CLOUDSWITCHED

FREE RESOURCE — AUDIT CHECKLIST

Database Performance Audit Checklist

A comprehensive audit of your database performance across query optimisation, indexing, storage I/O, connection pooling, caching and monitoring — 50+ checkpoints across six sections to identify bottlenecks and improve responsiveness.

6	50+	/10	Fillable
AUDIT SECTIONS COVERED	ITEMS TO CHECK	SCORE EACH SECTION	TICK & TYPE IN ANY VIEWER

QUERY TUNING · INDEXING · STORAGE I/O · MONITORING

PREPARED FOR	PREPARED BY	VERSION	FORMAT
Cloudswitched Knowledge Library	Cloudswitched Ltd.	2026 Edition	Interactive PDF

00

How to use this checklist

Work through each of the six sections with your DBA or your managed service provider. Run the recommended diagnostics **before** scoring each area, and tick only the checkpoints that are **fully in place** — in production and verifiable today, not "we're working on it".

Flag any section scoring **below 6** for immediate remediation and carry it into a prioritised performance improvement plan on the summary pages. This is an **interactive PDF** — tick the boxes and type your scores, owners and notes directly in any modern PDF viewer.

WHAT THIS CHECKLIST IS

A practical performance baseline covering the levers that actually move database responsiveness — query plans, indexing, storage throughput, connection pooling, memory and observability. It is not a substitute for a deep engagement on a specific workload, but it is a fast way to surface the bottlenecks worth investigating first.

SCORING GUIDANCE

Award points where you have evidence the control is in place — a configuration, a monitoring dashboard, a report, a tested change. Half measures do not score. Round each section to a whole number out of 10 and carry it to the summary table.

The six sections

- **01 Query Performance & Optimisation** — slow query log, execution plans, parameterisation, N+1.
- **02 Indexing Strategy** — covering and composite indexes, fragmentation, unused indexes.
- **03 Storage & I/O Performance** — SSD/NVMe, log separation, TempDB, latency.
- **04 Connection Management & Pooling** — pool sizing, leaks, timeouts, governor.
- **05 Caching & Memory Configuration** — buffer pool, hit ratios, Redis, invalidation.
- **06 Monitoring & Alerting** — dashboards, thresholds, deadlocks, retention.

01

Query Performance & Optimisation

Poorly performing queries are the most common cause of database slowdowns. Identify and resolve the worst offenders first — pull the slow query log before you score this section.

- ☐ The **slow query log** is enabled and reviewed to identify queries exceeding acceptable execution times *(threshold: 1 second for OLTP workloads)*.
- ☐ The **top 20 most resource-intensive queries** have been identified using execution plans and ranked by cumulative CPU and I/O cost.
- ☐ All frequently executed queries use **parameterised statements** rather than dynamic SQL to enable plan caching and prevent SQL injection.
- ☐ Complex queries have been reviewed for **unnecessary joins, subqueries and correlated subqueries** that could be refactored for efficiency.
- ☐ Queries returning large result sets include appropriate **pagination** rather than fetching all rows and filtering in the application layer.
- ☐ The query **execution plan cache hit ratio** is monitored and consistently above 90% to avoid repeated compilation overhead.
- ☐ Application code has been reviewed to eliminate **N+1 query patterns** where loops issue individual queries instead of batch operations.
- ☐ Stored procedures and views are periodically reviewed for **performance regression** as data volumes grow *(schedule quarterly reviews)*.
- ☐ **Database statistics** are updated regularly so the query optimiser has accurate cardinality estimates for plan selection.

WHAT GOOD LOOKS LIKE

Every slow query has an owner and a ticket. The top offenders are ranked by total cost, not single-run time, and parameterised statements keep the plan cache warm. Quarterly reviews catch regressions before users do.

SECTION 01 SCORE _____ / **10** *Aim for 8+. Below 6 = query / plan-cache gap.*

OWNER _____ **TARGET DATE** _____

02

Indexing Strategy

Effective indexing dramatically improves read performance, but over-indexing degrades writes. Strike the right balance for your workload rather than indexing everything.

