SQLintersection

Eliminating Low Hanging Fruit

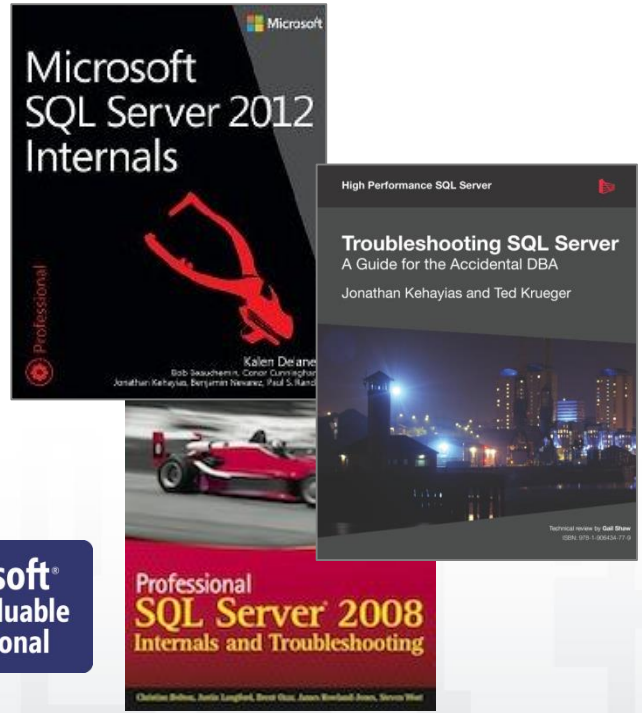
Jonathan Kehayias
Principal Consultant
SQLskills.com

SQL *intersection*

Jonathan Kehayias

- **Consultant/Trainer/Speaker**
- **Principal Consultant, SQLskills.com**
  - Email: Jonathan@SQLskills.com
  - Blog: http://www.SQLskills.com/blogs/jonathan
  - Twitter: @SQLPoolBoy
- **SQL Server MVP since October 2008**
- **Microsoft Certified Master: SQL Server 2008**

SQLskills
immerse yourself in sql server



Microsoft CERTIFIED Master | SQL Server® 2008

MVP Microsoft® Most Valuable Professional

SQL intersection

Overview

- Row-By-Agonizing-Row processing
- Develop and test against realistic scale
- Profile your workload during development
- Use appropriate data types
- Sargability matters

SQL intersection

Row-By-Agonizing-Row Processing

- **Certain constructs force SQL Server into RBAR (row-by-agonizing-row) processing of results**
- **Well known:**
  - WHILE loops
  - Cursors
- **Less well known:**
  - Scalar user defined functions
  - Correlated subqueries

SQL intersection

Cursors and Loops

- **Characteristics**
  - Explicit cursor declaration
  - WHILE loop
  - SqlDataReader in application
- **Problems**
  - Row based processing over Set Based
- **Replace with**
  - Appropriate set based operation
  - Move looping code into SQLCLR or Middle Tier
  - Consume and dispose of SqlDataReader as quickly as possible

SQL *intersection*

Correlated Sub-Queries

- **Characteristics**
  - Refer to the outer query in the inner query in SELECT statement
  - SELECT Statement used as column value in UPDATE
- **Problems**
  - May cause row-by-row processing to occur
  - Performance decreases exponentially as row count increases
- **Replace with**
  - Table Join
  - Derived Table Join
  - Cross Joined Table Valued Function

SQL *intersection*

Scalar User Defined Functions

- **Characteristics**
  - Encapsulate common code blocks/business logic in a single call.
  - If columns are passed as parameters it is not inline
- **Problems**
  - Cause row-by-row processing to occur
  - Performance decreases exponentially with data access
- **Replace with**
  - Inline expressions
  - Derived Table Join
  - Cross Joined Inline Table Valued Function

SQL *intersection*

Develop and Test Against Realistic Scale

- **Development databases often do not contain realistic datasets which can hide/mask potential performance problems**
  - Key Lookups on small data sets may become index scans on larger data sets
  - Missing index impacts may be hidden by data residing in memory for small data sets
  - RBAR problems are often hidden until data sizes scale up
- **Testing a single execution in isolation is not load testing**
  - Testing needs to be performed at scale through load generation to measure accumulated effects
  - Only testing at scale can identify *"death by 1000 cuts"* problems

