

SQLintersection

Five Things I Wish Developers Knew about SQL Server

Jonathan Kehayias
Principal Consultant
SQLskills.com



SQL
intersection



Jonathan Kehayias

Principal Consultant, SQLskills



Jonathan@SQLskills.com



[@SQLPoolBoy](https://twitter.com/SQLPoolBoy)



www.sqlskills.com/blogs/jonathan

Trainer/Speaker

In addition to consulting, I teach content for our IE0: Accidental DBA course, IEAG: Clustering and Availability Groups course, and our IEPTO2: Performance Tuning and Optimization course

Data Platform MVP

I have been thankfully recognized as an MVP by Microsoft since 2008

Author

Microsoft SQL Server 2012 Internals

Professional SQL Server 2008 Internals and Troubleshooting

Troubleshooting SQL Server: A Guide for the Accidental DBA

Reminder: Intersect with Speakers and Attendees

- **Tweet tips and tricks that you learn and follow tweets posted by your peers!**
 - Follow: #SQLIntersection and/or #DEVIntersection
- **Join us – Wednesday Evening – for SQLafterDark**
 - Doors open at **7:00 pm**
 - Trivia game starts at **7:30 pm**
 - Winning team receives something fun!*
 - Raffle at the end of the night
 - Lots of great items to win including a seat in a SQLskills Immersion Event!*
 - The first round of drinks is sponsored by SentryOne and SQLskills



Overview

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

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

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

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

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

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

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

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

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**

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

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

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

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

Review

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

Questions?



Don't forget to complete an online evaluation on EventBoard!

Five Things I Wish Developers Knew about SQL Server

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



SQL
intersection

Thank you!