

SQLskills Online Immersion Event

IEQuery: Immersion Event on Fixing Slow Queries,
Inefficient Code, and Caching/Statistics Problems

Part I: Capturing Query Information and Analyzing Plans

Erin Stellato
Erin@SQLskills.com
@ErinStellato



Erin Stellato

Principal Consultant, SQLskills



Erin@SQLskills.com



@erinstellato



www.sqlskills.com/blogs/erin

Trainer/Speaker

In addition to consulting, I teach content for our IEQ: Accidental DBA course, and our IEPTO2: Performance Tuning and Optimization course

PASS Volunteer

I was a member of the PASS Nomination Committee this past year, have previously served on the board of my local user group (ONSSUG) and supported the Performance Virtual Chapter

Data Platform MVP

I have been fortunate to be recognized as an MVP by Microsoft since 2012



Schedule

- **Tuesday:** Erin Stellato
 - Capturing Query Information and Analyzing Plans
- **Wednesday:** Jonathan Kehayias
 - Removing Anti-Patterns in Transact-SQL
- **Thursday:** Kimberly Tripp
 - How to Differentiate Caching/Statistics problems and SOLVE THEM!



© SQLskills, All rights reserved.
<http://www.SQLskills.com>

Schedule – Tuesday, Wednesday & Thursday

- **Lecture 1:** 90 minutes from 10:00am to 11:30am PDT
 - Please use the WebEx chat for questions and responses **during** the lecture
- **Open Q&A 1:** 30 minutes from 11:30am to 12:00 noon PDT
 - I'll address any unanswered questions from chat
 - May have time for "open mic" questions
- **Mandatory break:** 30 minutes from 12:00 noon until 12:30pm PDT
 - Get up, stretch, get something to eat/drink, recharge!

Time conversions:
10:00am PDT
= 1:00pm EDT
= 5:00pm UTC



© SQLskills, All rights reserved.
<http://www.SQLskills.com>

Schedule – Tuesday, Wednesday & Thursday

- **Lecture 2:** 90 minutes from 12:30pm to 2:00pm PDT
 - Again, please use WebEx chat for questions/responses **during** the lecture
- **Open Q&A 2:** Up to 60 minutes from 2:00pm to 3:00pm PDT
 - Address any unanswered questions from chat
 - Time for “open mic” questions

Time conversions:
10:00am PDT
= 1:00pm EDT
= 5:00pm UTC



© SQLskills, All rights reserved.
<http://www.SQLskills.com>

Course Resources & Pluralsight Access

- Paul (Paul@sqlskills.com) will send you an email for a FREE 30-day access code to ALL SQLskills online training class on Pluralsight
 - No strings attached, no credit card needed
 - If you do not receive an email by end of day Thursday, email Paul
- Course resources will be sent upon completion (Friday)
 - I will update slides and add notes/resources based on questions
 - The resource zip will be password protected. Please do not distribute it; it is for course attendees only. The password will be sent via email.
- Email Paul (Paul@sqlskills.com) if you have problems finding or opening the resource file



© SQLskills, All rights reserved.
<http://www.SQLskills.com>

Course Resources

- Course content (this deck)
- Demos
 - All instructor-led demos
 - All reference scripts
- Reference links (in the zip file)
- Online recordings for the sessions

Please do not redistribute: These resources are for IEQuery attendees only. We ask that you please respect our IP and time in creating these resources and do not share/forward/sell on eBay



Overview

- Query Metrics
- Capturing and comparing query plans
- Information in a plan
- Plan analysis

Query Metrics



Sources for Data Related to Query Performance

SSMS

DMVs

Extended
Events/Trace

Query
Store

PerfMon

Third-party
tools*

Purpose of a Baseline

- Provide a starting point for comparison of additional data over time
 - Often represents the “normal” or typical state of the environment
 - Helps you understand where the system is today
- Baseline data is invaluable during a performance crisis
- Can also be used to identify usage patterns and trending, and can be extremely helpful for capacity planning
- Used to measure the impact of changes

Benchmarking

- A benchmark measures performance using a *specific set of indicators* to determine the performance level in a way that can be compared to other systems, business requirements, or previous benchmarks
- It is a **standard** for comparison
- May be established to determine the capacity limits of a system
 - Maximum number of concurrent connections
 - Maximum number of transactions/batches per second
 - Use to forecast replacement/upgrade requirements before exceeding limits

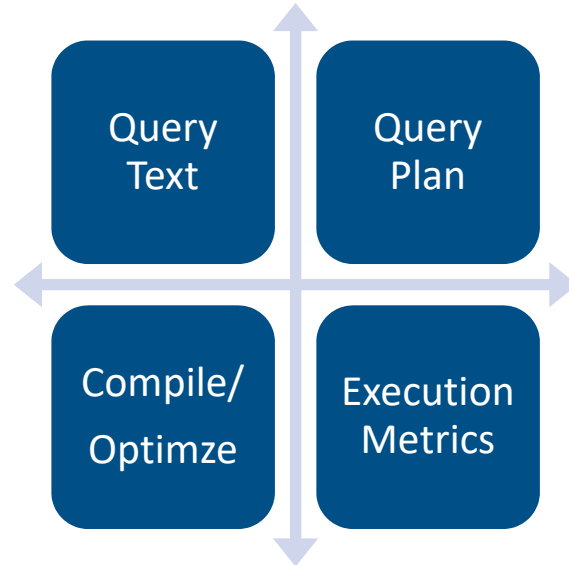
Benchmark vs. Baseline

- Use benchmarks and baselines to reach a target or specific goal
 - "This stored procedure used to run in 200 ms." (*this is your benchmark*)
 - "The SP now takes 3 seconds." (*this is your baseline*)
 - Compare the baseline to the benchmark
 - Improve the current value in steps (*this is tuning*)

How do you baseline query performance?

- Use a third-party monitoring tool
- Snapshot data from multiple DMVs
- Use sp_whoisactive to snapshot data
- Capture data using Extended Events
 - Trace for SQL Server 2008R2 and earlier
- OpenQueryStore
 - SQL Server 2008 – SQL Server 2014
- Query Store
 - SQL Server 2016+

Specific Data of Interest



Data source will depend on SQL Server version and the problem you're trying to solve

Query Execution Data

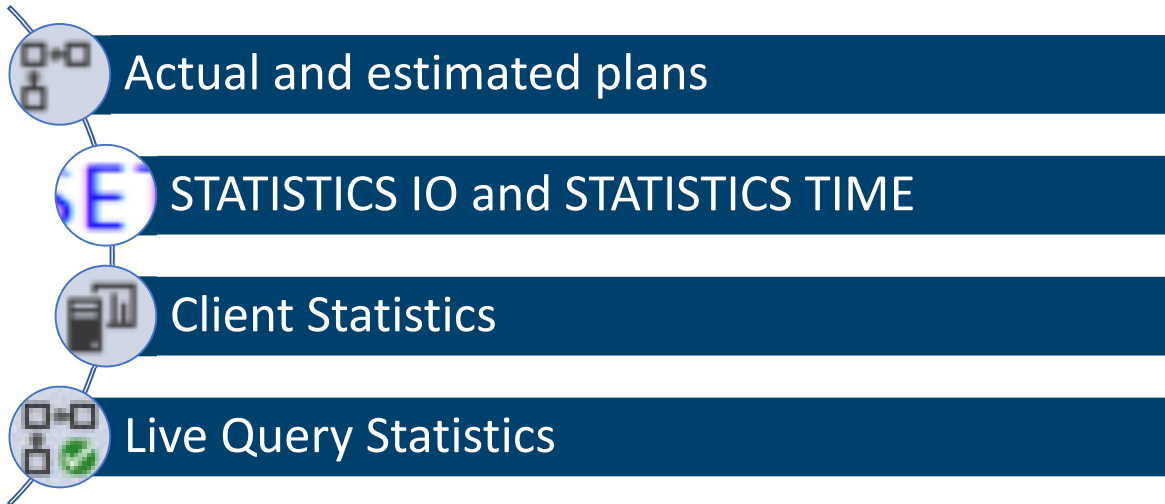
Individual Metrics

- SSMS
- Extended Events
- Trace

Aggregated Metrics

- DMVs
- Query Store
- PerfMon

SSMS



DMVs

- `sys.dm_exec_sql_text`
 - Provided a `sql_handle`, returns text of the batch
- `sys.dm_exec_cached_plans`
 - One row for each query plan currently in memory
- `sys.dm_exec_query_plan`
 - Provided a `plan_handle`, it returns the plan in XML format
- `sys.dm_exec_text_query_plan`
 - Output is in text format
- `sys.dm_exec_query_stats`
 - Aggregate performance statistics for cached plans
- `sys.dm_exec_procedure_stats`
 - Aggregate performance statistics for cached stored procedures
- `sys.dm_exec_function_stats` (added in 2016)
 - Aggregate performance statistics for cached functions

