CLOUDSWITCHED

FREE RESOURCE — GUIDE

Database Scaling & Optimisation Guide

Practical guide to scaling your database infrastructure for growing UK businesses — capacity planning, vertical and horizontal scaling, read replicas, and cost optimisation strategies.

5
SCALING STRATEGIES

30+
EXPERT TIPS

Frameworks
INCLUDED THROUGHOUT

Free
PRINT & USE NO STRINGS

About This Guide

This guide provides practical, actionable advice for UK businesses. Work through each section to build a comprehensive understanding of the topic. Use the information to make informed decisions and implement best practices.

Need Help With Your IT?
Our team can help you implement the recommendations in this resource.

info@cloudswitched.com
+44 2030 043 450
New London House, 6 London St, London EC3R 7LP

1 Capacity Planning & Assessment

Effective scaling starts with understanding where you are today and where you need to be. Data-driven capacity planning prevents both over-provisioning waste and under-provisioning outages.

Before investing in scaling infrastructure, you need a clear picture of your **current resource utilisation, growth trajectory, and performance bottlenecks**. Many organisations scale prematurely or in the wrong dimension — adding CPU when the bottleneck is I/O, or adding replicas when the problem is poorly optimised queries.

Key Metrics to Baseline

- **CPU utilisation:** Track average and peak CPU usage over a full business cycle (at least 4 weeks). Sustained utilisation above 70% during business hours indicates scaling pressure.
- **Memory utilisation:** Monitor buffer pool hit ratio, page life expectancy, and memory grants. If your working set exceeds available memory, query performance degrades significantly.
- **Storage I/O:** Measure IOPS, throughput (MB/s), and latency for both read and write operations. Storage is frequently the first bottleneck to hit in growing databases.
- **Connection count:** Track concurrent connections over time. Approaching the maximum configured limit causes connection timeouts and application errors.
- **Database size:** Monitor total database size and growth rate. Project when you will reach storage limits, licensing thresholds, or backup window constraints.

Growth Projection Framework

METRIC	CURRENT VALUE	6-MONTH PROJECTION	12-MONTH PROJECTION	ACTION THRESHOLD
Database size (GB)				80% of storage limit
Peak concurrent connections				70% of max configured
Peak CPU utilisation (%)				70% sustained
Buffer pool hit ratio (%)				Below 95%
Average query response time (ms)				Exceeds SLA target
Peak IOPS				80% of storage tier limit
Daily transaction volume				Capacity planning input

Optimise Before You Scale

Scaling hardware to compensate for inefficient queries is expensive and unsustainable. Always exhaust query optimisation, indexing improvements, and caching strategies before investing in infrastructure scaling. A single poorly written query can consume more resources than an entire application.

Document your findings in a **capacity planning report** that is reviewed quarterly. This report should drive budget requests, procurement timelines, and architecture decisions well in advance of hitting capacity limits.

2 Vertical Scaling Strategies

Scaling up (adding more resources to your existing server) is the simplest approach and should be your first consideration for most workloads.

Vertical scaling — increasing CPU, memory, or storage on a single server — is the **lowest-risk, lowest-complexity scaling approach**. It requires no application changes, no data redistribution, and no architectural redesign. For the majority of UK SMEs, vertical scaling is sufficient to handle growth for years.

When Vertical Scaling Works Best

- **Memory-bound workloads:** If your buffer pool hit ratio is below 95% or page life expectancy is dropping, adding RAM delivers immediate and significant performance improvement.
- **CPU-bound analytics:** Reporting queries, aggregations, and complex joins benefit directly from additional CPU cores and faster processors.
- **I/O-bound workloads:** Migrating from spinning disks to SSD, or from SSD to NVMe, can deliver 10–100x improvements in I/O latency and throughput.
- **Cloud-hosted databases:** Cloud providers allow vertical scaling with minimal downtime — Azure SQL and AWS RDS can scale compute tiers in minutes.