- ☐ All tables have a **primary key** defined and a clustered index strategy aligned with the most common access patterns.
- ☐ **Missing index recommendations** from the engine's advisor have been reviewed and selectively implemented based on workload analysis.
- ☐ **Unused and duplicate indexes** have been identified and removed to reduce storage overhead and write amplification.
- ☐ **Composite indexes** are ordered with the most selective column first and cover the most frequently filtered and sorted columns.
- ☐ **Covering indexes** are used for high-frequency queries to enable index-only scans and eliminate table lookups.
- ☐ **Index fragmentation** is monitored and maintenance is scheduled during off-peak windows *(rebuild above 30% fragmentation)*.
- ☐ The ratio of **index reads to index writes** is tracked to ensure indexes deliver value relative to their maintenance cost.
- ☐ **Filtered or partial indexes** are considered for tables with large volumes of rarely queried historical data.
- ☐ The impact of new indexes is **tested in staging** before production deployment to verify improvement without side effects.

WATCH OUT FOR

Indexes added one incident at a time, never reviewed, never removed. Write-heavy tables slowly choke under a dozen overlapping indexes nobody can account for. If you cannot name what each index is for, you have too many.

SECTION 02 SCORE _____ / 10 *Aim for 8+. Below 6 = index balance gap.*

OWNER _____ **TARGET DATE** _____

03

Storage & I/O Performance

Database performance is ultimately constrained by storage throughput. Ensure your I/O subsystem is not the silent bottleneck behind every slow query.

- ☐ Database files are hosted on **SSD or NVMe** storage with sufficient IOPS for peak workload demands *(spinning disks are inadequate for production)*.
- ☐ **Data files and transaction logs** are on separate physical volumes to prevent I/O contention between read/write and logging.
- ☐ **TempDB / temporary tablespace** is on dedicated fast storage as it handles sort operations, hash joins and spills from memory.
- ☐ **Storage latency** is continuously monitored and consistently below 5ms for reads and 2ms for writes.
- ☐ Database **autogrowth settings** use fixed-size increments rather than percentage-based growth to prevent file fragmentation.
- ☐ Database file sizes are **pre-allocated** with adequate headroom to minimise autogrowth events during business hours.
- ☐ **Disk queue lengths** are monitored and sustained queue depths above 2 per disk are investigated as I/O bottlenecks.
- ☐ **Backup operations** are scheduled to avoid peak I/O periods and use compression to reduce storage throughput requirements.

MEASURE BEFORE YOU SPEND

Before buying faster storage, confirm storage is actually the constraint. Read/write latency and disk queue length tell you in minutes whether the I/O subsystem or the query plans are at fault.

SECTION 03 SCORE _____ / 10 *Aim for 8+. Below 6 = I/O subsystem gap.*

OWNER _____ **TARGET DATE** _____

04

Connection Management & Pooling

Efficient connection management prevents resource exhaustion and ensures consistent response times under load. Connection leaks fail quietly until the pool is empty.

- ☐ Application connection strings use **connection pooling** with minimum and maximum pool sizes tuned to the workload *(avoid unlimited max pool size)*.
- ☐ The **maximum concurrent connections** on the server reflects actual peak demand plus a reasonable headroom margin.
- ☐ **Connection timeout settings** release abandoned connections promptly rather than holding resources indefinitely.
- ☐ The application properly **closes and returns connections** to the pool after each operation — no connection leaks identified.
- ☐ **Connection pool health checks** validate connections before reuse to prevent errors from stale or broken connections.
- ☐ **Long-running queries** are identified and terminated if they exceed acceptable thresholds to prevent connection starvation.
- ☐ A **resource governor or workload management** is configured to prevent any single application from monopolising connections.
- ☐ **Connection pool metrics** (active, idle, waiting, timed-out) are monitored and alerting is configured for pool exhaustion.

THE CLASSIC FAILURE

Everything is fine until a traffic spike, then the pool drains, requests queue, timeouts cascade and the database looks 'down' while sitting nearly idle. The fix is almost always a leaked connection or an unbounded pool.

SECTION 04 SCORE _____ / **10** *Aim for 8+. Below 6 = connection / pool gap.*

OWNER _____

TARGET DATE _____

05

Caching & Memory Configuration

Properly configured memory and caching reduce I/O pressure and deliver consistent sub-millisecond response times for frequently accessed data.