Extended Events

- `sql_batch_completed`
 - Completion of T-SQL batch
 - `rpc_completed`
 - Completion of a remote procedure call
 - `sp_statement_completed`
 - Completion of T-SQL statement within a stored procedure
 - `sql_statement_completed`
 - Completion of T-SQL statement
 - `query_post_compilation_showplan`
 - `query_pre_execution_showplan`
 - `query_post_execution_showplan`
- } **Not recommended**

Query Store

- Described as a flight data recorder
- Captures information about query execution
 - Query text
 - Query plan
 - Compilation time
 - Last execution time
 - Duration, CPU, logical reads, physical reads, writes – aggregated across a time interval

Data Captured by Query Store

Plan Store

- Compile time and duration
- Last execution time
- Query text
- Query plan

Runtime Stats Store

- Execution counts
- Duration
- CPU
- Logical reads
- Physical reads
- Write
- Memory use
- DOP
- Log bytes/used
- tempdb

Wait Stats Store

- Wait statistics per plan

SQL Server
2017 and
Azure SQL
Database

Capturing and Comparing Query Plans

Query Plans

- The majority of plans are stored in cache
- This means that the plan cache can be queried
- The plan retrieved from cache is the one that has been used
 - This is often referred to as the *estimated* plan
 - When you retrieve the plan from cache, you only see estimates (number of rows, number of executions)
- The *actual plan* can only be retrieved during query execution
 - The “actual” query plan is *typically* the same as the plan that exists in cache, and it *also includes* actuals (actual number of rows, actual number of executions)
 - Used to determine if there is disparity between estimates and actuals



© SQLskills, All rights reserved.
<http://www.SQLskills.com>

Finding a Query Plan

Estimated Plan

- sys.dm_exec_query_plan
- sys.dm_exec_text_query_plan
- SSMS
 - SET SHOWPLAN_XML
 - Graphical Showplan
- Query Store

Actual Plan

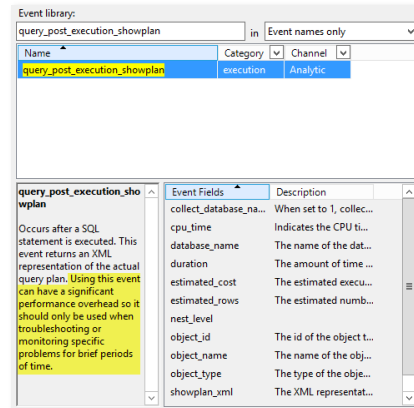
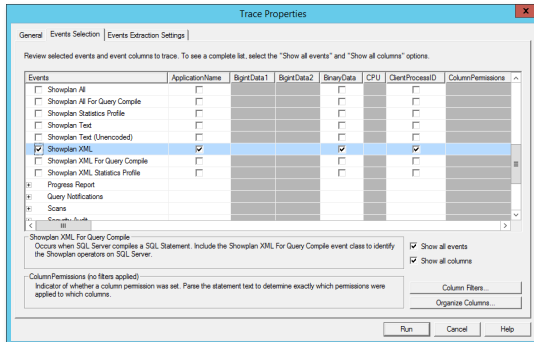
- SSMS
 - SET STATISTICS XML
 - Graphical Showplan
- Trace
- Extended Events



© SQLskills, All rights reserved.
<http://www.SQLskills.com>

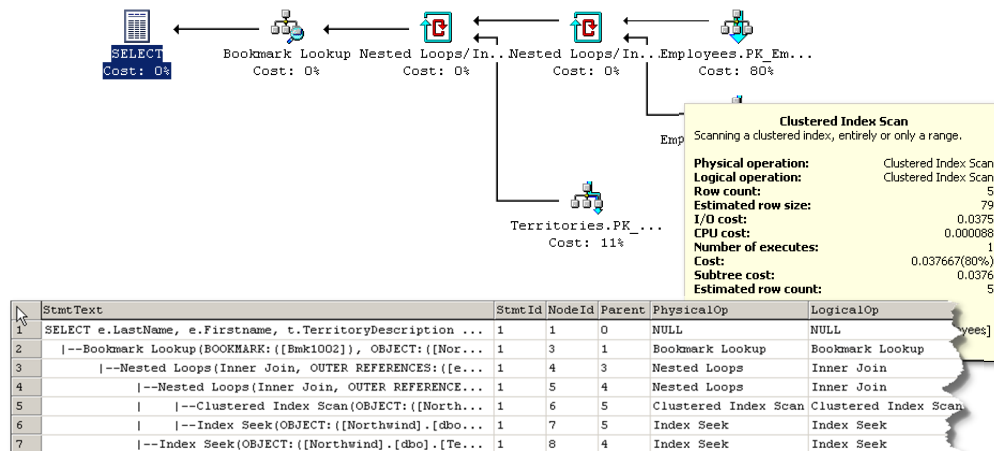
Capturing Plans with Trace or Extended Events

- Plans can be captured with both SQL Trace and Extended Events
 - Neither is a recommend option



Remember Plans in SQL 2000?

Query 1: Query cost (relative to the batch): 100.00%
Query text: `SELECT e.LastName, e.Firstname, t.TerritoryDescription FROM dbo.Employees e JOIN (`



Plans in XML vs. Text

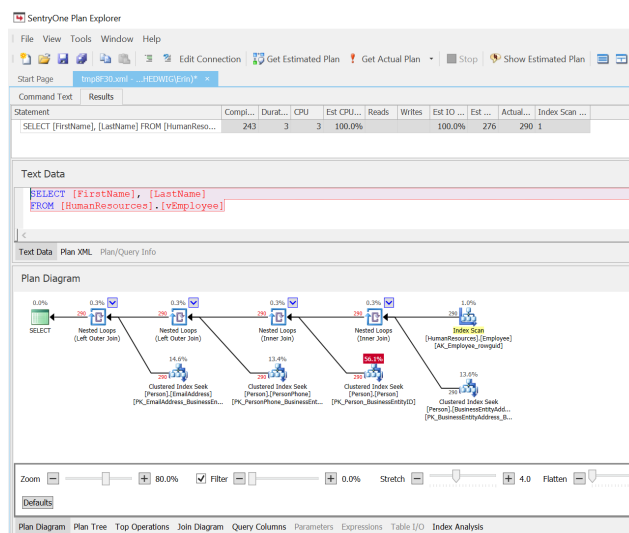
- XML showplan formats may seem more unwieldy and verbose than their text-based / tabular counterparts, but there are some key advantages:
 - Can be processed via XPath and XQuery
 - Easier to add attributes/elements from version to version
 - Save with .sqlplan extension to open in SSMS
- XML schema describes the structure of an XML document
 - <http://schemas.microsoft.com/sqlserver/2004/07/showplan/>
 - Much of what you can find in the graphical plan you can see in XML, so the primary benefit is in parsing and finding key elements and attributes programmatically



© SQLskills, All rights reserved.
<http://www.SQLskills.com>

Viewing Plans in SentryOne Plan Explorer

- Free, third-party tool
- Good for all sizes of plans
- Pulls out key information in a compact way
- Can open a .sqlplan file, or copy in the XML



© SQLskills, All rights reserved.
<http://www.SQLskills.com>

Plan Comparison

- Introduced in SSMS 17.0
 - Compare against a saved plan
 - Compare against a generated, actual plan
 - Compare two plans in Query Store
- Differences in operator properties are highlighted
- Includes additional details in the Scenarios tab when large differences in estimates and actuals exist

Demo: Capturing and comparing query performance data

Information in a plan



Plans

Information Included

- Tables/indexes access
- Operators
 - Scans vs. seeks
 - Join type
- Estimated cost of each operation
- Number of rows expected
- More information continues to be added
 - Duration, I/O, wait statistics

Information *NOT* Included

- Locks
- Who executed the query
- Application/host name
- Wait statistics*

Query Parameters

- The plan contains valuable information on compiled vs. runtime value
 - ParameterCompiledValue
 - ParameterRuntimeValue
- New attribute, ParameterDataType, added in 2016 SP1
 - <https://support.microsoft.com/en-us/help/3190761/>
- Some queries are very sensitive to different parameters
- Test different values using recompile (or a cold cache) to see if different plans are created for different values

Trace Flags Enabled

- The TraceFlags element is available in SQL Server 2014 SP2 which lists the trace flags enabled (global or session) at the time of query compilation (IsCompileTime = true) and execution (IsCompileTime = false)
 - <https://support.microsoft.com/en-us/kb/3170115>

```
<TraceFlags IsCompileTime="true">  
  <TraceFlag Value="4199" Scope="Global" />  
</TraceFlags>  
<TraceFlags IsCompileTime="false">  
  <TraceFlag Value="4199" Scope="Global" />  
</TraceFlags>
```

```
<TraceFlags IsCompileTime="true">  
  <TraceFlag Value="8649" Scope="Session" />  
</TraceFlags>
```

