SQL Performance Tips
For Development Teams

Ben Johnston

Data Architect

I-Tech Solutions, Inc.

Ben Johnston

Data Architect
I-Tech Solutions, Inc.

I've worked with SQL Server and database systems for over 25 years, building warehouses, transactional systems, reporting environments, and system integrations. Consulting with medium and large enterprises, I specialize in database design, performance tuning, security, troubleshooting, and Azure upgrades.

https://www.red-gate.com/simple-talk/author/ben-johnston/

linkedin.com/in/bensql

https://github.com/BenSQL/SQLPerformance

Design goals

Indexes

Query design

Troubleshooting

Basics

Design

- Good design reduces problems later
- Understand good patterns and reasons for design decisions
- Setup SQL projects to facilitate deployments
- Non-breaking changes

Table / Database design

- Foundation of performance
- Table Design
  - Normalization
  - Natural key
  - Cluster
  - Foreign keys
  - Computed columns
- Optimize I/O and query performance
  - Basic design first, optimize for performance next

Table / database design

- Foreign keys
  - Not just a design consideration
  - Assists index decisions
  - Assures no implicit conversions
  - Assists report modeling
  - Archival processes
  - Static data masking in lower environments

Practical design considerations

- Indexing
  - Almost all tables with very few exceptions
- Clarity of intent
- Adding columns
  - All new columns should be added to the end of the table
    - Adding new columns to the middle (between / before any existing columns) can cause a significant performance impact during deployment.
  - In general, table definitions should be in 3rd normal form. If adding a column will alter this, it should be placed in a separate table.

Data types

- The column definition should reflect the data in the column
- Represents an actual business attribute
- Careful data type selection can improve performance
- Consistency – avoid implicit conversions
- Smallest data type that fits attribute for the foreseeable future

Data types

- Maximize bytes / row and ease of use
  - Float / real – very limited usage
  - Wide character set – only when business dictates
  - Strings that use only numerals – are still strings
  - Be aware of size boundaries
  - Developers should be able to defend the decision for any datatype choice
  - Avoid use of XML / JSON / sql_variant / BLOB

Design demo

Indexes

Index candidates

- Clustered primary key
- Unique constraints / business rules / natural keys
- JOIN criteria / foreign keys
- WHERE criteria
- Covering indexes
- Filtered indexes for specific scenarios
- Query plan / DMV recommendations

Index types

- Clustered
- Nonclustered
- XML
- JSON
- Spatial
- Columnstore

Index demo

Query Design

Producing a fast, efficient query

- Minimize network traffic
- Minimize physical I/O
- Minimize logical I/O
- Minimize CPU usage – lower priority
- Use set based operations

Query design

- Efficiency
  - Only use tables necessary for requirements
  - Avoid SELECT * - only return columns used
  - WHERE clause (limit data returned)
  - SET based operations
  - Cursors / loops / RBAR
  - Avoid functions in WHERE and JOIN
- JOIN
  - Avoid OR conditions
  - UNION is often more efficient
- SET NOCOUNT ON
- Triggers
- Query hints
- Implicit transactions versus explicit transactions

Query Demo

Troubleshooting Performance

Troubleshooting performance

- DMVs / DMFs
- Profiler / Extended Events
- Blocking
- Deadlocks
- Wait types
- Server metrics
- Stat data
- Index data

Troubleshooting performance - Queries

- Currently running
- Most expensive queries
  - CPU
  - Execution count
  - Time
  - I/O
- Errors

Query plans

- Capturing plans
- Basic plan inspection
  - Conversions
  - Recommended indexes
  - High-cost items
  - Scans versus seeks
  - Indexes used / not used
- Query Store
  - Settings
  - Forcing a plan

Troubleshooting Demo

Mitigations

- Query hints
- Indexed views
- Reporting tables
- Read-only replicas

Code smells / Anti-patterns

- Entity Attribute Value (EAV) patterns
- Excessive logic in stored procedures
- Direct access to production data
  - Reporting
  - Ad-hoc queries
- Manual deployments