- ☐ The server has **sufficient RAM** to hold the active working set in the buffer pool without excessive paging to disk.
- ☐ **Buffer pool / cache hit ratio** is monitored and consistently above 95% for OLTP workloads *(below 90% indicates insufficient memory)*.
- ☐ A **maximum server memory** setting leaves adequate headroom for the operating system and other processes.
- ☐ **Application-level caching** (Redis, Memcached) is deployed for frequently read, rarely changed data to offload repetitive queries.
- ☐ The **query plan cache** is appropriately sized to retain plans for frequently used queries without consuming excessive memory.
- ☐ **Page life expectancy** is monitored and consistently above 300 seconds to confirm adequate buffer pool sizing.
- ☐ **Memory grants** for large sort and hash operations are monitored to prevent excessive grants that starve other queries.
- ☐ **Cache invalidation** strategies ensure stale data is not served from application caches after database updates.

THE CHEAPEST WIN

For most OLTP databases, RAM is the highest-leverage upgrade available. A buffer pool large enough to hold the working set turns disk reads into memory reads and lifts every query at once.

SECTION 05 SCORE _____ / **10** *Aim for 8+. Below 6 = memory / cache gap.*

OWNER _____

TARGET DATE _____

06

Monitoring & Alerting

Proactive monitoring detects performance degradation before users notice. Reactive troubleshooting is always more expensive than the alert you did not configure.

- ☐ A **dedicated database monitoring solution** is deployed *(e.g. Datadog, SolarWinds DPA, pganalyze, Percona Monitoring)* with real-time dashboards.
- ☐ Alerts are configured for **CPU, memory pressure, disk I/O latency and connection count** exceeding defined thresholds.
- ☐ **Slow query alerts** trigger when execution times exceed baseline or when new queries appear in the slow query log.
- ☐ **Deadlock detection** is enabled with automatic alerting and logging to identify contention between concurrent transactions.
- ☐ **Backup success and failure** is monitored with immediate alerting on any backup failure or missed backup window.
- ☐ **Replication lag** (if applicable) is monitored continuously with alerts when lag exceeds your RPO thresholds.
- ☐ Scheduled **performance reports** are generated weekly comparing key metrics against baseline to spot gradual degradation.
- ☐ All **monitoring data is retained for at least 90 days** to support capacity planning, trend analysis and post-incident review.
- ☐ An **annual performance review** compares year-over-year metrics and feeds into capacity planning and budget discussions.

BASELINES BEAT THRESHOLDS

A static ‘CPU > 80%’ alert is noise. The valuable signal is deviation from your own baseline — this Tuesday 10am vs every Tuesday 10am. Retain 90 days so the baseline is real.

SECTION 06 SCORE _____ / **10** *Aim for 8+. Below 6 = observability gap.*

OWNER _____

TARGET DATE _____

Σ

Audit score summary

Transfer the score for each section into the table. Set a priority (H/M/L) based on the gap to target. Anything below 6 is a candidate for the top-3 actions on the next page.

#	AUDIT AREA	SCORE / 10	PRIORITY
01	Query Performance & Optimisation	_____ / 10	☐ H ☐ M ☐ L
02	Indexing Strategy	_____ / 10	☐ H ☐ M ☐ L
03	Storage & I/O Performance	_____ / 10	☐ H ☐ M ☐ L
04	Connection Management & Pooling	_____ / 10	☐ H ☐ M ☐ L
05	Caching & Memory Configuration	_____ / 10	☐ H ☐ M ☐ L
06	Monitoring & Alerting	_____ / 10	☐ H ☐ M ☐ L
Σ	**TOTAL SCORE**	_____ / 60	—

READING THE TOTAL

Six sections, ten points each — a maximum of 60. Treat the total as a direction of travel, but let the individual section scores drive the work: one section at 3 matters more than a balanced 7 across the board.

Interpretation & priority actions

Translate your total score into a risk band, then commit to a small number of next steps. The goal is one page of decisions, not a wish list.

Score interpretation

48–60	**Excellent.** Your database is well-tuned. Focus on continuous monitoring and capacity planning for growth.
36–47	**Good foundation, gaps exist.** Prioritise any section scoring below 6 with owners and deadlines.
Below 36	**Significant gaps.** Performance risk is material — consider an urgent review with a database specialist.

Top 3 priority actions

01 _____

02 _____

03 _____

Additional notes

_____

_____

_____

AUDIT COMPLETED BY _____ DATE _____ NEXT REVIEW DUE _____

Need help closing the gaps?

Cloudswitched tunes and operates databases for UK SMEs — query optimisation, indexing, storage and monitoring, with fixed-price remediation plans.

info@cloudswitched.com

cloudswitched.com/services

CLOUDSWITCHED

IT SUPPORT & PROJECT SERVICES

info@cloudswitched.com

New London House, 6 London St
London EC3R 7LP · United Kingdom