Operators

- Operators are building blocks for a query, each one has a specific functionality
 - Some operators access data (scan, seek), others perform operations such as aggregations or joins
 - There is a one-to-many mapping of logical to physical
 - For example, an INNER JOIN could be implemented as a Loop, Hash, or Merge join
- Specific operators are not "good" or "bad"
 - Some can consume more resources than others or have overhead of which you should be aware
 - Some are more appropriate in given contexts
- SQL Server 2017 has 100+ operators
- Operators may also be referred to as iterators



© SQLskills, All rights reserved.
<http://www.SQLskills.com>

Query Optimization and Cost

- Each operation in the query tree is given a cost, and those are totaled to determine total cost for each possible plan
- Cost-based optimization looks at statistics and the size of the data that would be retrieved to estimate I/O
- Cost used to equate to elapsed time in seconds required to run on a specific Microsoft employee's machine (from SQL Server 7 era)



© SQLskills, All rights reserved.
<http://www.SQLskills.com>

Operator Cost

- “Cost” today in the context of query plans is a unit-less measure
 - Cost <> time
 - Cost <> CPU
- Cost is used for relative comparison across plan operators and between plans
- “cost threshold for parallelism” option refers to this very same cost (default 5)
 - The default value of 5 is very low considering today’s CPU architecture
- Estimated costs **remain estimates**, even for actual plans

Cost Sensitivity to Row and Page Count

- Estimated costs are sensitive to specific pieces of data
 - e.g. the number of anticipated data pages and rows
- Consider the Clustered Index Scan operator
 - Estimated **I/O** cost of a Clustered Index Scan shows sensitivity to anticipated data **page** counts, but not row counts.
 - Think about the estimated I/O cost impact that high fragmentation may have (same number of rows across many partially-filled data pages)
 - Estimated **CPU** cost of a Clustered Index Scan sensitivity to **row** counts, but not data page counts.
 - Regarding row counts, even for narrow rows, consider the estimated CPU cost impact that high row counts may have.

Time and Wait Stats

- In SQL Server 2016 SP1, time and wait type information is added to the plan
- Within the QueryTimeStats attribute you can see CpuTime and ElapsedTime
 - Also get UDF execution statistics in SQL 2016 SP2 and SQL 2017 CU3
- The top 10 wait types are also included within the WaitStats element
 - <https://support.microsoft.com/en-us/help/3201552>

```
<QueryTimeStats CpuTime="4" ElapsedTime="4" />
  <WaitStats>
    <Wait WaitType="PAGEIOLATCH_SH" WaitTimeMs="4" WaitCount="31" />
    <Wait WaitType="SOS_SCHEDULER_YIELD" WaitTimeMs="5" WaitCount="843" />
    <Wait WaitType="ASYNC_NETWORK_IO" WaitTimeMs="72" WaitCount="6" />
    <Wait WaitType="MEMORY_ALLOCATION_EXT" WaitTimeMs="496" WaitCount="535073" />
  </WaitStats>
```



© SQLskills, All rights reserved.
<http://www.SQLskills.com>

Operator Execution Statistics

- In SQL Server 2014 SP2 and SQL Server 2016+ per-operator query execution statistics are available in the XML
 - RunTimeCountersPerThread
 - Includes CPU time, elapsed time, IO information
 - <https://support.microsoft.com/en-us/kb/3170113>

```
<RunTimeInformation>
  <RunTimeCountersPerThread Thread="0" ActualRebids="1" ActualRewinds="0" ActualRows="18097" ActualEndOfScans="1"
    ActualExecutions="1" ActualElapsedDms="11" ActualCPUs="11" ActualScans="0" ActualLogicalReads="0"
    ActualPhysicalReads="0" ActualReadAheads="0" ActualLobLogicalReads="0" ActualLobPhysicalReads="0"
    ActualLobReadAheads="0" />
</RunTimeInformation>

<RunTimeInformation>
  <RunTimeCountersPerThread Thread="4" ActualRows="5574" ActualRowsRead="5574" Batches="0" ActualEndOfScans="1" ActualExecut
    <RunTimeCountersPerThread Thread="3" ActualRows="7401" ActualRowsRead="7401" Batches="0" ActualEndOfScans="1" ActualExecut
    <RunTimeCountersPerThread Thread="1" ActualRows="4312" ActualRowsRead="4312" Batches="0" ActualEndOfScans="1" ActualExecut
    <RunTimeCountersPerThread Thread="2" ActualRows="14178" ActualRowsRead="14178" Batches="0" ActualEndOfScans="1" ActualExec
    <RunTimeCountersPerThread Thread="0" ActualRows="0" ActualEndOfScans="0" ActualExecutions="0" ActualElapsedDms="0" ActualCP
  </RunTimeInformation>
```



© SQLskills, All rights reserved.
<http://www.SQLskills.com>

Live Query Statistics and Query Profiling

- Introduced in SQL Server 2016 (SSMS 13.x), and can work with SQL Server 2014 through the current release
- Using this option enables the statistics profile infrastructure for the *current session*
 - This infrastructure can be enabled globally using TF 7412 or with an Extended Events session that captures the query_thread_profile event
 - **This introduces performance overhead**
 - In SQL Server 2016 SP2 CU3 and SQL Server 2017 CU11, you can enable the profile infrastructure globally with the query_plan_profile event, but metrics will only be collected for queries that include the QUERY_PLAN_PROFILE hint
 - **Note this does not include the statement**



© SQLskills. All rights reserved.
<http://www.SQLskills.com>

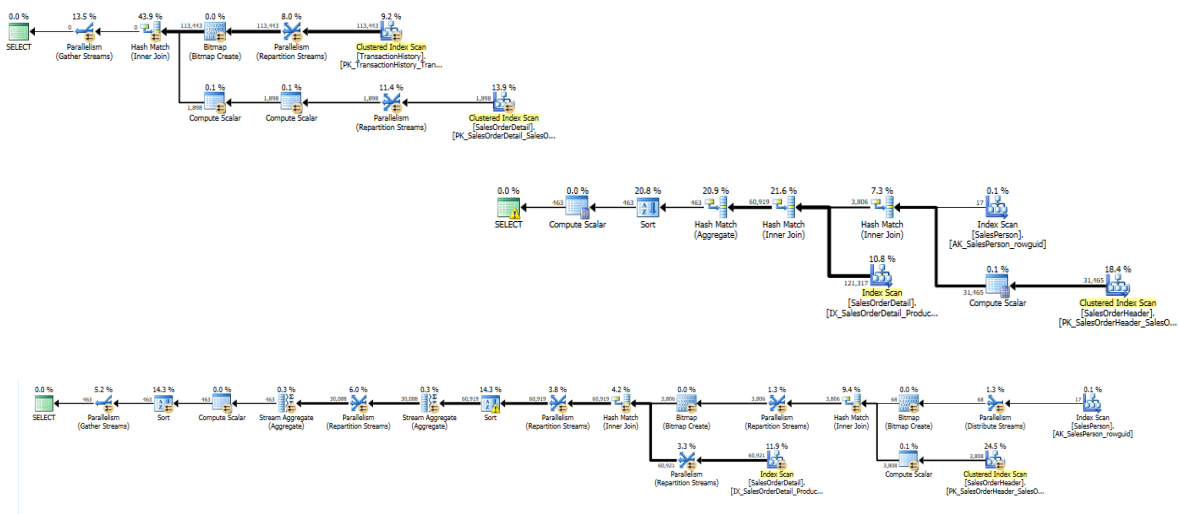
Demo: Essential information in a plan



Plan Analysis



Where do you start with a plan?

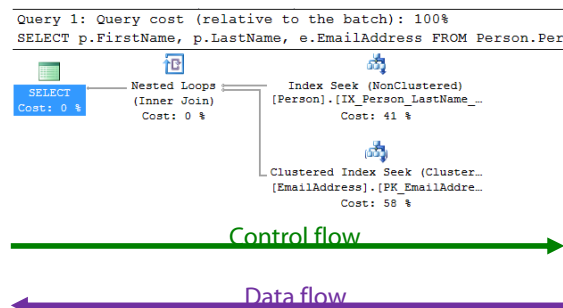


Where “should” you start with a plan?

- There is no one “right” answer
- Don’t believe statements that specific operators or behaviors translate to an absolute problem
 - We’ll discuss things to watch for, but see them as areas of investigation
 - Some might be red herrings instead of red flags
- The emphasis is on patterns you are more likely to see
- You have to practice reading and understanding plans to determine where you can improve a query, and this takes time

Reading Plans

- An operator reads rows from a leaf-level data source OR from child operators and return rows to the parent
- Control flow starts at the root (left-to-right)
- Data flow starts at the leaf level (right-to-left)



Data Access Operators

- Retrieve all rows from a table
- Indicates a heap table
- Is it a red flag?

Table Scan



- Retrieve rows from a clustered index
- Is it a red flag?

Clustered Index Scan



- Retrieve rows from a non-clustered index
- Is it a red flag?