Coding standards

- Not directly tied to performance
- Helps ensure performance standards are followed
- Consistency helps
  - Troubleshooting
  - Code reviews
  - Maintenance

Admin Settings - FYI

Server level

- blocked process threshold (s)
- max degree of parallelism
- cost threshold for parallelism
- fill factor (%)
- max server memory (MB)
- min server memory (MB)

Database level

- MAXDOP
- OPTIMIZE_FOR_AD_HOC_WORKLOADS
- PARAMETER_SENSITIVE_PLAN_OPTIMIZATION

Final thoughts

- Don’t compromise security for performance or convenience
- Simplicity and maintainability need to be part of the design
  - Clever code is hard to maintain
  - Simple code often performs better
- Archive / partition data
- Data consistency
- Hardware

Additional Resources

- https://www.red-gate.com/simple-talk/featured/sql-server-execution-plans-third-edition-by-grant-fritchey/
- https://www.sqlskills.com/blogs/paul/wait-statistics-or-please-tell-me-where-it-hurts/
- https://learn.microsoft.com/en-us/sql/t-sql/data-types/data-types-transact-sql?view=sql-server-ver17
- https://learn.microsoft.com/en-us/sql/sql-server/maximum-capacity-specifications-for-sql-server?view=sql-server-ver17
- https://learn.microsoft.com/en-us/sql/relational-databases/performance/manage-the-query-store?view=sql-server-ver17&tabs=ssms

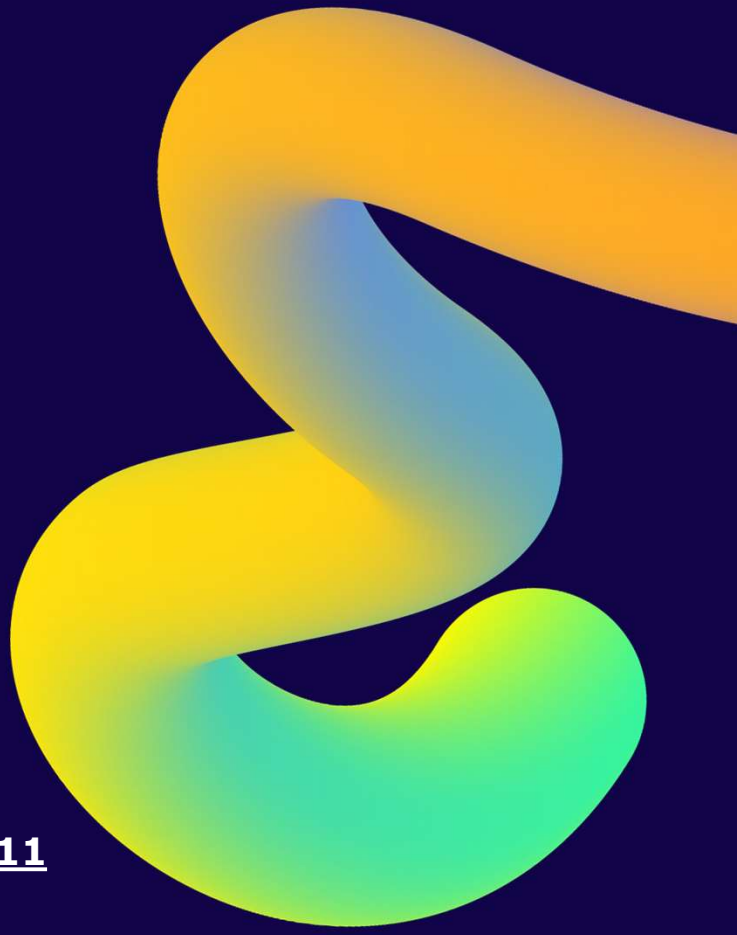
Your feedback is important to us

Evaluate this session at:

www.PASSDataCommunitySummit.com/evaluation

Thank you

I appreciate your attendance



Ben Johnston

https://www.red-gate.com/simple-talk/author/ben-johnston/

https://www.linkedin.com/in/ben-johnston-41609011

https://github.com/BenSQL/SQLPerformance