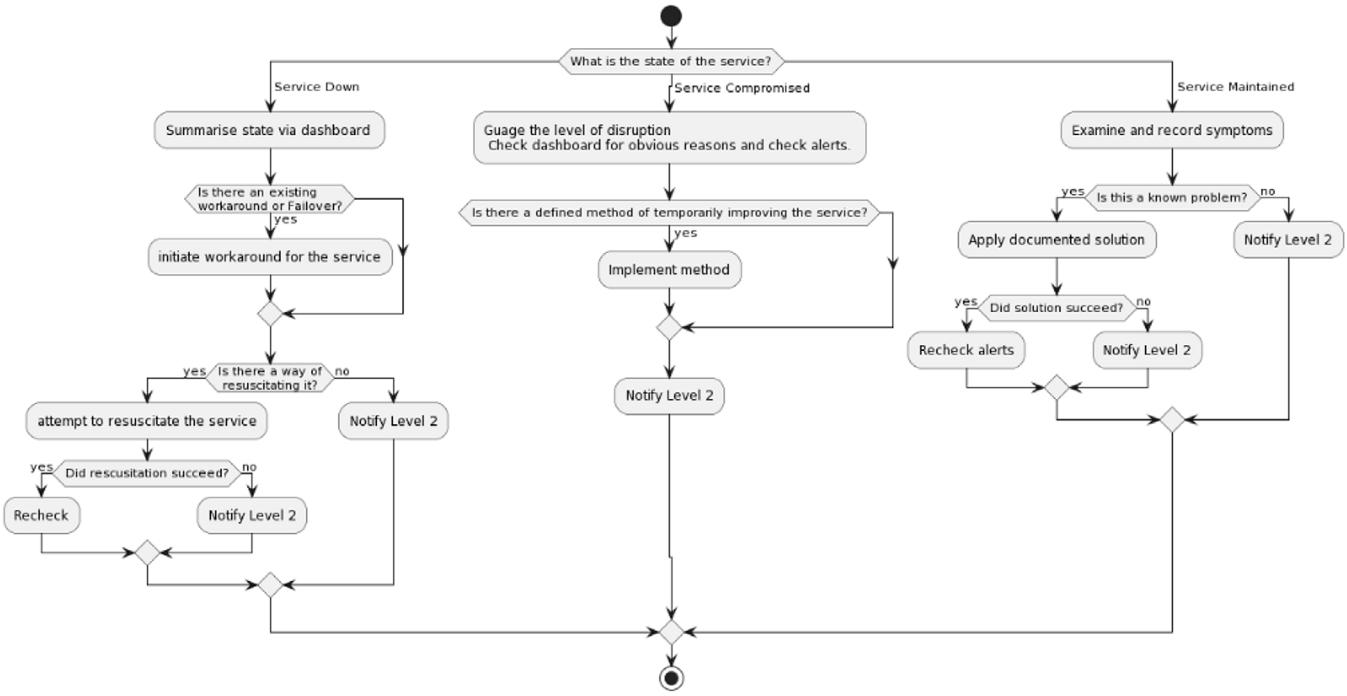
Having narrowed down the issue to a particular service, such as a database, they will perform an initial triage, reviewing the database dashboard and alerts in more detail. They will try to gauge the symptoms of the "patient", their severity and extent and make a quick diagnosis (*does the patient require an ambulance?*).

If the symptoms require immediate action, then "off to the hospital". In other words, Tier 1 will document their findings and escalate the event quickly as an urgent incident, notifying the Tier 2 team qualified to deal with it. If necessary, the teams can agree a policy that certain categories of urgent incident, or any that Tier 1 cannot resolve within a specified timeframe, will auto-escalate to higher levels of expertise or management. Alternatively, if the problem is non-trivial but Tier 1 determines that it is currently affecting only a small set of users, and seems unlikely to deteriorate, then it can be passed onto Tier 2 as a less urgent incident event requires expert help.

In an optimized response strategy, they will have access to "decision trees" that use documented knowledge from previous incidents to help them spot the indicators of known issues, such as a particular pattern of alerts. In these cases, there may be a **documented procedure** they can follow to try to resuscitate "reboot" the patient or alleviate its symptoms with a temporary fix. If they succeed in restoring the patient to a stable condition, the "trip to the hospital" may still be required, but perhaps without the screaming sirens and flashing lights. They will escalate the incident to Tier 2, with details of the 'closed' alert, for further investigation and a permanent solution.

Alongside urgent incidents, first responders will, as part of their routine checks, review all other non-critical alerts in the monitoring system dashboards, and categorize any that require an incident response. Categorization is done at this point so that any trivial alerts turn can be handled easily at this stage, without any escalation. An alert relating to an isolated spike in database resource usage (CPU, IO, Memory), for example, should not, indeed must not, become an incident, unless it threatens the functioning of business systems. The first responder will often follow instructions to confirm a diagnosis, which will require them to check the context of the alert that triggered the notification, to verify its nature. Was the alert followed or preceded by a particular event, such as a ETL job or deployment, for example?

In a well-managed incident response, the Tier 1 team will be able to respond to many of these non-critical issues, by following a set of instructions, or *playbook*. This allows a process of continuous improvement that over time reduces the number of alerts and log entries that report minor issues, by fixing them and preventing recurrence. If the alert is successfully dealt with, the team will deal with the incident documentation and reporting. Otherwise, it can be escalated as an incident that requires a routine response.



Tier 2 – Operational response: diagnosis

Incident assessment, diagnosis, containment, resolution, reporting

If Tier 1 cannot quickly resolve the problem that caused the alert, or if it turns out to be a severe incident, then it will be escalated to Tier 2 via a notification. At this point the 'patient' is metaphorically at the hospital, or at least the doctor's office.

The Tier 2 response will usually involve on-site or on-call Technical Support or Operations Support staff personnel. They will have broad technical knowledge of the affected system and will be able to investigate the issue and determine the root cause, and either administer a treatment, or propose one that will require the assistance of an expert 'surgeon' (Tier 3).

To establish how and why the incident happened and how it can be fixed, they will need to gather additional information, undertake drill-down into the service, and often consult with relevant personnel. In addition to seeing the current alerts, the Tier 2 response team will need access to resource usage metrics describing the health, performance, and security of the affected service, and the underlying details of any user processes and requests (queries) involved.

For example, perhaps Tier 1 escalates an incident where the end of quarter business review failed and note that it coincided with a high severity alert from the database monitoring system that a disk drive ran out of space. Tier 2 investigates why the process failed and determines that the lack of available disk space caused the "file copy" part of the process to error out. They will give the business an estimate for when the reports will be delivered, based on how much time it's likely to take to recoup the space and restart the process. They may also need to escalate the incident to Tier 3 for further investigation into what caused the disk drive to fill unexpectedly.

Where an incident is successfully resolved within Tier 2, they will **document** which alerts led to the notification, what the symptoms were as seen from the monitoring tools, and how the incident was resolved. This record will then be available to Tier 1 for any subsequent repeats of the incident.

If the Tier 2 response team cannot resolve the problem, they will document been ascertained regarding the nature of the problem, any attempts at resolution or intermediate measures put in place and proposed next steps. This may include automatically creating a work ticket for any development or testing work required to verify a solution. All this information is useful for the postmortem analysis ('*what have we learned from this?*') and can help build out the range and effectiveness of any documented response procedures.

Tier 3 – Specialist response: resolution, prevention, continuous improvement

Incident analysis, resolution, prevention, system improvement

The patient is now "*under the surgeon's knife*", receiving the expert care of the specialist in the component of the system that has caused the problem. Depending on the issue reported, this might be the network specialist, the programmer who wrote the application code, the security advisor, the database administrator or some other subject matter expert.

In response to a notification from Tier 2, the appropriate Tier 3 technology specialists or subject-matter expert will review the diagnosis, actions and recommendations of Tier 2 support and perform a thorough investigation. They will need access to the full range of information available in the monitoring system dashboard, including metric, alert, trace and log data. They will review this alongside baselines, diagnostics, reporting and analysis to confirm the underlying cause and diagnosis, using additional specialized tools to gather further evidence, as required. They will then administer the necessary treatment to resolve the problem and run follow-up tests to confirm that it worked.