Nonclustered Index Scan



- Retrieve rows based on a SEEK predicate from a clustered index
- Is it always a good thing?

Clustered Index Seek



- Retrieve rows based on a SEEK predicate from a non-clustered index
- Is it always a good thing?

Nonclustered Index Seek



- Retrieve rows from a column-store index

Columnstore Index Scan



Predicates

- Seek Predicate
 - Used in actual index seek operation
 - Leveraging index keys
- Predicate (Residual Predicate)
 - Search condition that isn't SARGable – so it remains as an extra predicate
 - For Merge Join: check Graphical Showplan Properties for "Residual" value
 - For Hash Match: check "Probe Residual" value in plan itself (tooltip over operator)

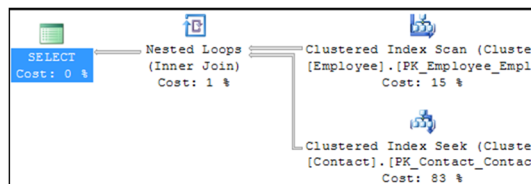
Filter



- Predicates can be evaluated within operators that read data from table/indexes
- Query Optimizer aims (when possible) to “push” filter down the tree (leaf level) to reduce rows moved
- If a predicate is high in cost or complexity a separate Filter operator may be used
- When you see these, take note of where they are happening
 - Late in the data flow can translate to higher overhead as the operators pull data

Seek Predicate with No Residual

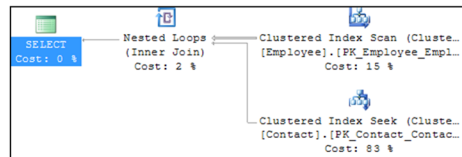
```
SELECT
[e].[EmployeeID],
[c].[Title],
[c].[FirstName],
[c].[MiddleName],
[c].[LastName],
[c].[Suffix]
FROM [HumanResources].[Employee] [e]
INNER JOIN [Person].[Contact] [c]
ON [c].[ContactID] = [e].[ContactID];
```



Clustered Index Seek (Clustered)	
Scanning a particular range of rows from a clustered index.	
Physical Operation	Clustered Index Seek
Logical Operation	Clustered Index Seek
Actual Number of Rows	290
Estimated I/O Cost	0.003125
Estimated CPU Cost	0.0001581
Number of Executions	290
Estimated Number of Executions	290
Estimated Operator Cost	0.071616 (83%)
Estimated Subtree Cost	0.071616
Estimated Number of Rows	1
Estimated Row Size	187 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	True
Node ID	3
Object	
[AdventureWorks].[Person].[Contact].	
[PK_Contact_ContactID] [c]	
Output List	
[AdventureWorks].[Person].[Contact].Title,	
[AdventureWorks].[Person].[Contact].FirstName,	
[AdventureWorks].[Person].[Contact].MiddleName,	
[AdventureWorks].[Person].[Contact].LastName,	
[AdventureWorks].[Person].[Contact].Suffix	
Seek Predicates	
Seek Keys[1]: Prefix: [AdventureWorks].[Person].	
[Contact].ContactID = Scalar Operator([AdventureWorks].	
[HumanResources].[Employee].[ContactID] as [e].	
[ContactID])	

Seek Predicate *with* a Residual

```
SELECT
[e].[EmployeeID],
[c].[Title],
[c].[FirstName],
[c].[MiddleName],
[c].[LastName],
[c].[Suffix],
[c].[EmailAddress]
FROM [HumanResources].[Employee] [e]
INNER JOIN [Person].[Contact] [c]
ON [c].[ContactID] = [e].[ContactID]
WHERE [LastName] = 'Ellerbrock';
```



Clustered Index Seek (Clustered)	
Scanning a particular range of rows from a clustered index.	
Physical Operation	Clustered Index Seek
Logical Operation	Clustered Index Seek
Actual Number of Rows	1
Estimated I/O Cost	0.003125
Estimated CPU Cost	0.0001581
Number of Executions	290
Estimated Number of Executions	290
Estimated Operator Cost	0.071616 (83%)
Estimated Subtree Cost	0.071616
Estimated Number of Rows	1
Estimated Row Size	200 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	True
Node ID	3

Predicate	
[AdventureWorks].[Person].[Contact].[LastName] as [c].	
[LastName]='N'Ellerbrock'	
Object	
[AdventureWorks].[Person].[Contact].	
[PK_Contact_ContactID] [c]	
Output List	
[AdventureWorks].[Person].[Contact].Title,	
[AdventureWorks].[Person].[Contact].FirstName,	
[AdventureWorks].[Person].[Contact].MiddleName,	
[AdventureWorks].[Person].[Contact].LastName,	
[AdventureWorks].[Person].[Contact].Suffix,	
[AdventureWorks].[Person].[Contact].EmailAddress	
Seek Predicates	
Seek Keys[1]: Prefix [AdventureWorks].[Person].	
[Contact].ContactID = Scalar Operator[AdventureWorks].	
[HumanResources].[Employee].[ContactID] as [e].	
[ContactID]	

Additional Predicate Diagnostics

- ActualRowsRead attribute was added in SQL Server 2012 SP3 and SQL Server 2014 SP2 (also in SQL Server 2016+) to provide additional information about predicate evaluation
 - <https://support.microsoft.com/en-us/kb/3107397>
- Compare against ActualRows, which is the number of rows output from the operator

Index Seek (NonClustered)	
Scan a particular range of rows from a nonclustered index.	
Physical Operation	Index Seek
Logical Operation	Index Seek
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Actual Number of Rows	59
Number of Rows Read	348
Actual Number of Batches	0
Estimated I/O Cost	0.0046065
Estimated Operator Cost	0.0051463 (100%)
Estimated CPU Cost	0.0005398
Estimated Subtree Cost	0.0051463
Estimated Number of Executions	1
Number of Executions	1
Estimated Number of Rows	69.6
Estimated Row Size	53 B

Demo: Analyzing plans Part I



Join Operators



Nested Loop

- Uses each row from one input to find rows from a second input that satisfy the join predicate
- Usually seen with smaller data sets and lookups, where the inner input is indexed on the join predicate
- See with Key Lookup and RID Lookup



Merge Join

- Joins two inputs which are sorted on the joining columns and returns matching rows
- Typically benefits moderate-sized data sets



Hash Join

- Joins two unsorted inputs and outputs the matching rows
- Often seen with large data sets and when grouping aggregates



Adaptive Join

- Indicates that the optimizer has the choice of a Hash Join or Nested Loop
- Plan is still cached, join type is determined at run-time
- Ideal for workloads with varied inputs/skewed data

Join Considerations

- Beware of advice telling you that specific join types (or operators, for that matter) are “good” or “bad”
- Each join type has performance characteristics of which to be aware – take note of times when the join type is not expected
- Join hints and/or forcing order = red flag
 - Generally, “edge” cases or extreme tuning scenarios warrant their use
 - Otherwise, ask questions and find out why this is happening

Demo: Analyzing plans Part II

Query Memory

- Some queries require memory to store data while sorting and joining rows, thus a memory grant is requested
 - Lifetime of the grant is equivalent to the lifetime of the query
- When available memory is insufficient, queries that require lots of memory may wait to execute (RESOURCE_SEMAPHORE wait type)
- Under-estimating memory (due to cardinality estimation issues) can cause spills to tempdb (I/O)
- Over-estimating memory can reduce concurrency!
- Memory grant information can be found in the plan

Operator Memory

- Each type of operator requires varying amounts of memory in order to perform the associated operation
- Some operators require *more* memory because they cache rows
 - More rows = more memory required
 - SQL Server performs estimates of the required memory and tries to reserve the memory grant prior to execution
 - This is where cardinality estimation is critical

Stop-and-Go Operators

- Stop-and-go operators must read ALL rows from the child operator before it can pass rows to the parent or perform specific actions
- Examples of stop-and-go operators include:
 - Hash Join
 - Outer (top) table in “blocking input”, not inner
 - Must read and process the entire build phase before the probe phase can start
 - Sort
 - Hash Aggregate
- Examples of operators that can keep streaming one row for every row that is read:
 - Nested Loop
 - Merge Join



© SQLskills, All rights reserved.
<http://www.SQLskills.com>

Sort



- As named, this operator orders rows received from an input
- Variations include:
 - Distinct Sort
 - Top N Sort
- Noteworthy: Sort tempdb spills
- Keep an eye on these for the following reasons:
 - Sort can occur in tempdb for large data sets and memory constraints (see Sort Warnings)
 - Has resource overhead (CPU / I/O / memory)
 - May not be needed if you have supporting indexes or unnecessary ORDER BY



© SQLskills, All rights reserved.
<http://www.SQLskills.com>

Batch Mode Memory Grant Feedback

- To better manage query memory grants, a feedback mechanism was introduced in SQL Server 2017 for batch mode operators
- The memory required for a query can be recalculated after query execution, and the grant numbers are updated in the cached plan
- Memory grants respect the limits set by Resource Governor or query hints
- Feedback loop can be disabled after multiple executions with variable memory requirements
 - Monitor with Extended Events