Cost Considerations for UK Businesses

RESOURCE	ON-PREMISES COST	CLOUD COST (APPROX. MONTHLY)	IMPACT
RAM (32 GB → 64 GB)	£200–£400 one-off	£50–£150 increment	Buffer pool, caching
CPU (8 cores → 16 cores)	£1,000–£3,000 one-off	£100–£400 increment	Query processing, concurrency
Storage (SSD → NVMe)	£500–£2,000 one-off	£30–£100 increment	I/O latency, throughput
SQL Server licensing (per core)	£3,000–£15,000 per core pair	Included in managed tier	Significant cost factor

Licensing Costs Can Dwarf Hardware Costs

For databases licensed per core (SQL Server Enterprise, Oracle), adding CPU cores doubles your licensing cost. Always calculate the total cost including licensing before scaling CPU vertically. This is often the trigger that drives migration to open-source alternatives or cloud-managed services.

Vertical scaling has a ceiling — eventually you reach the maximum specification available for a single server. For most UK SMEs running business applications, that ceiling is **far higher than you think**. A modern server with 128 GB RAM, 32 cores, and NVMe storage can handle millions of transactions per day.

3 Horizontal Scaling & Sharding

Scaling out by distributing data across multiple servers is powerful but complex. Understand the trade-offs before committing to this path.

Horizontal scaling — distributing your database across multiple servers — is necessary when vertical scaling reaches its limits or when **geographic distribution, fault tolerance, or extreme throughput** demands require it. However, it introduces significant architectural complexity that must be justified by genuine requirements.

Sharding Strategies

- **Range-based sharding:** Data is distributed by value ranges (e.g., customers A–M on shard 1, N–Z on shard 2). Simple to implement but prone to hotspots if data distribution is uneven.
- **Hash-based sharding:** A hash function distributes data evenly across shards. Provides better load distribution but makes range queries across shards more complex.
- **Geographic sharding:** Data is partitioned by region (e.g., UK customers on UK servers, EU customers on EU servers). Supports data sovereignty requirements and reduces latency for local users.
- **Tenant-based sharding:** Each customer or tenant gets their own database or schema. Common in multi-tenant SaaS applications and simplifies data isolation.

Complexity Trade-Offs

CONSIDERATION	SINGLE SERVER	HORIZONTALLY SCALED
Query complexity	Standard SQL	Cross-shard queries require coordination
Transaction support	Full ACID	Distributed transactions are slow and complex
Data consistency	Immediate	Eventually consistent (in many designs)
Operational overhead	Single server to manage	Multiple servers, rebalancing, monitoring
Application changes	None required	Significant refactoring for shard-aware queries
Backup and recovery	Standard procedures	Per-shard backup coordination
Cost	Single server cost	Multiple servers plus orchestration tooling

Most UK SMEs Do Not Need Sharding

Sharding is an engineering solution for extreme scale. If your database is under 500 GB and serves fewer than 1,000 concurrent users, vertical scaling and read replicas will almost certainly meet your needs at a fraction of the cost and complexity. Do not over-engineer.

Consider Managed Distributed Databases

If you genuinely need horizontal scaling, consider managed distributed databases (Azure Cosmos DB, Google Cloud Spanner, CockroachDB) that handle sharding, replication, and rebalancing automatically. The operational overhead of self-managed sharding is substantial and rarely justified.

Before committing to horizontal scaling, exhaust every other option: **query optimisation, indexing, caching, connection pooling, read replicas, and vertical scaling**. Only shard when you have clear evidence that no simpler approach can meet your requirements.

4 Read Replicas & Load Distribution

Offloading read traffic to replicas is often the most cost-effective scaling strategy for read-heavy business applications.

Most business applications are **heavily read-biased** — reporting, dashboards, searches, and data lookups far outnumber write operations. Read replicas allow you to scale read capacity independently of write capacity without the complexity of full horizontal scaling.

Read Replica Architecture

- **Primary handles all writes:** INSERT, UPDATE, and DELETE operations are directed exclusively to the primary database server, which maintains the authoritative copy of all data.
- **Replicas handle read traffic:** SELECT queries for reporting, dashboards, search, and analytics are routed to one or more read replicas, offloading the primary server.
- **Replication lag awareness:** Data on replicas may be seconds behind the primary. Applications must be designed to tolerate this — reading from a replica immediately after writing to the primary may return stale data.
- **Load balancing:** A connection proxy or application logic distributes read queries across multiple replicas for additional throughput and resilience.