If an issue is sufficiently serious, especially if compromises the service-level agreement, the managers, directors, or senior leadership will need to be informed by the Tier 3 team so that they can take strategic decisions.

Tier 3 will also be involved in any required system 'rehabilitation', suggesting improvements to ensure full recovery and prevent recurrence. They will contribute to post-incident reviews and make recommendations for improving incident response processes. This will include making improvements to the monitoring system, notification system, and database application. For example:

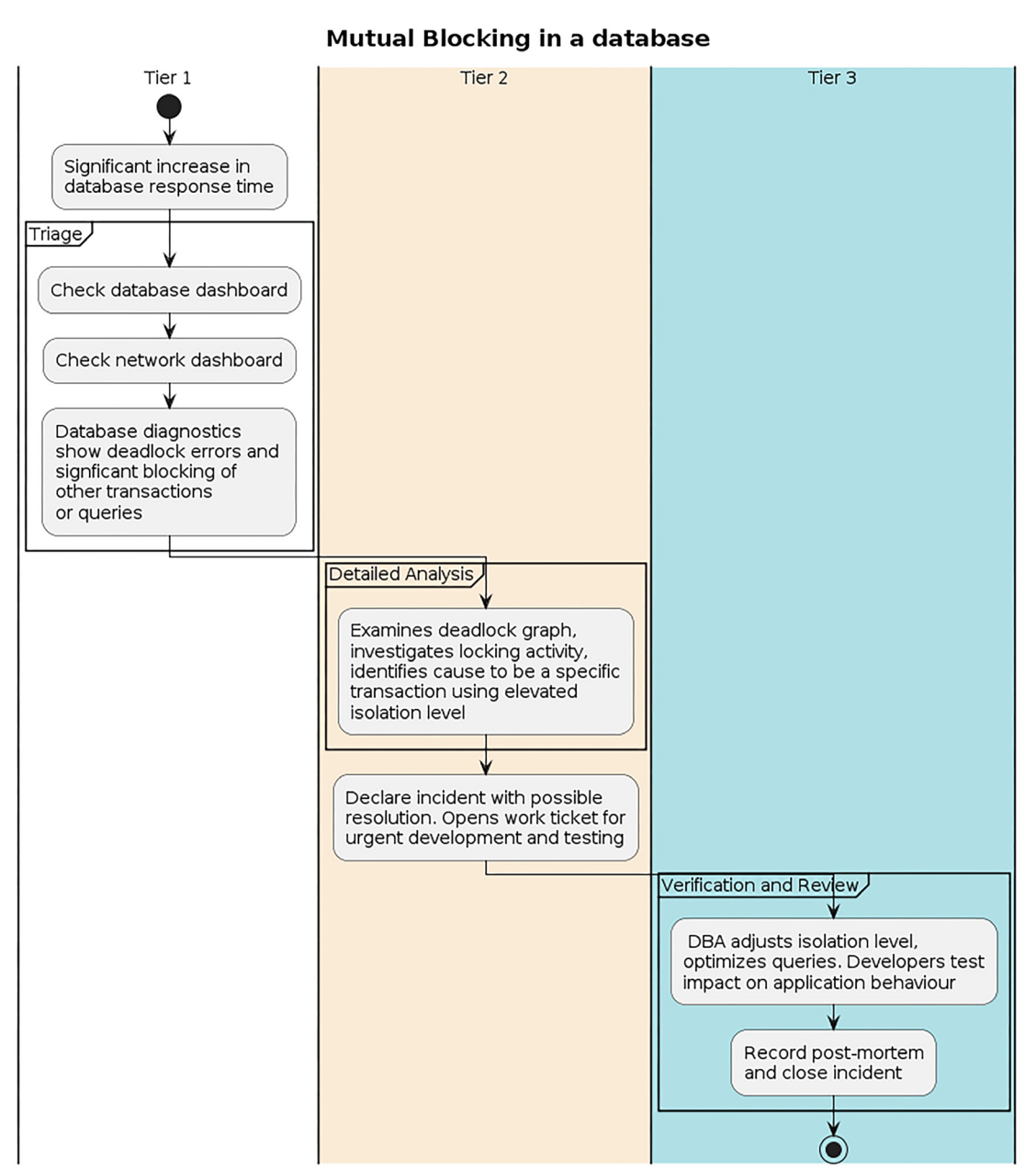
- Introduce new tests to detect known causes of service deterioration
- Improve alert documentation, threshold settings and response guidelines
- Improve pre-allocation of resources based on trends, projections and predictive analytics
- Introduce new custom alerts to detect more quickly an incident that affects business systems and end users.
- Make recommendations to managers and stakeholders for systems improvement, and training requirements.

A tiered incident response example

The stages in the lifecycle at which an incident gets escalated, and to whom, will vary between organizations depending on their activities, their size, and the technologies they use. Within an organisation, it will vary according to the nature of the problem detected and on the specialized monitoring tools and personnel available to support the work at each tier and perform diagnosis. The following workflow offers just one example of how the tiered incident response might work. It indicates the sort of information each tier of the response needs to do their job and illustrates how a tiered approach makes full use of the skills of a wide range of personnel with the various teams, rather than always relying on the availability and expertise of a few.

The frontline response team (Tier 1) receive an urgent notification that application response times have increased dramatically. They perform initial assessment and triage using available dashboards for each service, and establish database performance degradation, caused by a deadlock error and associated blocking, as the likely cause of the problem. They document their findings and escalate the incident to Tier 2, in this case to the Operations team.

The Tier 2 responders review the full details of the database alerts, and the processes and queries involved in the deadlock. Their investigation reveals that one of the processes is using a highly restrictive transaction isolation (`SERIALIZABLE`). They perform initial tests that reveal that use of `SNAPSHOT` isolation could resolve the problem. They document their findings and escalate the incident to Tier 3 for verification and testing. The DBA optimizes the queries, and the developers test for any unintended consequences for application behavior or response. The solution is documented, and the incident closed.



How to collect the required monitoring data

As the preceding discussion makes clear, an effective tiered response relies on a monitoring system that supplies a diverse range of information to support the work of the people who respond to an alert or incident, at each stage.

It starts with the timely detection of threats to the availability or service level of the business systems, services and processes. This information is essential for Tier 1 of the response since it will identify the location and nature of problem. It is a waste of effort to dive straight into the detailed diagnostics for the database system if the problem is elsewhere in the network. After that, the subsequent tiers of the response need detailed diagnostics and alerts, per type of service. For the database monitoring system, it means provision of standardized diagnostics metrics across all supported database systems and hosting platforms, to in-depth diagnostics per service, for incident investigation and resolution.

So, how do we collect, store and analyze all this monitoring data? This section describes the three most common approaches: scripting, general purpose morning platform, and specialized monitoring tool. It describes their advantages and drawbacks and why we advocate the use of specialized monitoring tools, per type of service, as the best approach for enterprise system monitoring.

Whichever approach you take, it makes sense to also monitor the database with an external process. Although the database system itself can, depending on the database management system being used, report many types of problem without an external database monitoring tool, there are plenty of occasions where that isn't possible. After all, unconscious patients cannot describe their symptoms or notify the medical staff. It is also easier for an external process to maintain baselines and historical data about usage. Also, for compliance with security regulations and standards, any monitoring system for database security issues must be a separate process from the database itself. Monitoring tools that are external to the database are difficult for an intruder to neutralize.

Scripting

Database management systems have evolved to provide much of the required 'stress' information and many monitoring systems consist of custom scripts that extract these metrics from the various system objects, dynamic metadata views, performance counters and event tracing systems that the RDBMS provides. The scripts can be automated, capturing and saving their output at scheduled intervals, and issuing alerts when the metric values suggest a potential problem. With an ETL tool you can even build your own "monitoring data warehouse" to track, compare and analyze the activity and events on the databases and servers, over time.