SQL *intersection*

Profile Your Workload During Development

- **Learn to use Extended Events (2012+) or SQL Trace to profile your workload during testing**
- **Know the important events to watch for during development**
  - Statement/Batch/RPC completed events
  - SP completed/Module End events – (procedure/trigger/function executions)
  - Execution warnings (sort, hash, missing join predicate)
- **Profiling during development can uncover nasty RBAR issues and performance effecting side effects of trigger executions**
- **Be aware of *"observer overheads"* but not typically a problem with development/test workloads**

SQL intersection

Use Appropriate Data Types

- Understand the storage costs and implications of data types during schema design – especially for keys

Data type	Range	Storage
bigint	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	8 Bytes
int	-2,147,483,648 to 2,147,483,647	4 Bytes
smallint	-32,768 to 32,767	2 Bytes
tinyint	0 to 255	1 Byte

SQL intersection

Use Appropriate Data Types (2)

- **Consider precision requirements for dates and times**
  - DATETIME2 = 6, 7, or 8 bytes with nanosecond precision
    - 6 bytes for precisions less than 3
    - 7 bytes for precisions 3 and 4
    - All other precisions require 8 bytes
  - DATETIME = 8 bytes with fractions of second precision
    - Fraction of seconds rounded to .000, .003, or .007 seconds
  - SMALLDATETIME = 4 bytes with minute precision
  - DATE = 3 bytes
- **Don’t store dates or times as CHAR, VARCHAR, NCHAR, or NVARCHAR**

SQL intersection

Sargability Matters

- **A query is sargable (Search ARGument ABLE) if an index seek can be used to speed up the execution of the query**
- **Anti-patterns to sargable expressions include:**
  - Functions in the WHERE clause
  - Implicit/Explicit data type conversions on a column
  - Leading wildcard expressions with LIKE '%<SearchTerm>'
  - Catch all queries and search procedures

SQL *intersection*

Functions on WHERE Clause Columns

- **Characteristics**
  - Used to change the data stored to match criteria being checked
  - Conversion of data to a different type
- **Problems**
  - Causes Table/Index Scan over Seek
- **Replace with**
  - Appropriate Table Design to support business needs
  - Indexed/Persisted Computed Column
  - Indexed View
  - Other coding paradigm to

SQL intersection

Implicit/Explicit Column Conversions

- **Characteristics**
  - Column data type is of lower precedence than filtering parameter / joining column data type
  - Common in LINQ to SQL/EF and other ORMs
- **Problems**
  - Causes Table/Index Scan over Seek
- **Replace with**
  - Higher precedence column data type
  - Matching data type for filtering parameter

SQL *intersection*

Catch-All Search Queries

- **Characteristics**
  - Used to search across multiple columns using parameters
  - Not all parameters require input values
  - WHERE clause similar to (@Param1 IS NULL OR Column1 = @Param1)
- **Problems**
  - No optimized execution plan
  - Causes Table/Index Scan over Seek
- **Replace with**
  - Separate search procedures for different parameters passed
  - Parameterized Dynamic SQL

SQL *intersection*

Review

- Row-By-Agonizing-Row processing
- Develop and test against realistic scale
- Profile your workload during development
- Use appropriate data types
- Sargability matters

SQL intersection

Questions?

SQLintersection

Don’t forget to complete an online evaluation!

Eliminating Low Hanging Fruit

Your evaluation helps organizers build better conferences and helps speakers improve their sessions.

SQL *intersection*

Thank you!

Our Spring Show Lineup Speaks for Itself!

May 21-24, 2017 in Orlando, Florida

- **40 SQL Sessions, 3 SQL keynotes, and 7 workshops:**
  - 2 workshop on Saturday, May 20
    - *PowerShell for the DBA from 0-60 in a Day* with Ben Miller
    - *SQL Server 2014 and 2016 New Features and Capabilities* with Tim Chapman and David Pless
  - 2 workshops on Sunday, May 21
    - *Performance Troubleshooting using Waits and Latches* with Paul Randal
    - *Azure for the SQL Server DBA* with Tim Radney
  - 3 workshops on Thursday, May 25
    - *Finding and Fixing Performance Problems in SQL Server* with Erin Stellato and Jonathan Kehayias
    - *Advanced SQL Server High Availability* with Brent Ozar
    - *Cortana Intelligence Suite-Microsoft R for Architects* with Buck Woody

Show focuses on versions 2008 and higher with special sessions on new features in SQL 2014 / 2016 plus sessions about Azure and vNext!

SQL after Dark

Fantastic evening events interact + meet the speakers get YOUR questions answered!

- **Industry-experts and Microsoft speakers**

Sessions on performance tuning, troubleshooting, coding / development, query tuning, architecture, new features + vNext, plus much more – learn real-world solutions & bring back immediate ROI

SQL intersection