Parallelism

- Query optimization can generate either a serial or parallel plan
 - First determine cost of a serial plan, and if that cost is greater than CTP, consider parallel plans
 - The Query Optimizer ultimately caches one of them, not both
 - Parallel plans can then be used as intended, or run in serial with parallelism operators removed
- The MAXDOP server setting limits:
 - The maximum number of threads that can be used per operator
 - It does not limit the total number of threads for the plan

Identifying Parallelism in the Plan

- When parallelism occurs in query execution, you will see exchange operators in the plan
 - Distribute Streams 
 - Repartition Streams 
 - Gather Streams 
- Within the XML Showplan you'll see RelOp:
 - Parallel="true"
- You will also see the parallelism icon in the graphic for other operators that can run in parallel or serial modes
- Not all operators are parallel-aware


Clustered Index Scan (Cluste...
[FactInternetSales].[PK_Fact...
Cost: 73

NonParallelPlanReason

- Introduced in SQL Server 2012
 - Identify why a plan didn't run in parallel
 - Not all-encompassing, but a step in the right direction
 - Examples:
 - MaxDOPSetToOne (hint or max degree of parallelism)
 - EstimatedDOPIsOne (processor affinity)
 - Post by Simon Sabin that lists reasons for 2012:
 - <http://sqlblogcasts.com/blogs/simons/archive/2015/04/26/non-parallelizable-operations-in-sql-server.aspx>

Parallelism Performance Aspects

- Not inherently bad or good
 - Context matters, for example OLTP vs. DW
- Indicates higher cost query, so that should attract attention
- Some objects and operators inhibit parallelism
 - UDFs and TVFs (CLR, scalar, multi-statement), built-in functions (like OBJECT_ID), TOP
- Watch for data skew across threads
 - You can check this in XML or Properties window
 - Remember to also check for CXPACKET waits on non-zero thread IDs
- Watch for memory pressure
 - Increased memory grant requirements for parallel operations



© SQLskills. All rights reserved.
<http://www.SQLskills.com>

Demo: Analyzing plans Part III



Summary: the “Watch List”

- “Thick” lines
- High estimated cost within a query
- Scans (Index/Table)
- Lookups
- Late filtering or residual predicates
- Unexpected join selection
- Stop-and-go operators
- Sort operations (Query Optimizer added or not)
- Warnings
- Parameter sensitivity
- Hint or plan guide usage (forcing specific plan operations)



© SQLskills, All rights reserved.
<http://www.SQLskills.com>

Key Takeaways

- There are many sources of data for query performance, and numerous components worth investigating in a query plan
- Where to start when analyzing performance, and a plan, is a personal preference, and something that evolves over time...there is no “right way”
- Query tuning doesn’t just consist of reviewing a plan in isolation, take advantage of multiple sources of information to improve performance
 - Understand what behavior to “expect” for operators you see repeatedly, and take note of behaviors outside the norm
 - Delve into plan details via the Properties window, and by mining the XML for information not easily surfaced in the UI



© SQLskills, All rights reserved.
<http://www.SQLskills.com>

Additional Resources

- **Pluralsight Courses**

- SQL Server: Analyzing Query Performance for Developers
 - <https://bit.ly/2HKaoYo>
- SQL Server: Query Plan Analysis
 - <http://bit.ly/ZyExm6>
- SQL Server: Introduction to Query Store
 - <https://bit.ly/2J42aJP>



© SQLskills, All rights reserved.
<http://www.SQLskills.com>

Additional Resources

- **Whitepapers**

- *Statistics Used by the Query Optimizer in Microsoft SQL Server 2008*
 - <http://msdn.microsoft.com/en-us/library/dd535534.aspx>
- *Optimizing Your Query Plans with the SQL Server 2014 Cardinality Estimator*
 - <http://msdn.microsoft.com/en-us/library/dn673537.aspx>

- **Books**

- SQL Server Execution Plans by Grant Fritchey



© SQLskills, All rights reserved.
<http://www.SQLskills.com>

Additional Resources

- Craig Freedman's SQL Blog (not updated recently but very good!):
 - <http://blogs.msdn.com/b/craigfr/>
- Conor Cunningham Blog(s):
 - http://blogs.msdn.com/b/conor_cunningham_msft/
 - <http://www.sqlskills.com/blogs/conor/>
- Paul White's Blog
 - http://sqlblog.com/blogs/paul_white/



© SQLskills. All rights reserved.
<http://www.SQLskills.com>

Review

- Query Metrics
- Capturing and comparing query plans
- Information in a plan
- Plan analysis



© SQLskills. All rights reserved.
<http://www.SQLskills.com>

Thank you!



SQLskills Online Immersion Event

IEQuery: Immersion Event on Fixing Slow Queries,
Inefficient Code, and Caching/Statistics Problems

Part II: Removing Anti-Patterns in Transact-SQL

Jonathan Kehayias
Jonathan@SQLskills.com
@SQLPoolBoy





Jonathan Kehayias
Principal Consultant, SQLskills



Jonathan@SQLskills.com



@SQLPoolBoy



www.sqlskills.com/blogs/jonathan

Trainer/Speaker

In addition to consulting, I teach content for our IEO: Accidental DBA course, IECAG: 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

Schedule – Wednesday

- **Lecture 1: Set based concepts for developers**
 - How SQL Server processes different query patterns
 - Design considerations that affect performance
 - Data access patterns
 - Normalization and data-types
 - Eliminating row-by-row processing
 - CURSORS and WHILE Loops
 - Scalar User Defined Functions
 - Table Valued Functions
 - Using FOR XML and a table of numbers

How SQL Server Processes Queries

Row Mode

- Efficient for OLTP workloads
- Typical for row store format tables
- Execution tree operators read each required row across all columns

Batch Mode

- Efficient for data warehouse and scanning large amounts of data
- Typical for columnstore format tables
- Processes multiple rows as a batch and eliminates the need for an exchange operator during parallel processing

SELECT Statements

- Define the result set format and columns of data to return in the column list
- Define the tables that contain the source data in the FROM and JOIN clauses
- Defines how the tables logically are related to one another in the ON clause following a JOIN or the WHERE clause
- Defines the filtering conditions for the output rows in the WHERE or HAVING clause of a GROUP BY
- Does not define how SQL Server will execute the request unless specific hints are used (non-procedural)

Query Execution Plans

- **Parsing**
 - Syntax is checked to determine if the query is written properly
 - Produces a parse tree that is handed off
- **Binding**
 - Algebrizer analyzes the parse tree for name resolution
 - Do the tables exist, are columns in the where clause in the tables, permissions
 - Produces a query processor tree that is then handed to the optimizer
- **Optimization**
 - The optimizer attempts to determine the most efficient method of executing the request based on costs
 - Produces an execution plan that is used by the execution engine to process the request

Operator Precedence

Level	Operators
1	~ (Bitwise NOT)
2	* (Multiplication), / (Division), % (Modulus)
3	+ (Positive), - (Negative), + (Addition), + (Concatenation), - (Subtraction), & (Bitwise AND), ^ (Bitwise Exclusive OR), (Bitwise OR)
4	=, >, <, >=, <=, <>, !=, !>, !< (Comparison operators)
5	NOT
6	AND
7	ALL, ANY, BETWEEN, IN, LIKE, OR, SOME
8	= (Assignment)

Normalization

- Process of organizing data in a database into tables with relationships to other tables
 - Better flexibility – design changes
 - Reduce redundancy – wasted disk/memory space
 - Enforce data integrity – reduce where data changes must occur
 - Remove inconsistent dependencies – group data into related subjects
- Rules define a standard of database normalization
 - First Normal Form
 - Second Normal Form
 - Third Normal Form – typical for OLTP



© SQLskills, All rights reserved.
<http://www.SQLskills.com>

First Normal Form (1NF)

- Eliminate repeating groups in individual tables
 - Does not store multiple values within a single column
 - Defines each value as atomic – cannot be broken to smaller pieces
- Create a separate table for each set of related data
 - Does not use multiple columns in a table to store similar data
 - E.g. Phone Numbers, vendor codes
- Identify each set of related data with a primary key
 - 1NF requires a unique constraint on the table - no exact duplicate rows exist



© SQLskills, All rights reserved.
<http://www.SQLskills.com>

First Normal Form (1NF)

Name	Address	Phone1	Phone2	PostalCode	Country
Jonathan Kehayias	123 Disney Hwy, Orlando, FL 34582	407-528-1124	727-485-3325	34582	USA
Paul Randal	36 BAOSHAN JIUCUN, BAOSHAN DISTRICT	251-112-2425	927-555-1212, 855-243-8514	201900	CHINA

ID	Name	Address	Phone
1	Jonathan Kehayias	123 Disney Hwy	407-528-1124
2	Jonathan Kehayias	123 Disney Hwy	727-485-3325
3	Paul Randal	4432 Nowhere Lane	251-112-2425
4	Paul Randal	4432 Nowhere Lane	927-555-1212
5	Paul Randal	4432 Nowhere Lane	855-243-8514