The result is a specialized database monitoring system, built for purpose and customizable to provide exactly the data you need for maintaining your database system. However, this can quickly become a very complex and expensive approach:

- **Time-consuming maintenance** – it quickly becomes a very time-consuming task to keep the monitoring system up-to-date with all new editions and features, as they go into production use
- **Scalability concerns** – similarly, extending the scripted approach to cover new server, platforms and database types, and then integrating all the data, is a big job.
- **Alert flooding** – simple threshold-based alerting can get very noisy. Intelligent alerting techniques that help, such as aggregation and alerting only on significant deviations from 'normal', are very hard to get right in a 'DIY' approach. You cannot afford to mess up and lose alert history.
- **Low 'usability' of data** – it is hard, distracting, work to not only collect all the information but then also manually decode, assemble and present it in a form that others, such as Tier 1 responders, can use to spot problems. Most scripted solutions don't achieve this and so the incident response relies largely on the relatively few individuals with the domain knowledge and expertise to interpret the data.

Custom scripts and tools that allow a DBA to extract the detailed information they need to do live troubleshooting are a valuable *supplement* to any database monitoring solution. However, this approach on its own does not scale to support the requirements of a tiered monitoring strategy for enterprise data systems.

General-purpose monitoring platforms

In this 'top down' strategy, an operations team will use a general-purpose infrastructure monitoring tool and application monitoring tool such as Datadog, SolarWinds, New Relic or Dynatrace. This tool will provide broad observability over a range of servers and services. applications and other components on the network and will raise alerts when it detects a problem with one of them.

These systems are often provided as a SaaS platform, with integrations, or plugins, for monitoring each type of specialized service or component. For each monitored service, such as a particular variety of RDBMS, you typically install a lightweight 'agent' on each instance, and it automatically collects and stores data on query performance, server health, resource usage and so on. If the database usage is monitored within the application's context, you may need to install an application and a database agent.

There will usually be a dashboard for each system for review and analysis of the collected data. This means you get to see metrics and alerts for your database system, for example, right alongside those for other system components and applications. This is helpful for the incident identification and 'triage' requirements of the Tier 1 response. For example, the team might quickly correlate an application performance alert with a high severity database alert for 'blocking' and escalate the alert as an incident to the appropriate team.

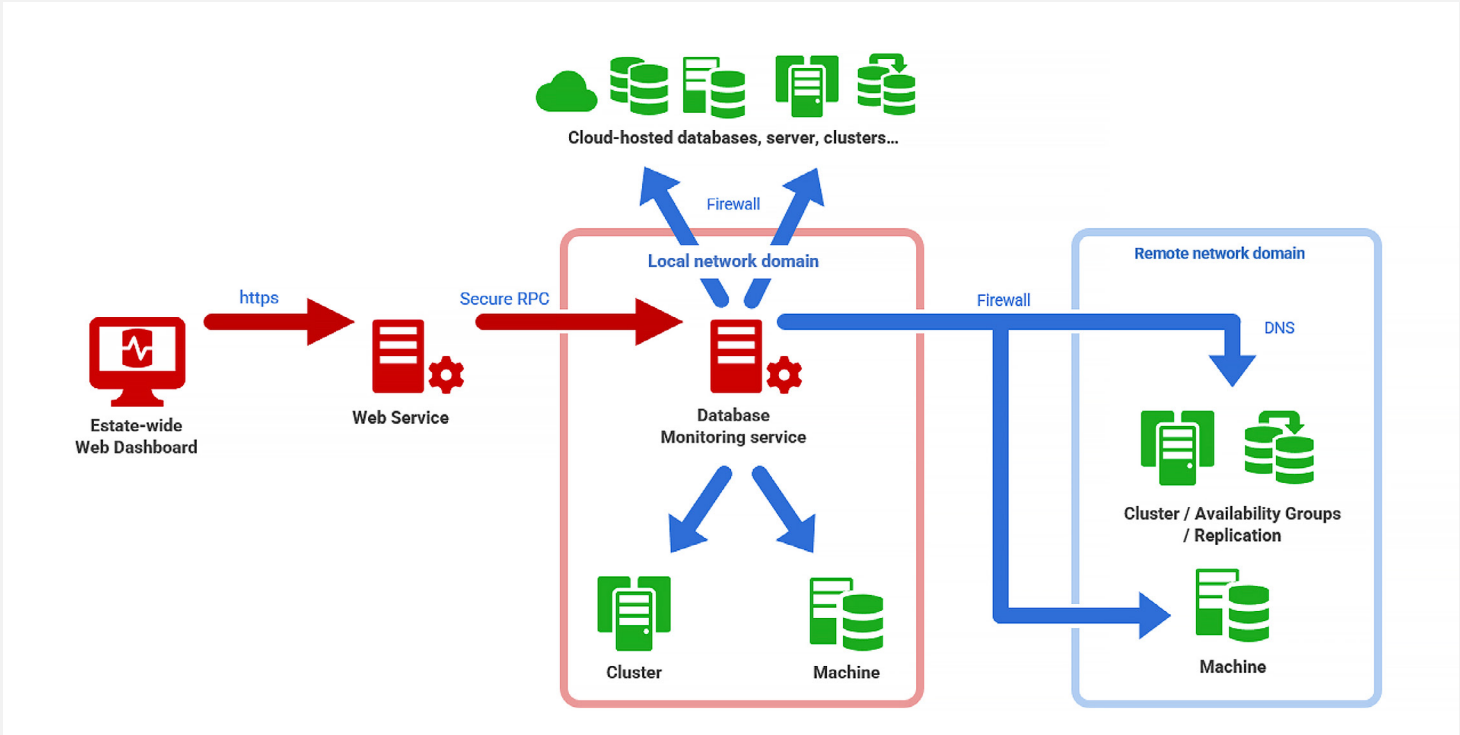
However, its broader success in a scalable, tiered incident response hinges on how well the general-purpose platform provides the in-depth diagnostic information needed by subsequent tiers to resolve the more difficult database contention issues. How well does it support all the RDBMS-specific extensions? Does it account for all inconsistencies in metadata queries across database systems, variations in how each reports its dynamic running state and 'improvisation' in the support for procedural SQL by each vendor?

Often the quality of support across database systems is inconsistent, and gaps appear in the diagnostic data and alert details. The incident response can be delayed if the general-purpose platform supplies no metrics or baselines for the underlying cause. For example, let's say a SQL Server database is experiencing serious connection and blocking, and its performance has deteriorated to the point where an incident is escalated to Tier 2, via notification. The team suspect `tempDB` contention but the general-purpose tool currently provides no in-depth `tempDB` metrics to allow Tier 3 to confirm the cause and examine why this happened. They will need to use other tools (such as custom scripts) to investigate the cause and propose a solution, but of course there will be no baselines for these metrics. This also undermines the tiered response strategy as it removes the consistency in the data used across each tier and makes collaboration and communication more difficult.

When further information is needed, consistently, it usually means writing and maintaining custom integrations, or paying for additional services, to cover the sources of stress and contention that the plugin does not provide. The more "top down" customization effort it requires, the more this approach runs into the same maintenance and scalability issues identified for the scripted approach.

Specialized monitoring tools

It is the job of the specialized database monitoring tool, such as Redgate Monitor in the example below, to provide a comprehensive set of built-in metrics and alerts and collect this data automatically across all server types and hosting platforms, regardless of where they are physically located. If the tool supports more than one type of database system, such as a range of RDBMSs, then the monitoring service for each RDBMS type should be specifically designed for that type of database engine, providing diagnostic data for every source of potential failure. It should incorporate intelligent alerting and report the diagnostic data in digestible form, for each tier of the incident response, ideally in a single estate-wide dashboard.



By using a specialized tool with predefined metrics, tailored to the unique characteristics and performance patterns of databases, plus configurable alerting, you'll eliminate the issues of scalability and maintenance that limit the scripting approach. You can establish baselines for all metrics needed for accurate detection and diagnosis of faults or anomalies, across multiple machines and network domains, and all this data will be available to every tier of the incident response. You'll also avoid the situation where a general-purpose tool raises an alert, but custom tools or scripts used subsequently by the DBA to investigate cannot detect "what changed" because there are no baselines for those metrics.

