Optimizing Oracle & SQL Server in the Cloud A practical framework for high-performance database management on AWS and Azure

By : Adithya Sirimalla
Enliven Technologies
Conf42.com Robotics 2025

The Critical Role of Database Performance

Why It Matters Now

As robotics, automation, and data-driven systems scale across industries, the efficiency of underlying databases becomes mission-critical. Every millisecond of query latency can impact real-time decision-making in automated systems.

Modern cloud platforms offer unprecedented scalability, but without proper optimisation, organisations risk poor performance, escalating costs, and unstable execution plans.

Today's Agenda

1. Proven optimisation techniques for Oracle and SQL Server
2. Cloud-native tools on AWS and Azure
3. Practical strategies for query and resource management
4. Cross-platform benchmarking and governance
5. Emerging trends and future roadmap

Foundational Optimisation Techniques

Before leveraging cloud-native tools, mastering core database optimization principles is essential. These proven techniques form the backbone of high-performance database management.

SQL Plan Management

Ensures execution plan stability by capturing and controlling query execution paths, preventing performance regressions during system changes or updates.

AWR Analysis

Oracle's Automatic Workload Repository provides comprehensive performance diagnostics, identifying bottlenecks through historical performance data and wait event analysis.

Query Store Insights

SQL Server's Query Store tracks query performance over time, enabling plan regression identification and forced execution plan selection for optimal performance.

Cloud-Native Performance Tools

AWS Performance Insights

Amazon's Performance Insights offers real-time monitoring and analysis of database load, pinpointing bottlenecks quickly. It visualizes activity by wait events, SQL, and users, supporting RDS and Aurora instances.

Azure SQL Intelligent Tuning

Microsoft's intelligent tuning uses machine learning to automatically detect and fix performance issues. It includes automatic index management and plan correction, adapting to evolving workload patterns.

Optimising Query Execution

Optimising query execution is crucial for database efficiency and resource utilisation.

Identify Problematic Queries

Use tools like AWR reports or Performance Insights to find queries consuming excessive resources or with long durations.

Analyse Execution Plans

Examine execution plans for issues like table scans or missing indexes to pinpoint optimisation opportunities.

Apply Optimisations

Implement targeted improvements like index creation or query rewriting. Test thoroughly before deployment.

Monitor and Iterate

Continuously track performance after optimisation and adjust as database workloads evolve.

Resource Utilisation and Workload Throughput

1

Memory Configuration

Implement adaptive memory allocation and monitor memory pressure to optimize buffer cache, shared pool, and PGA (Oracle) or buffer pool (SQL Server).

2

I/O Optimisation

Utilize high-performance cloud storage (e.g., AWS gp3/io2, Azure Premium SSD) and implement table/index partitioning to reduce I/O overhead for large datasets.

3

CPU Efficiency

Right-size compute resources based on workload demands. Track CPU utilization with tools like CloudWatch/Azure Monitor for effective vertical or horizontal scaling.

4

Connection Pooling

Configure connection pooling at application and database levels to minimize connection overhead, especially crucial for reducing latency in cloud environments.

Execution Plan Stability

Maintaining consistent query performance requires careful management of execution plans. Plan instability often leads to unpredictable performance degradation, particularly after statistics updates. Tools like SQL Plan Management (Oracle) and Query Store Plan Forcing (SQL Server) capture known-good plans, preventing regressions.

These mechanisms allow you to baseline optimal plans during low-activity periods and enforce them during production workloads. Regular plan reviews and automated alerts for significant plan changes are essential to ensure continued optimal performance as data evolves.

Partitioning Strategies for Scale

Why Partition?