Second Normal Form (2NF)

- Must first meet the requirements of 1NF
- Create separate tables for sets of values that apply to multiple records
 - Records should only depend on another table's primary key
 - E.g. Addresses stored in an address table and referenced by AddressKey in Customer, Orders, Shipping, etc tables
- Define relationships between tables with a foreign key
 - Enforce data integrity and prevent orphaned data

Second Normal Form (2NF)

PersonID	Name	Address	PostalCode	Country
1	Jonathan Kehayias	123 Disney Hwy	34582	USA
2	Paul Randal	36 BAOSHAN JIUCUN, BAOSHAN DISTRICT	201900	China

PhoneID	PersonID	PhoneNumber
1	1	407-528-1124
2	1	727-485-3325
3	2	251-112-2425
4	2	927-555-1212
5	2	855-243-8514

Third Normal Form (3NF)

- Ideal for OLTP applications
 - Reduces duplication of data
 - Typically free of INSERT/UPDATE/DELETE anomalies
- Must first meet the requirements of 2NF
- Eliminates columns that do not depend on the key
 - E.g. Cities, states, zip codes
 - May not always be followed for every table (see above examples for addresses)
 - Should be applied for frequently changing data that affects multiple relationships or records – only update parent table field

Third Normal Form (3NF)

PersonID	Name	Address	PostalCodeID
1	Jonathan Kehayias	123 Disney Hwy	1
2	Paul Randal	36 BAOSHAN JIUCUN	2

PhoneID	PersonID	PhoneNumber	PostalCodeID	PostalCode	Country	City	Provence
1	1	407-528-1124	1	34582	USA	Orlando	FL
2	1	727-485-3325	2	201900	China	BAOSHAN DISTRICT	Shanghai
3	2	251-112-2425					
4	2	927-555-1212					
5	2	855-243-8514					

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

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 (changes in SQL Server 2019)
 - 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



© SQLskills, All rights reserved.
<http://www.SQLskills.com>

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



© SQLskills, All rights reserved.
<http://www.SQLskills.com>

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

Schedule – Wednesday

- **Lecture 2: Scalability and Testing**
 - Making row based processing scale when required
 - Understanding Sargability and Index Scans
 - Implicit Conversions
 - IN vs UNION ALL for performance
 - Profiling during development and testing
 - Fatal flaws to avoid

Scaling Row Based Processing

- Application design patterns
 - Multi-threaded apartment for asynchronous execution
 - Retrieve initial data set from SQL Server and execute row based processes on background or "worker" class threads in the application
 - Best for situations where initial data set is not changing
 - Watch for blocking conditions for long open result sets – fire hose cursor consumption of data
- SQL Server design pattern
 - Service Broker queue activation allows parallel processing of messages natively within SQL Server
 - Transactional based and ensures restart of processing after failure
 - Implemented entirely with TSQL as a part of the database

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 eliminate Scan



© SQLskills, All rights reserved.
<http://www.SQLskills.com>

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



© SQLskills, All rights reserved.
<http://www.SQLskills.com>

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

Replacing IN with UNION

- **Characteristics**
 - WHERE clause similar to (Column 1 IN (12, 16))
 - WHERE clause similar to (Column1 = 12 OR Column1 = 16)
- **Problems**
 - May causes Table/Index Scan over Seek
 - May cause a range seek
- **Replace with**
 - Separate SELECT statements with WHERE clause for each criteria using UNION ALL to concatenate results to final output
 - Dynamic SQL to build the UNION ALL query string

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



© SQLskills, All rights reserved.
<http://www.SQLskills.com>

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



© SQLskills, All rights reserved.
<http://www.SQLskills.com>

Review

- Set based concepts for developers
 - How SQL Server processes different query patterns
 - Design considerations that affect performance
 - Eliminating row-by-row processing
- Scalability and Testing
 - Making row based processing scale when required
 - Understanding Sargability and Index Scans
 - Profiling during development and testing



© SQLskills, All rights reserved.
<http://www.SQLskills.com>

Additional Resources

- SQLPerformance.com Articles
 - <https://www.sqlperformance.com/>
- Paul White's Blog
 - http://sqlblog.com/blogs/paul_white/
- Jonathan's Posts
 - Parallel Maintenance
 - <https://www.sqlskills.com/blogs/jonathan/parallel-maintenance/>
 - Implicit Conversions
 - <https://www.sqlskills.com/blogs/jonathan/finding-implicit-column-conversions-in-the-plan-cache/>
- Paul's Posts
 - DBCC CHECKDB performance and computed-column indexes
 - <https://www.sqlskills.com/blogs/paul/dbcc-checkdb-performance-and-computed-column-indexes/>



© SQLskills, All rights reserved.
<http://www.SQLskills.com>

Thank you!



SQLskills Online Immersion Event

IEQuery: Immersion Event on Fixing Slow Queries,
Inefficient Code, and Caching/Statistics Problems

**Part III: How to Differentiate Caching / Statistics
problems and SOLVE THEM!**

Kimberly L. Tripp
Kimberly@SQLskills.com
@KimberlyLTripp





Kimberly L. Tripp
Founder / President, SQLskills



Kimberly@SQLskills.com



@KimberlyLTripp



www.sqlskills.com/blogs/Kimberly



Consultant / Trainer / Speaker / Writer

- Author / instructor for SQL Server Immersion Events: IEPTO1, IEPTO2, and IEHADR
- Author / presenter for Pluralsight
- Instructor and exam author for SQL Server and Sharepoint MCMs
- Author / manager of SQL Server 2005 and 2008 Launch Content
- Author / speaker at Microsoft TechEd, SQLPASS, ITForum, TechDays, and SQLintersection
- Author of SQL Server Whitepapers on MSDN/TechNet: Partitioning, Snapshot Isolation, Manageability, SQLCLR for DBAs
- Author / presenter for MSDN and TechNet (25+)
- Instructor / SME for Microsoft University

Data Platform MVP

Recognized as an MVP by Microsoft since 2002

When I'm not working with SQL

- Traveler/adventurer and avid diver/photographer: www.BlueWaterImages.com
- @BlueWaterImages



You Know The Drill

- **Lecture 1: 90 minutes from 10:00am until 11:30am PT**
 - Please use "CHAT" for questions *during* the lecture
- **Open Q&A 1: 30 minutes from 11:30am until 12:00 noon PT**
 - Will address unanswered questions from chat
 - May open up for "open mic" questions
- **Mandatory break: 30 minutes from 12:00 noon until 12:30pm PT**
 - Everyone needs a bit of a break!
- **Lecture 2: 90 minutes from 12:30pm until 2:00pm PT**
 - Please use "CHAT" for questions *during* the lecture
- **Open Q&A 1: up to 60 minutes from 2:00pm until 3:00pm PT**
 - Will address unanswered questions from chat
 - Will open up for "open mic" questions

Time Conversions

10:00am PT

= 1:00pm ET

= 6:00pm UTC



Slide 108

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Course Resources & Pluralsight Access

- Paul (paul@sqlskills.com) will send you an email for a FREE 30-day access to ALL SQLskills online training classes on Pluralsight
 - No strings attached, no credit-card required
 - If you don't receive it, send mail to Paul
- **Course resources will be sent upon completion (Friday+)**
 - I may want to update a slide and/or add some notes/resources based on questions and discussions
 - The resource zip will be password protected. Please do not distribute; this is for course attendees only. The password will be sent in email.
- **Email Paul (Paul@SQLskills.com) if you have problems finding it or opening it from Monday (Oct 29) forward; I plan to finalize the resources (complete the whiteboard, etc.) on Friday!**



PLURALSIGHT



Slide 109

<http://www.SQLskills.com>

© SQLskills Online Immersion Event. All rights reserved.

Course Resources

- **Course content (all presenter decks for all three days)**
- **Whiteboard(s)**
- **Demos**
 - All instructor-led demo scripts
 - All reference scripts
- **Reference links (in the "resources zip")**
- **Online recordings for these sessions**

Please do not redistribute: These resources are for IEQuery attendees only.
Please respect our IP and time in creating these resources and do not share/forward.



Slide 110

<http://www.SQLskills.com>

© SQLskills Online Immersion Event. All rights reserved.

Schedule – Thursday

▪ Lecture 1: Troubleshooting Statement Execution and Caching

- ❑ Different ways to execute statements
- ❑ Statements caching for reuse
- ❑ Statement auto-parameterization
- ❑ Dynamic string execution
- ❑ sp_executesql
- ❑ Stored procedures
- ❑ Literals, variables, and parameters
- ❑ The life of a plan in cache
- ❑ Plan cache limits
- ❑ Bringing it all together



Slide 111

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Schedule – Thursday

▪ Lecture 2: Troubleshooting Plan Problems Related to Statistics (not Caching)

- ❑ Statement selectivity
- ❑ What kinds of statistics exist
- ❑ How does SQL Server use statistics
- ❑ Creating additional statistics
- ❑ Updating statistics



Slide 112

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Overview: Root-cause Analysis

Where is the ACTUAL problem?