The historical problem with using a specialized monitoring tool for each type of component is that they were rarely are they used in any sort of coordinated way across teams or departments, and so did not work effectively together in a broader monitoring, alerting and incident response strategy. The common difficulty was in providing an effective way of having a specialized tool feed selected alerts into an IT department's existing notification system, to allow a frontline response team to make an initial assessment of the nature and scale of the problem. Detailed database metrics and baselines are of little value until you can confirm that the database is the source of the problem. However, this is now much simpler: we just use a simple integration such as a **webhook** to connect each monitoring tool to a centralized notification system or "incident management" platform. This allows a first responder to review notifications for service or application disruptions through a unified interface, regardless of the source of the alert, and then assess the situation and take appropriate actions. For an example of how this works in Redgate Monitor, see **Integrating Redgate Monitor into a Tier-1 Alert and Notification System**.

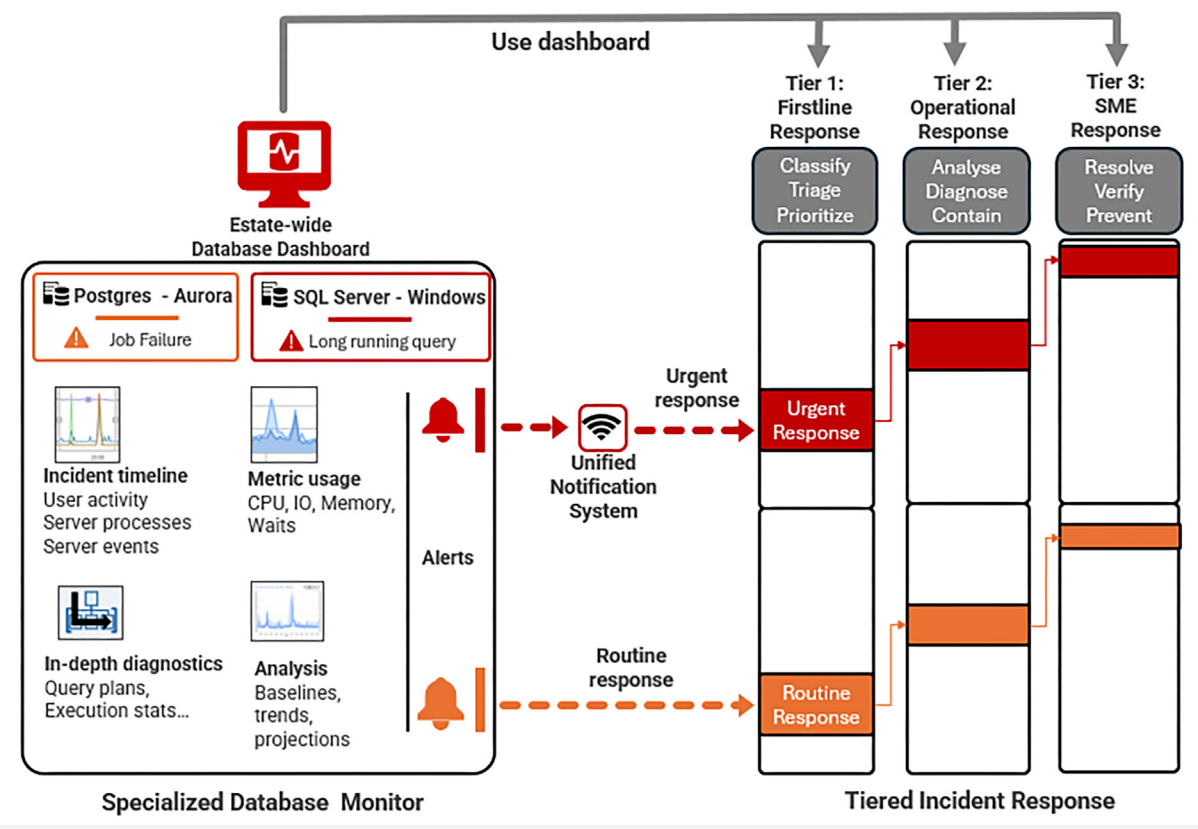
You can optimize the response further by adding to the specialized database monitoring tool a relatively small number of custom metrics that measure and establish baselines for some of the KPIs of the business applications (such as the time taken to checkout, or to generate an invoice). The team can then view those metrics right alongside the built-in diabase diagnostics and server activity graphs. It makes it much simpler to spot cause and effect, and so allows the team to respond promptly to any issues that are demonstrably affecting the running of the business and make process improvements to avoid recurrence.

It's simpler to "expand up" to capture and maintain a small number of universal measures of business process performance, than it is to have to "drill down" and fill in extensive gaps in the detailed database diagnostics.

How a specialized database monitoring tool supports scalable incident response

As discussed previously, the strategy advocated in this whitepaper employs a range of specialized monitoring tools, one for each type of major component or service in the IT system, and so might include the coordinated use of metrics, alerts and notifications from network, server, application, and database monitoring tools. Each specialist tool provides in-depth diagnostics for that service, via a dedicated dashboard. By integrating with a unified notification or incident management system, each specialist tool participates in a broader monitoring and alerting strategy that alerts teams immediately to any urgent issue that threatens the operations of the business, or the quality of service provided to end users.

The goal is an integrated solution that provides both monitoring and problem resolution functionalities for your data systems and will allow smooth transitions from identifying issues to resolving them, as an incident passes up the response tiers.



The following sections review the basic information required from a database monitoring and alerting system that will allow the response team at each tier to fulfill their various roles. It will consider the alert filtering and control mechanisms that are required to prevent alert flooding, and to ensure that important alerts reach the right people, promptly, and provide sufficient context for them to be able to diagnose and act on them, swiftly.

After that, we'll examine the requirement for a **global dashboard** that presents analysis, summaries and timelines for all of this collected metric and alert data, in order that any responder can make sense of it and knows how to act.

Immediate notification of service or process disruption

A problem that affects a customer-facing service for a large proportion of end users, or a fault with a business process that impairs the company's ability to trade, or an issue that threatens data security, is an 'emergency' that needs an urgent response. If a database becomes unavailable, a purchase process isn't responding correctly, or a search facility stops working, users will notice very quickly. The Tier 1 responders will need to establish very quickly if this was caused by a failure in the database system itself, or a problem with a cloud service or a local network fault or some other event that isn't directly related to the database.

The database monitoring tool, and other monitors in the system, must be configured and automated to send alerts for any events that threaten the operation of the business systems and services to the unified notification system, so the Tier 1 team are immediately alerted. For the databases, this is likely to include immediate notification of all issues affecting **connectivity** (can the database be reached?), **availability** (is the database responding?), **response time** (has the database service slowed significantly?) and for any **severe errors or warnings** that are logged by the database.

The Tier 1 team also needs immediate notification if any facilities or processes critical to the running of **business operations** or to **end user activities** are not functioning correctly. If an end-to-end process, such as a purchase process, has failed somewhere between service and user then the response team can immediately check the state of individual services for a cause.

To provide simple 'service availability' alerts, we can add simple **custom metrics** to the database monitoring system that check that individual services are reachable and respond to a simple query. To extend these checks to cover **end-to-end business processes**, such as the purchase process, the queries can return functional metrics that are unique to each database process, such as the number of 'shopping-baskets' checkouts in the last hour, the number of searches, and so on. These scripted checks can be scheduled to run from each section of the network. The resulting notifications will inform the team of the nature, severity, and extent of the problem. Are all users affected, for example, or those on a particular part of the network?

In an optimized system, these custom process checks can incorporate measures of the KPIs for the business applications (see later for more details).

Standardized sets of metrics and alerts per service

The monitoring systems will need to provide a standardized set of metrics and alerts sufficient to allow the early detection of issues and anomalies in the database environment. It will need to provide all this information, regardless of hosting platform (local or remote server or VM, any type of cloud service), and for the whole range of database systems, which may include several flavors of RDBMS, document databases, NOSQL databases and so on.

By detecting performance bottlenecks, database errors, design faults, configuration changes, resource constraints or security threats as soon as possible, the operations team can then take immediate measures to prevent them causing downtime or data loss. They'll need to be able to investigate, for example:

- **User activity** – for example, drops in transaction rate (throughput), increases in number of connections, details of long-running user requests (queries), and details of processes and queries involved in any significant blocking.
- **Resource usage** – is the server CPU limited or suffering memory starvation? Is there a disk IO bottleneck? Is disk space being eaten up and why?
- **Server processes**– statistics and details for all server jobs (esp. backup) and notifications if jobs fail, or fail to run, or run long.
- **Server properties, configuration** – do instance and database-level settings adhere to company standards/defaults? If someone changes a server memory setting or disables the automatic update of database statistics, an alert should be raised.
- **Errors** –any serious errors (like a deadlock) warnings or timeouts encountered by the database.
- **Security** – suspicious activity including failed logins, changes to security settings, unauthorized access attempts, evidence of injection attacks etc.