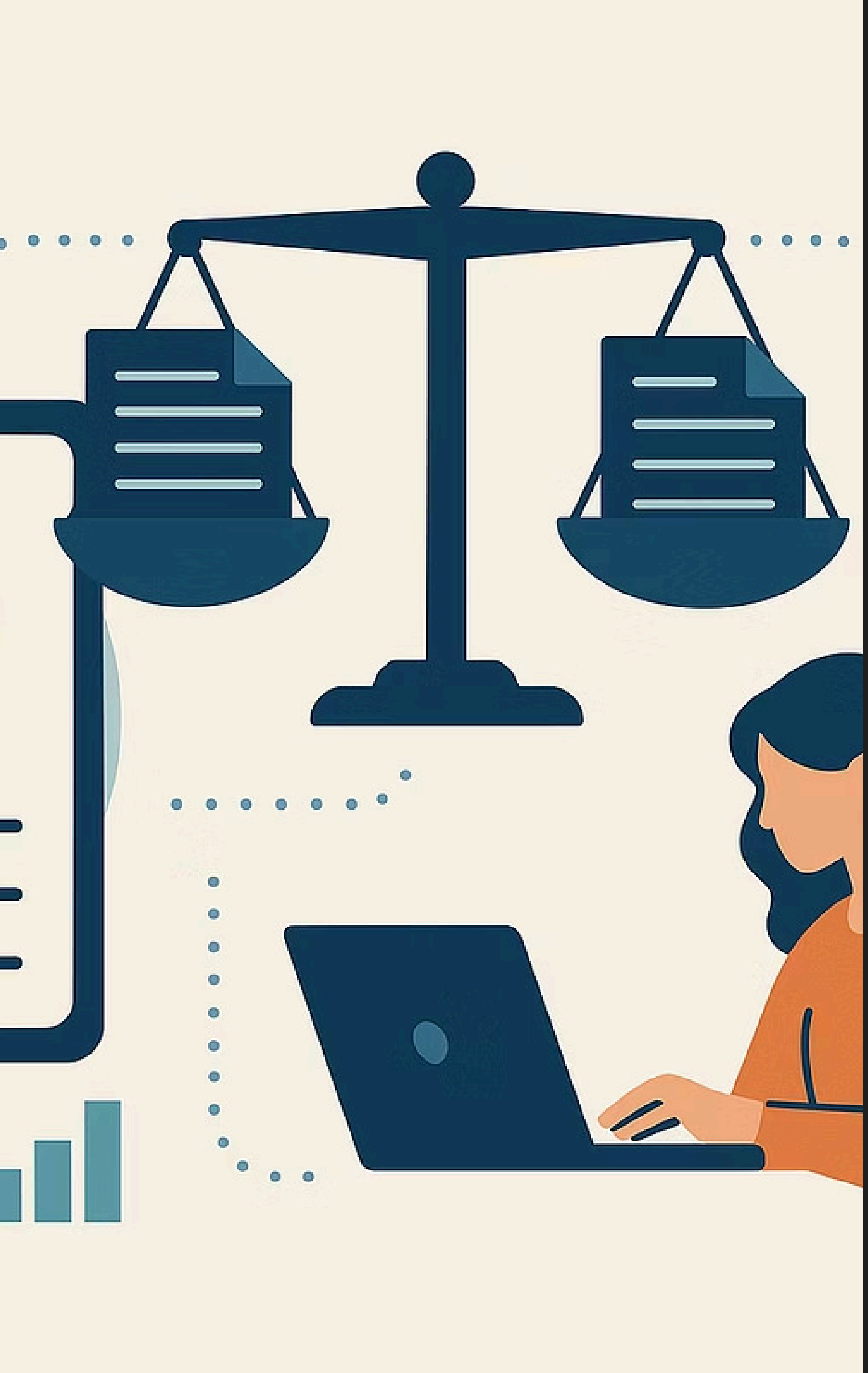
- Improved query performance through partition pruning
- Simplified maintenance operations
- Enhanced availability during maintenance windows
- Better resource utilisation

Effective Partitioning Approaches

Range Partitioning: Ideal for time-series data where queries typically access recent periods. Partition by date or timestamp columns to enable efficient archival and partition-wise operations.

Hash Partitioning: Distributes data evenly across partitions based on hash values, useful for distributing I/O load when no natural range exists.