- Cache?
- Statistics?



Performance Problems

- **Query performance inconsistencies**
 - Execution time varies
 - Statement's execution plan varies
 - IO / CPU metrics vary
- **Variations due to:**
 - Statement execution method
 - Parameters
 - Time (fragmentation, statistics not current, plan not valid)
- **Sledgehammer approaches**
 - Updating statistics
 - Rebuilding indexes
 - Clearing cache



Slide 114

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

BETTER: Troubleshooting Methodologies (1 of 2)

- **Is this a CACHED plan method?**
 - Stored procedure
 - sp_executesql
- **Test to see if the optimal plan varies from the cached plan?**
 - EXECUTE <procedure> <params> or sp_executesql (same way)
 - VS. recompiling: EXECUTE <procedure> <params> WITH RECOMPILE
 - NOTE: execute with recompile does NOT allow the parameterization embedding optimization [doesn't optimize as effectively as OPTION (RECOMPILE) so there are features that might not be leveraged EXCEPT when using OPTION (RECOMPILE) at the statement level but that's not a problem here...]
- **Do you get a different plan for the second?**
 - Parameter sensitivity ("sniffing") problem
 - Long term solution is to changing the code



Slide 115

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

BETTER: Troubleshooting Methodologies (2 of 2)

- **Is this a poorly performing but NEW (non-cached) plan?**
 - AdHoc statement
 - First execution
 - Plan doesn't change when using WITH RECOMPILE
- **Use ACTUAL EXECUTION**
 - Estimated plan vs. actual plan
 - Be sure to check executions * rows (not just estimated rows vs. actual rows)
 - Problems / solutions that can be exposed by incorrect row estimations
 - Statistics out-of-date -> updating scenarios
 - Estimate is incorrect because of skewed data -> filtered statistics scenarios
 - Estimate is incorrect because of estimation algorithm -> cardinality estimation scenarios



Slide 116

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Lecture 1: Troubleshooting Statement Execution and Caching



Overview: Statement Execution and Caching

- Different ways to execute statements
- Statements caching for reuse
- Statement auto-parameterization
- Dynamic string execution
- `sp_executesql`
- Stored procedures
- Literals, variables, and parameters
- The life of a plan in cache
- Plan cache limits
- Bringing it all together



Slide 118

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Different Ways to Execute SQL Statements

- **Ad hoc statements**
 - Possibly, as auto-parameterized statements
- **Dynamic string execution (DSE)**
 - EXECUTE (@string)
- **sp_executesql (forced statement caching)**
- **Prepared queries (forced statement caching through “parameter markers”)**
 - Client-side caching from ODBC and OLEDB (parameter via question mark)
 - Exposed via SQLPrepare / SQLExecute and ICommandPrepare
- **Stored Procedures**
 - Including statements with literals, parameters, and/or variables
 - Including dynamic strings
 - Including sp_executesql

*These two behave
EXACTLY the same way!*

*In this section, these behave the same way but some exceptions
exist with certain statement types inside stored procedures
(a bit more on this is coming up)*



Slide 119

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Some Statements can be Cached for Reuse (1 of 2)

- **Ad hoc statements and dynamic strings are evaluated at runtime**
 - If a statement is very simple (“safe”), then it can be parameterized and cached
 - It’s generally a good thing that the plans are cached
 - Saves CPU/time
 - Reduced footprint in the cache
 - This can lead to a small amount of prepared plan cache bloat when the parameters are typed per execution:

```
SELECT ... WHERE member_no = 12
↳ (@1 tinyint)SELECT ... WHERE [member_no]=@1
```

```
SELECT ... WHERE member_no = 278
↳ (@1 smallint)SELECT ... WHERE [member_no]=@1
```

```
SELECT ... WHERE member_no = 62578
↳ (@1 int)SELECT ... WHERE [member_no]=@1
```



Slide 120

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Why is this Such an Issue?

- EVIL ORMS!
- ORM = Object Relational Mapper
- Pro – Rapid Application Development
- Con – Fast to production, SLOW for the life of the application
- How **NOT** to structure your database-backed web applications: a study of performance bugs in the wild
- <https://bit.ly/2Ixpkl4>



Caitie McCaffrey @caitie 54m
Great read on some performance pit falls in ORMs

Adrian Colyer @adriancolyer
ORM could also stand for "Opaque Relational Mapper". Yang et al., investigate common ORM-related performance issues and a build a tool to help you find them in your own (Rails) applications. blog.acolyer.org/2018/06/28/how... Just a few lines of code changed can make a huge difference!

Caitie McCaffrey @caitie 31m
Also this paper doesn't discuss consistency issues. I think this is because the studied ORMs are all on top of relational databases.

Caitie McCaffrey @caitie 30m
But in my experience ORMs on top of Eventually Consistent or NoSql data stores also have a ton of consistency challenges because you need to understand the implementation to get the consistency correct.

Caitie McCaffrey @caitie 27m
Basically ORMs are very leaky storage abstractions, and I would strongly caution against using them.

I totally agree!

Slide 121

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Some Statements can be Cached for Reuse (2 of 2)

- Ad hoc statements and dynamic strings are evaluated at runtime
 - And, unfortunately, most statements won't be safe:
 - Many query limitations:
 - FROM clause cannot have more than one table
 - WHERE clause cannot have expressions joined by OR
 - WHERE clause cannot have an IN clause
 - Statement cannot contain a sub-query
 - VERY restrictive (see Appendix A in whitepaper for complete list)
 - Parameters do not change plan choice
 - Even when a statement's NOT safe, the un-parameterized statement (and the specific literal values) will be placed in the ad hoc plan cache
 - Used for later "exact textual matching" cases
 - Eats up the cache quickly because:
 - Most statements aren't safe
 - Lots of statements are executing



Slide 122

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Statement Auto-Parameterization: Keep It Simple!

- **How does parameterization work by default: SIMPLE**
 - Most statements are probably NOT deemed safe
- **Database option: parameterization FORCED**
 - Generally, not recommended
 - Many more statements are forced to be cached
 - PRO: if you have stable plans from a lot of adhoc clients then this might help to significantly reduce CPU
 - CON: If you have some statements that really aren't safe, you could end up executing bad plans... better to do this right from the start and control it yourself with stored procedures (much more difficult!)
 - Recommendation: can investigate plan stability **BEFORE changing to forced** by using query_hash and query_plan_hash (check out the Pluralsight course on [SQL Server: Optimizing Ad Hoc Statement Performance](#), Section: Plan Cache Pollution)
- **If statement is parameterized and safe (or forced), SQL Server places statement in cache for subsequent executions**
- **If statement is unsafe, SQL Server places statement in cache for subsequent executions through TEXTUAL MATCHING ONLY**



Slide 123

This Photo by Unknown Author
is licensed under [CC BY-ND](#)

<http://www.SQLskills.com>

© SQLskills Online Immersion Event. All rights reserved.

Ad Hoc Statement / Dynamic String Execution (DSE)

- **Statement / string is NOT evaluated until runtime (execution)**
- **Parameters allow virtually any statement to be built “dynamically”**
- **This statement is then treated as an ad hoc statement**
 - If it's safe – it will be parameterized, saved in cache, and reused
 - If it's unsafe – it will be recompiled for each/every execution
 - Just to be clear – DSE does not automatically mean it's compiled for every execution
 - Textual matching can occur (with statements in the ad hoc plan cache)
 - Parameterization can occur (again, only if it's safe)
- **String can be up to 2GB in size**
 - 2005+: Can declare variable of type (n)varchar(max)
 - sp_executesql only allows parameters where a typical SQL statement would allow them; however, these two can be combined!
- **Can be complex to write, read, perform**
- **And there's a whole discussion about security...**



Slide 124

<http://www.SQLskills.com>

© SQLskills Online Immersion Event. All rights reserved.