Of course, it is as important that the monitoring system allows the team to respond quickly and effectively to a notification of a database incident, as it is to generate alerts. When alerts are raised that warn of any unfavorable conditions in these metrics, or any logged errors, it ought to provide enough information for an initial investigation, sufficient to allow the appropriate response tier (often Tier 2) to quickly diagnose the problem.

By continuously recording query execution times, I/O throughput, CPU and memory resource utilization, disk space usage, and so on, as the user and process workload on the database fluctuates, the team can see the context in which the incident occurred. Analysis of cause-and-effect becomes much simpler. We also remove the guesswork from database configuration management, query optimisation and index design. This will lead to better overall performance and responsiveness of the database, within the current hardware.

It will also allow the team not only to respond quickly to active incidents but also to spot worrying trends in resource use or activity that need action to prevent an incident. If we know the current trends in resource utilization, including CPU, memory, and disk space, we can more accurately predict future growth, improve resource allocation decisions, and so avoid performance degradation or disruptions from high usage. It also leads to informed decisions about database maintenance and resource upgrades.

In-depth diagnostics

The monitoring system must provide in-depth metrics, alerts and activity snapshots that will allow the appropriate responder to spot an error, security incident or server stress condition, quickly and link it to its cause. It must then enable any responder to determine, as quickly as possible, the appropriate action, whether it's to tune a query, reschedule a server job, or allocate more server resources.

Developers and operations staff will need access to the full range of *performance diagnostics* that each monitored service can provide to investigate performance and contention issues. Security experts need information from security logs, event tracking systems and audit logs to allow investigation of any threat to data or data system security. Operations staff will need access to diagnostics to help them deal with urgent issues regarding backup and recovery, resilience, high availability, clustering, replication, and failover systems, or doing capacity planning.

It's impossible to be exhaustive, but for the database monitoring system, this will include:

- **Detailed performance metrics** – access to the full range of diagnostics available in the underlying system objects, dynamic metadata views, performance counters and event tracing systems. For example:
    - **Query execution statistics** – snapshot of expensive queries running around time of incident, filterable by execution time CPU usage, memory usage, IO usage.
    - **Query and query plan details** – SQL text of expensive queries plus execution plans, details of plan changes to spot plan caching issues, plan reuse issues, excessive recompilation, cardinality mis-estimations etc.
    - **Index usage statistics** – for index design issues or missing indexes
    - **File system diagnostics** – I/O statistics broken down by database file to locate contention hotspots on the disk subsystem
    - **Detailed memory diagnostics**– to detect memory starvation, plan cache churn etc.
- **Database security metrics** –to detect database privilege escalation, failed login attempts, unauthorized configuration changes, including those that expand the attack surface area, other suspicious patterns of user activity such as injection attacks, unauthorized schema changes, and so on.
- **Details of servers, storage, backups, jobs** – estate-wide access to details and statistics reading backups and other important maintenance and ETL jobs. Tracking, projections and alerts for free disk space, unallocated space in database files, unexpected database file growth and so on. Tracking of changes to server, instance and database configuration settings across all servers.

For any of these database problems, the Tier 2 and 3 response specialists will need to perform in-depth analysis of the full range of database metrics, alerts, baselines, diagnostics, reporting and analysis so that they can quickly become confident that they know what has gone wrong and why, and how it might be fixed. They will often need to find out enough about the problem that they can reproduce it on a test system, configured to match the production system as closely as possible, and verify any proposed fixes.

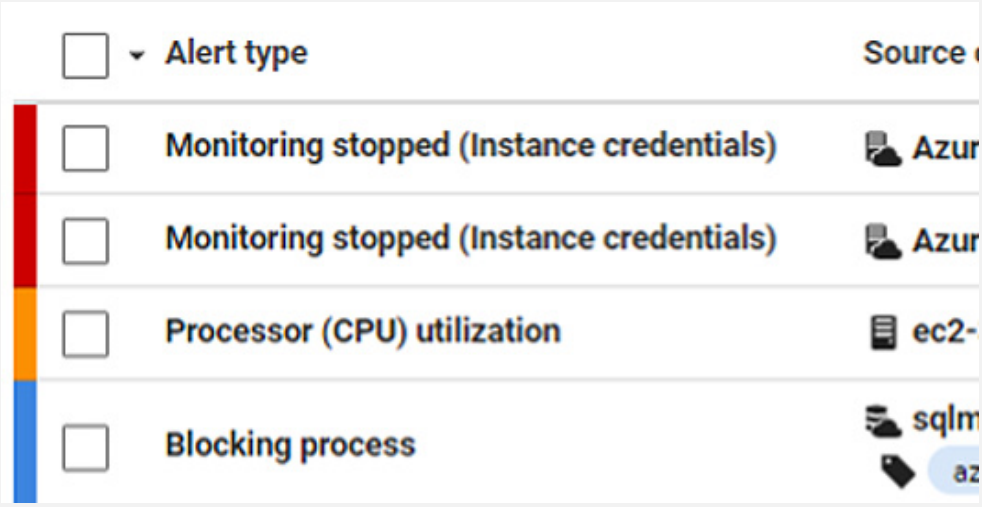
This will require access to the timelines that track resource use alongside user activity (processes and queries) and server events (deployments, data movement, maintenance jobs and so on), as described above. However, it will also often entail much more in-depth troubleshooting, using detailed **correlation analysis** for groups of metrics.

Sometimes, depending on the nature and impact of the incident reported, the response team may decide to implement temporary measures, or workarounds, to restore the service, while a full solution is investigated and tested. For a disk space alert, for example, this might involve temporary allocation of more space to a cloud-based service, by the Tier 1 team, while Tier 2 investigates the underlying cause of the issue. For a security incident, this might involve Tier 2 taking actions such as isolating the affected service, putting in place temporary access controls, or performing emergency backups. The Tier 3 response then does the detailed investigation, nullifies the threat, and documents the impact of, and responses to, the incident.

Intelligent, controlled alerting

The monitoring system should, of course, provide a set of accurate, reliable, configurable alerts that will inform the team when jobs fail or fail to run, database or server properties deviate from the established standard, there are sustained spikes or worrying trends in resource usage, database queries run slow, and so on. Where the monitoring tool used as input can provide alerts for not only the starting or worsening of a problem, but also the decrease or end of it, it becomes possible to provide much quicker feedback.

All alerts should be available for review in the **Alert** section of the monitoring tool's global dashboard. It is not just the alerts that are currently 'active' that need investigation. If a workaround or temporary measure, such as a "reboot", succeeds then the event that raised the alert will be ended, and any active alerts related to it will be listed as 'closed'. However, this does not mean the problem is fixed, and those 'closed' alerts may still need to be located, escalated as an incident.



Each alert merely represents a 'change' within the system and the effectiveness of the response relies on how easily the team can spot those changes that require prioritization, to avoid an incident. A shortage of memory within a server, for example, will be reported as an alert, but if appropriate action is taken, then an incident can be avoided.

In poorly configured monitoring systems, alerts that require action often get 'lost' among those that are inconsequential or redundant (repeat alerts for a known issue, for example). This very common issue is known as 'alert flooding'. The following sections discuss its impact, and then how we can avoid it via intelligent and configurable alerting mechanisms.

Event noise and alert flooding

Alert flooding is a consequence of excessive 'event noise'. Severe event noise makes it difficult for first responders to identify and prioritize critical issues, leading to alert fatigue and the potential for important alerts to be overlooked.

It happens because of misconfigurations, garrulous components, generic or overly sensitive alerting thresholds, or a lack of filtering mechanisms. It can be a particular problem in both scripted monitoring systems that often lack sufficient alert controls, and in generic monitoring systems, designed to show the health and performance of all infrastructure and applications and where events are configured to support a whole range of requirements.