List Partitioning: Organises data by discrete values such as region or category, enabling targeted queries to access only relevant partitions.



Workload Governance and Resource Management

Resource governance prevents contention, ensuring critical workloads get priority with Oracle and SQL Server's management tools.

Oracle Resource Manager

Allocates CPU, parallel execution, and session resources through consumer groups and plans, prioritizing workloads like OLTP or batch.

- CPU allocation
- Parallel execution limits
- Session controls

SQL Server Resource Governor

Manages CPU and memory via workload classification and resource pools, distinguishing between production, reporting, and admin tasks.

- CPU usage limits
- Memory controls
- I/O governance

Cross-Platform Performance Benchmarking

Understanding comparative performance across platforms is essential for workload placement and capacity planning in hybrid or multi-cloud environments.

1

Define Metrics

Establish key performance indicators (KPIs) like query response time, transactions per second (TPS), and resource utilization.

2

Create Test Workloads

Design synthetic workloads that mirror production patterns, including read/write ratios and query complexity.

3

Execute Tests

Run identical workloads on AWS RDS/Aurora and Azure SQL Database under controlled conditions.

4

Analyse Results

Compare performance metrics to identify strengths and trade-offs for architectural decisions.

AWS vs Azure: Diagnostic and Scaling Strengths

AWS Advantages

Performance Insights provides granular, real-time visibility into database load with minimal overhead. Aurora's serverless v2 offers seamless scaling with sub-second responsiveness.

Extensive Instance Variety: Wide range of EC2 and RDS instance types allows precise workload matching, from memory-optimised to compute-intensive configurations.

Azure Strengths

Intelligent Tuning leverages built-in machine learning to automatically detect and resolve performance issues, reducing manual intervention requirements.

Hyperscale Architecture: Azure SQL Database Hyperscale supports up to 100TB databases with rapid scaling and read replicas for exceptional read performance.

Cost Management in Cloud Databases

1

Right-Sizing Instances

Analyze resource consumption to right-size instances using optimizers like AWS Compute Optimizer or Azure Advisor.

2

Reserved Capacity

Commit to 1-3 year reserved instances for predictable workloads, saving 60-70% versus on-demand pricing.

3

Storage Optimisation

Implement lifecycle policies to move infrequently accessed data to lower-cost tiers like S3 Glacier or Azure Archive Storage.

4

Monitoring and Alerts

Configure cost anomaly detection and budget alerts to identify unexpected spending and ensure ongoing optimization.

Emerging Trends in Cloud Database Optimisation

Cloud database optimisation is advancing through intelligence, flexibility, proximity, and sustainability.

- **ML-Driven Tuning**

  Autonomous databases use ML to predict issues and auto-adjust configurations for optimal performance.

- **Serverless Databases**

  Serverless offerings automatically scale compute and storage, reducing costs and manual capacity planning.

- **Edge Computing**

  Edge deployments bring data closer to devices, ensuring ultra-low latency for real-time systems.

- **Sustainable Infrastructure**

  Optimising database efficiency reduces computational waste and carbon footprints, supported by cloud providers.

Your Optimisation Roadmap

01

Assess Current State

Conduct performance audits and establish baseline metrics.

02

Implement Quick Wins

Address missing indexes and inefficient queries for immediate benefits.

03

Establish Governance

Prevent resource contention with governance tools and proactive monitoring.

04

Plan Strategic Improvements

Plan longer-term initiatives like partitioning, prioritising by business impact.

05

Continuous Optimisation

Maintain ongoing optimisation with regular performance and cost reviews.

Key Takeaways

Combine proven techniques & cloud-native tools

Integrate traditional methods with cloud tools for comprehensive database optimization.

Optimise across all dimensions

Achieve high performance by optimizing query execution, resource utilization, and workload governance.

Understand platform strengths

Select AWS or Azure based on workload needs, leveraging their unique diagnostic and scaling capabilities.

Prepare for the future

Adapt to future trends like ML-driven tuning and serverless databases with adaptable architectures.

Thank You !