Use Cases for Read Replicas

- **Business reporting:** Dashboards, scheduled reports, and ad-hoc queries run against a replica without impacting transactional performance on the primary.
- **Search and analytics:** Full-text search, aggregations, and data analysis workloads are directed to dedicated read replicas optimised for analytical queries.
- **Backup offloading:** Backups are taken from a read replica rather than the primary, eliminating backup I/O impact on production transactions.
- **Geographic distribution:** Replicas in different UK or European regions reduce read latency for distributed teams and improve disaster recovery posture.

Implementation Considerations

DATABASE ENGINE	NATIVE REPLICATION	MANAGED REPLICA OPTION	TYPICAL LAG
PostgreSQL	Streaming replication	AWS RDS, Azure Flexible	Sub-second
MySQL / MariaDB	Binary log replication	AWS RDS, Azure MySQL	Sub-second to seconds
SQL Server	Always On Availability Groups	Azure SQL, AWS RDS	Sub-second (sync mode)
MongoDB	Replica sets (built-in)	Atlas (managed)	Sub-second

Start with One Replica

A single read replica is sufficient for most UK SMEs. It offloads reporting and analytics from the primary, provides a warm standby for disaster recovery, and serves as a backup source. Add additional replicas only when monitoring shows the first replica is saturated.

Read replicas are the **best value scaling investment** for most business databases. They require minimal application changes (primarily connection routing), leverage native database replication, and deliver immediate measurable improvement in primary server performance.

5 Cost Optimisation & Right-Sizing

Database infrastructure is often over-provisioned. Right-sizing ensures you pay for what you need and invest savings in genuine performance improvements.

UK businesses frequently over-provision database infrastructure "just in case" or because specifications were set during initial deployment and never revisited. A disciplined approach to **right-sizing, reserved capacity, and architecture optimisation** can reduce database costs by 30–50% without impacting performance.

Cloud Cost Optimisation

- **Right-size compute tiers:** Review actual CPU and memory utilisation over 30 days. If average utilisation is below 40%, you are likely paying for capacity you do not use. Downsize to the next tier and monitor.
- **Reserved instances:** For stable, predictable workloads, purchasing 1-year or 3-year reserved capacity on AWS or Azure saves 30–60% compared to on-demand pricing. Commit only for base-level requirements.
- **Storage tier optimisation:** Not all data needs premium storage. Move cold and archival data to standard SSD or even HDD tiers. Use automated tiering policies where available.
- **Dev/test environments:** Non-production databases should use the smallest viable tier and be shut down outside business hours. Automate start/stop schedules to eliminate overnight and weekend waste.
- **Serverless options:** For variable or intermittent workloads, serverless database tiers (Azure SQL Serverless, Aurora Serverless) scale to zero when idle and charge only for actual usage.

On-Premises Cost Optimisation

- **Licensing review:** Audit your database licensing to ensure you are not paying for Enterprise features when Standard edition meets your requirements. The cost difference is substantial — often £10,000+ per server.
- **Consolidation:** Multiple under-utilised database servers can often be consolidated onto fewer, more powerful hosts. This reduces licensing, hardware, and operational costs simultaneously.
- **Open-source migration:** Evaluate whether PostgreSQL or MariaDB can replace commercial databases for non-critical workloads. Licensing savings fund the migration effort many times over.
- **Hardware refresh cycles:** Modern hardware delivers significantly more performance per pound than equipment from 3–5 years ago. A strategic refresh can improve performance while reducing power and cooling costs.

Cost Tracking Framework

COST CATEGORY	CURRENT MONTHLY	OPTIMISED MONTHLY	ANNUAL SAVING	ACTION REQUIRED
Compute (CPU / RAM)				
Storage (SSD / HDD)				
Licensing				
Backup storage				
Monitoring tools				
Support contracts				
Dev/test environments				
TOTAL				

Never Sacrifice Resilience for Cost

Cost optimisation must never compromise backup integrity, disaster recovery capability, or security controls. Saving £200 per month on a backup solution is meaningless if a data loss event costs your organisation £50,000 in recovery, downtime, and regulatory fines. Optimise intelligently.

Schedule a **quarterly cost review** that examines utilisation data, compares actual spend against budget, and identifies optimisation opportunities. The cloud cost landscape changes rapidly — new instance types, pricing models, and discount programmes appear regularly. Staying current saves money.