Unfortunately, flooding will often occur at just the time you don't want it, such as during system failures, network issues, or large-scale events, which will tend to trigger a massive influx of alerts. This can prevent the monitoring system from effectively communicating important information, thereby causing delays in incident response and resolution.

Controlling alerts: categorization, filtering, aggregation

How do we avoid alert flooding? Every monitoring system should support alert classification, grouping and filtering. It should allow the user to assign a ranking of importance to individual alerts or classes of alert. It should be possible to filter alerts, to ensure, for example, that only alerts for events that indicate an active or imminent threat to the running of the service are sent to the unified notification system, for immediate assessment. It should be possible to adjust alerting thresholds to reduce false positives and filter out trivial events. This selection process isn't always easy though (if a password needs an unusual number of retries, is this trivial, or the first sign of a serious intrusion?).

In addition to these basic classification and threshold filtering mechanisms, an enterprise-class alerting system ought to support alert aggregation, as well as application of 'rules', by scripting, for both individual alerts and types of alerts:

- **Alert aggregation** – deduplicate redundant and repetitive alerts by consolidating them into a single notification.
- **Alert suppression** – suppression rules will temporarily suppress alerts during maintenance windows or known non-critical events.
- **Alert correlation** – apply rules that analyze relationships between different events and generate more meaningful, aggregated alerts.
- **Intelligent alerting mechanisms** – analyze historical data, trends, patterns and baseline information and generate alerts only where there are significant deviations from normal variation. Of course, this must take account of applications that do not exhibit a normal distribution, such as ticketing systems.

Many of these advanced alerting capabilities are very difficult to achieve in any DIY approach.

A global, estate wide dashboard for all database services

One of the biggest advantages of having a single, specialized monitoring tool collecting all this metric and alert data, for each type of service, is that it provides a single, consistent source of diagnostic data for that service. All the data, from the high-level summaries and visualizations used by the Tier 1 response team to the advanced analytics used by Tiers 2 and 3, and including the alert dashboard reviewed by all responders, use the same source of data. This makes collaboration and reporting, in incident management, much simpler.

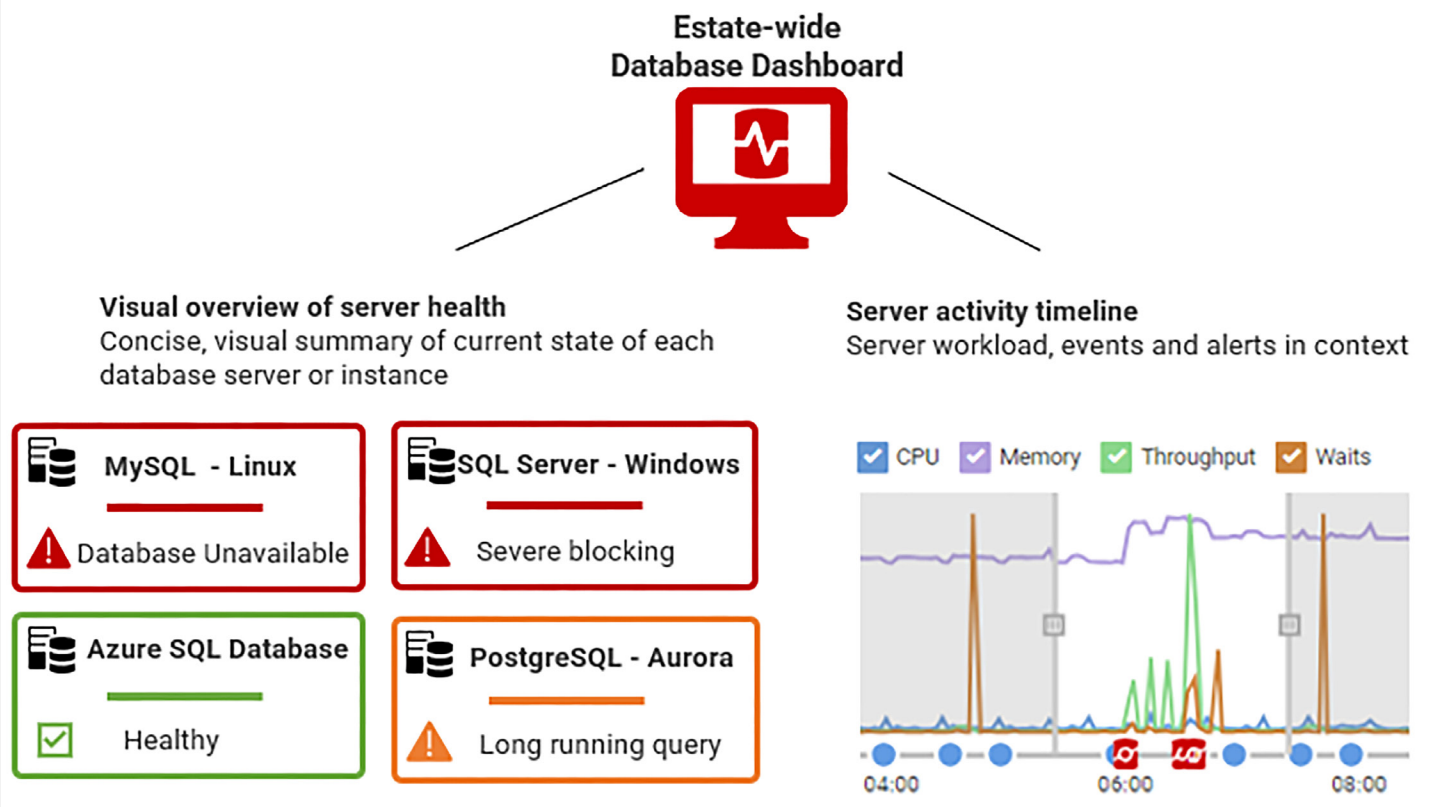
By collecting all the required performance metrics, on a schedule, and then performing intelligent analysis, the monitoring system can also add value to the data and makes it more useful and accessible to each response tier. It can present the users with the relevant summaries, comparisons and projections, at the right point in their performance investigation so that the team can, for example:

- **Quickly understand the impact on business operations or user**s – by tracking not just core database metrics but also allowing for custom metrics that monitor application KPIs.
- **Perform root cause analysis of complex problems** – by correlating the behavior of a combination of metrics, over time. Some issues may be correlated with and caused by, other processes.
- **Understand issues in the context of current activity** – using a graphical map or dashboard display that shows the processes, requests and events that happened around the time of the problem.
- **Review workload patterns in context of historical behaviour, using a baseline** – is current usage typical of the day of the week, or the month of the year?
- **Use trends and projections** – to estimate when trends in resource use, or server stress conditions, are likely to escalate into a problem that will affect the business.

Seeing the 'big picture'

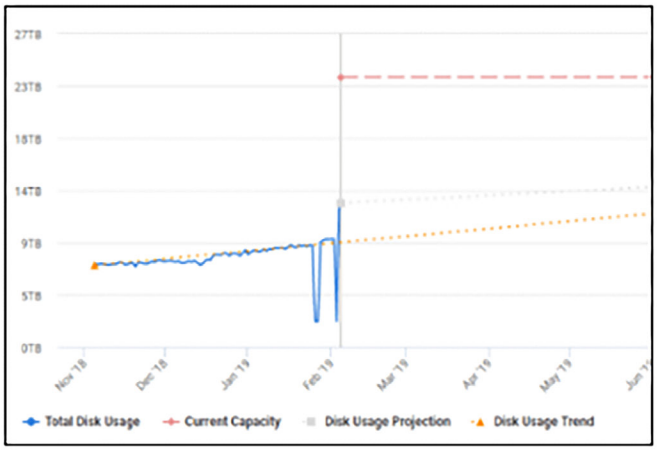
A global visual dashboard 'digests' all this data, at the highest level, and aims to use it to present a simple, graphical summary of the state of the system. If successful, it should allow any responder to rapidly appraise the state of even a complex application and network to assess the health, performance, and status of a system.

For example, the global dashboard in Redgate Monitor uses color-coded cards to present a simple visual summary of the health of every monitored server, instance and database Alongside this, a detailed **activity timeline** shows alerts, server events such as deployment data loads and so on) and resource usage around the time a problem was reported. Information like this will often make the source of the problem obvious to the Tier 1 team.

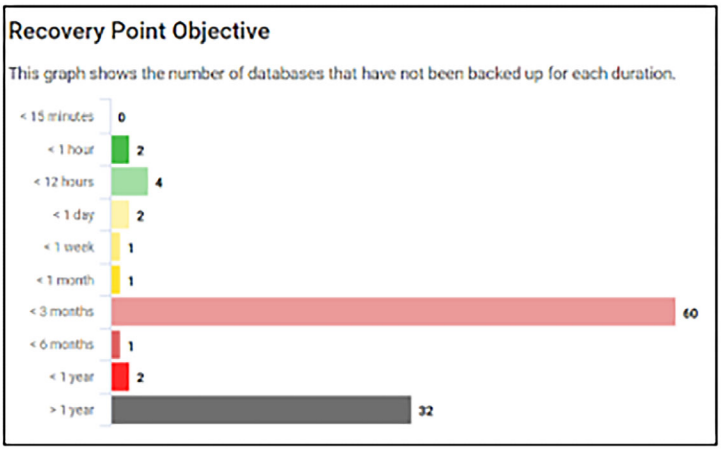


The dashboard will also produce charts, graphs, and other visual elements that can represent various aspects of system health, such as response times, resource utilization, error rates, or disk space consumption. These graphical systems are considered essential for operations teams and the first line response, to enable them to understand the context of urgent alerts and quickly identify potential issues. By collecting and analyzing data over time, the dashboard can also present graphical summaries, aggregations and projections. These will make it much easier to spot trends in usage and take pre-emptive corrective action.

Projections
Predict when trends in resource use, or server stress conditions, are likely to escalate into a problem that will affect the business



Summaries, rollups, aggregations
Review overall health of the database estate



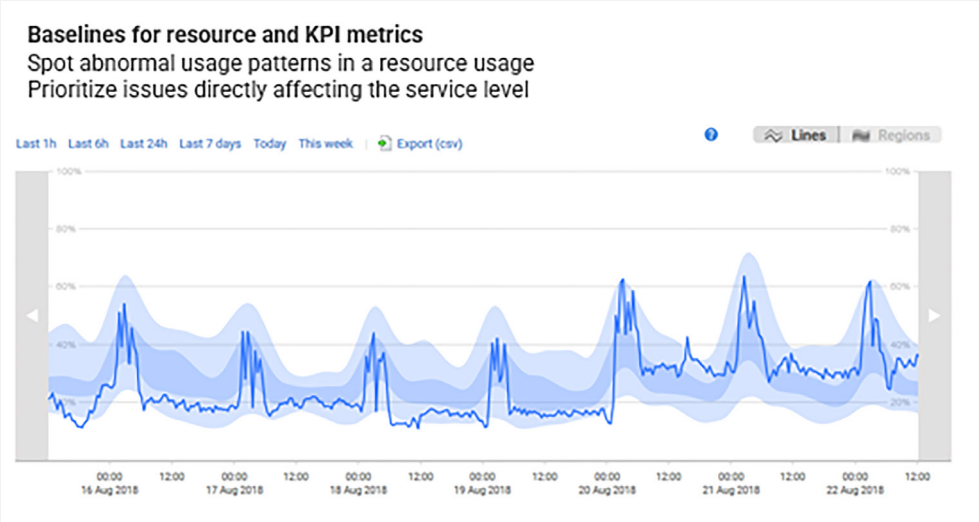
Once a responder has identified the source and nature of the problem they can then 'drill down' into the in-depth diagnostics, discussed previously, to investigate the cause and the appropriate course of action.

Using baselines and correlations

The Tier 2 and Tier 3 teams will need to drill successively deeper into a wider range of metrics and alerts, to arrive at an understanding of often complex patterns of database behaviour. A CPU metric indicating sustained high usage doesn't necessarily indicate the need for more processing power, and nor does high disk IO necessarily indicate an overwhelmed or underpowered disk subsystem. Enterprise data systems run complex queries, often very frequently, in parallel, and requesting large volumes of data. At times this will place high demands on CPU, disk IO, and on the memory required in the cache to store all the data. Any or all of these could become a bottleneck that affects the users of the application. One metric, viewed in isolation might indicate that a "slow disk subsystem" is causing an I/O bottleneck, but the 'bigger picture' will often reveal that it's simply the victim of excessive I/O caused by a problem elsewhere in the system.

Therefore, they will want to review not just individual metrics and alerts, but also how they interrelate, and how their behaviour varies over time. By understanding the behaviour of a combination of metrics, in response to a certain issue, the team can start to document the 'footprint' of even complex problems, making them easier to spot, and root cause analysis simpler.

They will also need to understand "*is this a normal pattern of behaviour*"? For example, if alerted to the fact that available storage capacity has decreased substantially over a period, does this need to be raised as an incident? To answer this question, we can use **baselines** to compare current behaviour to the behaviour observed over similar periods, historically. In this way, we might establish, for example, that it's perfectly expected for storage capacity to become depleted in the period just before an archive job runs.



If not normal behaviour, then has anything obvious happened to provoke the different pattern of behaviors? Is the increase resource usage simply a function of a busier than usual server? Has there been a recent deployment, maintenance task, system upgrade, data movement, or change in server or instance configuration? Perhaps, alternatively, it is a creeping problem that has developed over time into an issue that users are now noticing.

Use DevOps work practices to optimize database system monitoring

Organizations that wish to adopt a tiered approach to maintaining data systems will find it easier to do so if there exists a culture of cooperation between teams, and a mutual appreciation of the difficulties that accompany the development and support of a database application. This has always been true but good practice has been spelled out clearly by the DevOps movement.

There are several ways that developers and operations teams can cooperate to improve the ease with which databases can be monitored, the likelihood that issues can be detected and caught early, ideally before they ever reach the production system, and the effectiveness of the tiered response strategy.

Measure performance of KPIs

For a system under active development, the most important task for developers who are attuned to DevOps principles is to ('instrument') the application. This means providing metrics for that the monitoring system uses to record the performance of the Key Performance Indicators (KPIs) that are important to the user, as well as the components of the system.

Generally, these are best measured within the application, but from an independent network-based monitor. If we chose to monitor a KPI such as '*the time that elapses between registering a sale and dispatching an invoice*', for example, it is easier if this custom metric is built into the application but read by a custom monitor within the monitoring system. This allows us to use these custom monitors in the database monitoring system to track the activity and performance of application processes and send alerts to the notification system.

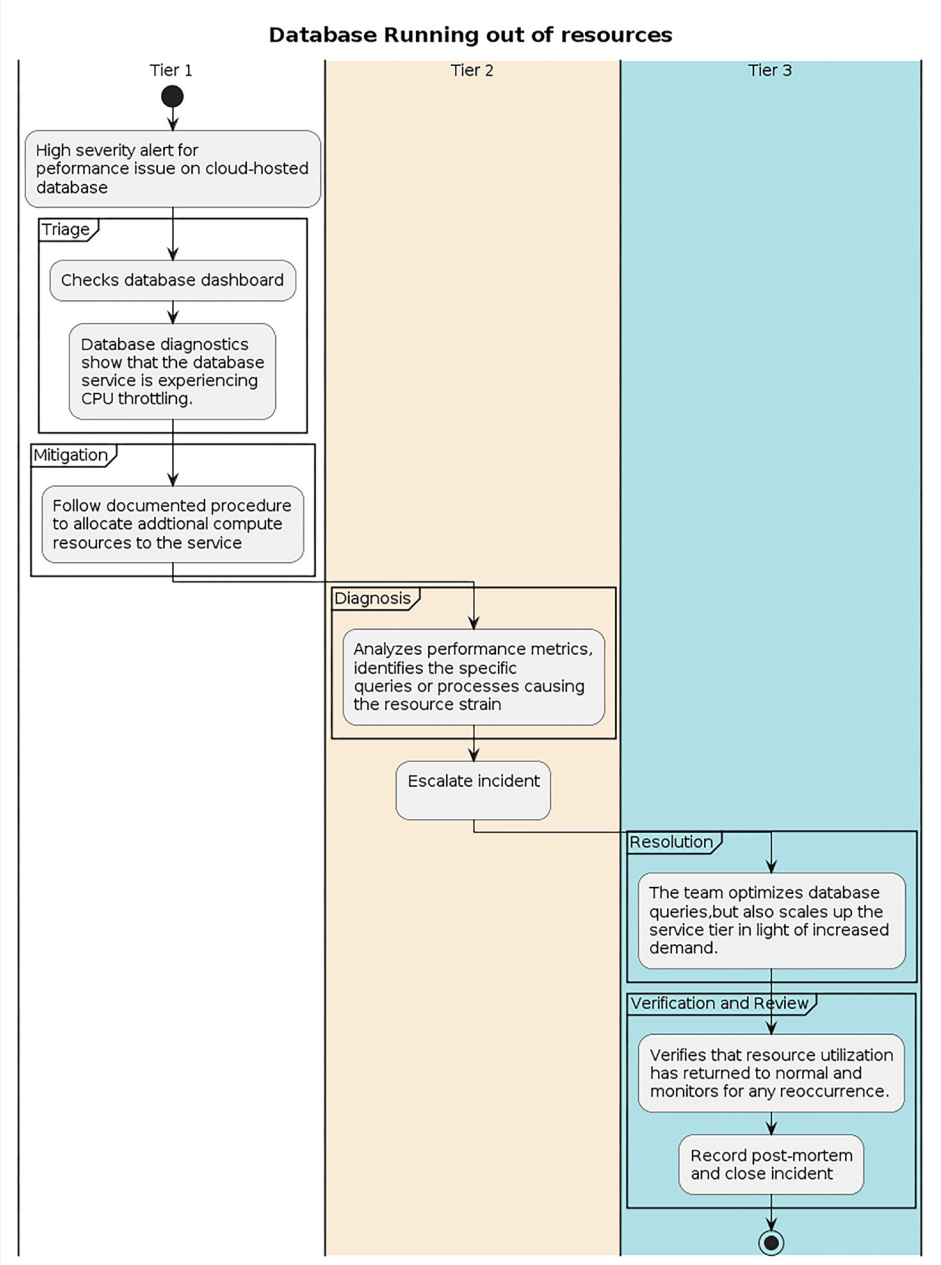
If we have custom metrics for the KPIs, and associated baselines and alert notifications, you will be alerted immediately of issues affecting end-user services, such as the failure of, or anomaly in, a business process such as the generation of an invoice. They also provide an objective way of measuring performance improvements. The bottom-line figure is in measuring the KPIs agreed in the Service Level Agreement, but it pays to get sign-off for the methods used to measure these KPIs so that there can be no source of friction or doubt about these figures.

Provide documented response procedures

An effective organization will understand the purpose and benefits of keeping records and documentation. The whole idea of empowering a first responder in Tier 1 relies on providing simple explanations for alerts, and clear instructions for 'resuscitation' of an application.

When alerted to a problem, the first responders can, and will, do everything possible to restore "normal service", while the underlying cause is investigated. If the problem has a simple and documented response, such as to make more storage space available to a database, switch in a pre-configured standby router, or increase the compute resources available to a cloud service, they will attempt it. In this way, the Tier 1 support team can fix many faults before any users even become aware of them. Even if following the prescribed process doesn't work, it will give the Tier 1 responders enough understanding of the problem to escalate to the next level of response (Tier 2) and notify the appropriate Tier 2 responder.

Tier 1 can take mitigation measures only if they have adequate documentation and instructions from the developers and operations experts, and a log of successful actions taken in the past for this type of alert. In the following example, the have documented instructions for increasing resource allocation, temporarily, to a database running on a cloud platform:



As with medical first-responders, Tier 1 responders need explicit instructions for every imaginable event, and a record of every successful intervention in the past. It pays to allow a range of people to be 'on call', to foster an appreciation of the requirements for an effective first-response.

Design database applications to be easy to monitor

A tiered response system becomes easier to optimize once database monitoring covers not only the production databases but has also been extended to the development and test systems.

This allows developers to test how the database behaves and performs, both for a 'normal' workload and when under stress, and to use monitoring to identify any serious flaws, bottlenecks, or inefficiencies in the operation of the database and the business processes it supports. They can rehearse possible causes of failure and understand what sort of monitoring data and alerting mechanisms need to be in place to "raise a red flag".

If the developers of a database had already predicted a particular problem, such as running low on disk space, they can provide documented instructions, and perhaps a batch file, that can be used by the first line of support, to allow them to solve the problem or at least allow the service to resume. When armed with basic instructions of how to deal with each of these categories of event, it is easier to fix or troubleshoot many problems promptly and effectively.

With this sort of system in place, Redgate Monitor can also play a useful part in coordinating the work of the whole team, in maintaining the required service level for any database system that uses SQL Server or PostgreSQL. When supplemented by custom monitors that can check the specific database application, this should provide everything required for a Tier 1 alerting system.

By following this approach, many of the systems that are needed for detecting problems and raising the appropriate alerts, will already be in place within the database system, at least as SQL queries or scripts, before the first release of the database application.

Encourage a culture of cooperation and continuous improvement

A DevOps culture can make a significant difference when applied to the development, running and monitoring of an organization's databases. It encourages closer liaison between development and operations, and uses of the tools and expertise appropriate to the type of problem that has occurred.

The Ops team provides guidance to developers on the most effective way the application can respond to each foreseeable problem, such as a failed backup or data corruption. Conversely, developers can assist the Ops team with automation scripts to deal with known, frequently occurring issues, for example using:

- **Infrastructure as code (IaC)** – can allow for consistent and repeatable provisioning of resources, and quick rebuilding of environments, reducing downtime and improving resilience.
- **Playbooks** – cross-functional teams can develop automated remediation scripts, to help address common issues without human intervention.

This sort of DevOps feedback loop involves using information from incidents, configurations, and application code to improve processes, continuously. This helps prevent recurring issues and serves to continuously improve the response system.

Conclusions

When maintaining an Enterprise database system, the sudden or creeping symptoms of 'ill health' are often caused by a complex chain of server events and user activity. To ensure a rapid and efficient incident response, we must continuously monitor, save and analyze a full range of metrics and alert data, across all supported platforms and types of database system, covering all the most common causes of bottlenecks and failures in this system. This ensures that the diagnostic data to link even complex symptoms to a cause will always be available when you need it. It is very difficult to compensate retrospectively for deficiencies in the available data, and it will lead to trial-and-error approach to incident response, delays and system downtime.

This whitepaper described a strategy that uses the Enterprise-class availability, performance and health diagnostics of a specialized database monitoring tool (Redgate Monitor) in an integrated, service-quality driven approach to incident management and response. It exploits the advanced alerting capabilities of the database monitoring tool to maximize signal-to-noise. It uses simple webhook integration to send urgent alert notifications to a unified notification system, for incidence management and response scheduling. This unified system also receives notifications from other monitors, such as the application performance monitor (APM) or network monitor.

In this integrated approach, the APM will warn you of any problems affecting the service level or availability of the applications. For example, perhaps a business process, such as invoice generation, is to unresponsive, or a website is experiencing timeouts.

Timelines, visual summaries and snapshots of activity, provided by the database dashboard, give the full context of any database incident and make the diagnostics accessible not just to your DBAs, network specialists and other experts but also to your frontline support team. By providing effective support for a tiered incident response, where every team can make a valuable contribution, we have a faster, more coordinated and ore scalable system of responding to service-quality problems that affect users, customers and the business.

The whitepaper outlines techniques to optimize this strategy. We add custom service KPI metrics to the database monitor to correlate them directly with events, user activity and resource use in the database system. We prevent issues by left-shifting monitoring into test and development systems. We document the 'footprint' of known issues with stepwise response procedures for a more effective frontline response, with fewer issues ever reaching operations staff and other experts.

Over time we achieve a system that prioritizes continuous improvement and prevention to improve the resilience of the infrastructures and business systems. When unexpected failures occur, the response is rapid, coordinated and effective, minimizing reliance on the heroic 'firefighting' of a few experts and so allowing them to focus far more on their strategic roles and objectives.

Try the **Redgate Monitor demo online**, **download a free trial**, and **contact our experts** for more info on scaling your database monitoring.