Understanding sp_executesql

- Usually used to help build statements from applications
- Parameters are typed explicitly
- Forces a plan in cache for the parameterized string (when one doesn't already exist) – subsequent executions will use this plan
 - Can be EXCELLENT if the statement's plan is stable even with different parameters
 - Can be horrible if the statement's most optimal plan varies from execution to execution (because of the parameters)
- Almost like dynamic string execution, but it's not!
 - Often compared to DSE [where you EXEC (@ExecStr)] but they're not the same
 - sp_executesql is a parameterized statement that works JUST like a stored procedure
 - DSE is just a way of building an ad hoc statement that's not evaluated until runtime
 - If it's safe – it's parameterized and reused
 - If it's not safe – then it's not (meaning, it will be in the ad hoc plan cache but not the compiled plan cache AND it will need to be compiled for each execution)



Slide 125

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Stored Procedure Caching Just Like sp_executesql

- Places a plan in cache for the stored procedure (when one doesn't already exist) – subsequent executions will use this plan
- Reusing plans can be good
 - When different parameters don't change the optimal plan, then caching / saving and reusing is excellent!
 - SQL Server saves CPU and time in compilation
- Reusing plans can be VERY bad
 - When different parameters are used and the optimal plans vary by parameter set, then reusing the plan can be horribly bad



Slide 126

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Literals, Variables, and Parameters (1 of 2)

▪ Literals

```
SELECT ... WHERE member_no = 12
```

- member_no is being compared to a specific, defined value
- Literals are “known” values at the time of optimization (compilation)
- They provide the best information for optimization

▪ Variables

```
DECLARE @mno int = 12
```

```
SELECT ... WHERE member_no = @mno
```

- @mno is being defined and set in the declaration line
- Variables are “unknown” values at the time of optimization (compilation) as the assignment does not occur until runtime
- They provide NO useful information for optimization
- The optimizer has to use “averages” instead of real values
 - This can be both good and bad...



hidden slide
w/extra details
Check out the Pluralsight
courses for in-depth information
about these options!



Slide 127

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Literals, Variables, and Parameters (2 of 2)

▪ Parameters look like variables but they're not...

▪ Variables CAN be defined within a stored procedure

```
CREATE PROCEDURE procname  
    ( @parameter1 int )
```

```
AS
```

```
DECLARE @mno int = 12
```

```
SELECT ... WHERE member_no = @mno
```

```
SELECT ... WHERE member_no = @parameter1
```

- @mno is being defined and set inside the stored procedure; this is a variable
- @parameter1 is being defined as a parameter and the value will be KNOWN at execution
- If this procedure does not yet have a plan defined in cache, then SQL Server will evaluate PARAMETER values during optimization (not VARIABLES)
- This leads to the phrase: parameters can be “sniffed,” variables are unknown.



hidden slide
w/extra details
Check out the Pluralsight
courses for in-depth information
about these options!



Slide 128

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

The Life of a Plan in Cache

- **A plan is generated when no plan already exists in cache**
- **Plans are never saved on disk and may not persist within the cache**
 - Some operations completely remove / evict the plan from the cache
 - Server-level: Server restart | DBCC FREEPROCCACHE | some RECONFIGURE changes
 - See Pluralsight recordings starting with Side Effect: Plan Cache Flush for version-specific details and demo (Optimizing Procedural Code course)
 - Database-level: DBCC FLUSHPROCINDB (undocumented) | sp_dbcmtlevel
 - Procedure-level: sp_recompile or they're aged out through non-use
 - Some operations cause the plan to be invalidated (but it still remains an object in the cache); these can be tracked
 - Schema of base object changes (ALTER TABLE / ALTER <object>)
 - Including when indexes are added to the base object
 - Statistics of base objects change
 - See recordings starting with Plan Invalidation for version-specific details and demos
- **When a plan exists in cache, all subsequent executions use that plan**



Slide 129

<http://www.SQLskills.com>

© SQLskills Online Immersion Event. All rights reserved.

Plan Invalidation

- **Versions prior to 2012**
 - If database option: auto_update_stats is ON
 - Updating statistics causes plan invalidation
 - If database option: auto_update_stats is OFF
 - Updating statistics does NOT cause plan invalidation
 - If you manually update statistics and have set auto_update_statistics to OFF, add sp_recompile @tname to your stats scripts (remembering SCH_M problems)
 - For more info: Erin Stellato's links about auto update stats/plan invalidation:
 - Statistics and Recompilations: <http://erinstellato.com/2012/01/statistics-recompilations/>
 - Statistics and Recompilations, Part II: <http://erinstellato.com/2012/02/statistics-recompilations-part-ii/>
- **SQL Server 2012+**
 - Plan invalidation is NOT affected by the setting of auto update stats
 - Plan invalidation does NOT occur if data has not changed
 - Only for UPDATE STATISTICS
 - An index rebuild will update statistics as well as cause plan invalidation, even if no data has changed thus increasing the importance for "only when data has changed" logic in maintenance routines.



Check out the Pluralsight
courses for in-depth information
about these options!



Slide 130

<http://www.SQLskills.com>

© SQLskills Online Immersion Event. All rights reserved.

Plan Cache Usage/Limits

- The “plan cache” (a.k.a. “procedure cache”)
- Uses “stolen” pages from the buffer pool (data pages)
- View cached plans: `sys.dm_exec_cached_plans`
- Plan cache memory limits:
 - SQL Server 2005 SP2 and higher
 - 75% of visible target memory from 0-4GB
 - + 10% of visible target memory from 4Gb-64GB
 - + 5% of visible target memory > 64GB
 - SQL Server 2005 RTM and SQL Server 2005 SP1
 - 75% of visible target memory from 0-8GB
 - + 50% of visible target memory from 8Gb-64GB
 - + 25% of visible target memory > 64GB
 - SQL Server 2000
 - SQL Server 2000 4GB upper cap on the plan cache
 - 32-bit? AWE memory is not “visible” to the plan cache (plan cache has to live in “real memory”)

2005 SP2 +	
Memory	Plan Cache
4GB	3.0GB
8GB	3.5GB
16GB	4.2GB
32GB	5.8GB
64GB	9.0GB
128GB	12.2GB
256GB	18.6GB
512GB	31.4GB

Consolidation Note
If you've consolidated / virtualized multiple servers then the real question is – how much memory have you given to each SQL Server?



Slide 131

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Reducing Plan Cache Bloat & Optimizing Statement Execution

- Analyze your plan cache to see if you have ad hoc “bloat”
 - See plan cache blog post slide for query/details to analyze your cache
- Turn on the server-wide configuration “optimize for ad hoc workloads”
- Setup a job to regularly monitor for plan cache bloat and clear the SQL Plans portion of the cache using `DBCC FREESYSTEMCACHE`
- Work to change the way that ad hoc requests are made
 - Use ad Hoc for statements with unstable plans
 - Use `sp_executesql` for statements with stable plans
 - If you want centralized logic, code reuse, and compiled / cached plans (when they're stable) and lots of other options (for when the plans are not stable), use stored procedures
 - Written by database developers that should
 - Know the data / workload / requirements
 - Know how SQL Server works
 - Can be written to balance CPU / caching for optimal performance!



Slide 132

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Review: Statement Execution and Caching

- Different ways to execute statements
- Statements caching for reuse
- Statement auto-parameterization
- Dynamic string execution
- `sp_executesql`
- Stored procedures
- Literals, variables, and parameters
- The life of a plan in cache
- Plan cache limits
- Bringing it all together



Slide 133

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Questions!



Lecture 2: Troubleshooting Plan Problems Related to Statistics



Overview: Plan Problems Related to Statistics

- Statement selectivity
- What kinds of statistics exist
- How does SQL Server use statistics
- Creating additional statistics
- Updating statistics



Slide 136

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Statement Execution Simplified

Optimization = Compilation



Optimization: this is really where you can have the greatest impact and where the most interesting events can occur...

- 1) Parse
- 2) Standardization / normalization / algebrization ⇒ query tree
- 3) Cost-based optimization
(statistics are used to come up with optimization...)
- 4) Compilation
- 5) Execution



Slide 137

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Cost-Based Optimization

- Find a reasonable subset of possible algorithms to access data based on:
 - The query: sometimes a rewrite helps (join rewritten as sub-query or vice versa)
 - Any joins: sometimes a derived table helps (sub-query in the FROM clause)
 - Any SARGs: sometimes rewriting SARGs helps (well-defined predicates)
 - Data selectivity
 - Join density
- The more information the optimizer has the better...
- How do you provide the BEST information?
- One of the best ways to “influence” your query plans is through effective statistics (and better indexes)
- Understanding optimization / estimation is important for troubleshooting a wide variety of solvable query problems!



Slide 138

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Selectivity

- Not just based on the number of rows returned
- Always relative to the number of rows in the table (usually expressed as a percentage)
- Low number of rows = high selectivity
 - Any index is useful if even ONE condition is highly selective!
- High number of rows = low selectivity
 - What is considered low selectivity? 5%, 10%, 15%???
 - Remember there's always a "tipping point" (where one option is better than another – based on the amount of data that's going to be processed)
 - The "tipping point" between using a non-clustered index that does NOT cover a query vs. performing a table scan happens at a very low percentage (usually 1-2%)
 - Non-clustered indexes that "support" a WHERE clause might not be *selective enough* to be used
 - This can cause plan changes / instabilities
 - NOTE: To learn more about the tipping point, check out my Indexing course on Pluralsight
 - Must consider some form of covering



Slide 139

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Understanding Selectivity How Would YOU Find This Data?

- Imagine a table of employee data, for a Chicago company
- The table is clustered by EmployeeID
- Imagine executing this query?

```
SELECT e.*  
FROM dbo.EmployeesPersonalAddresses AS e  
WHERE e.city = 'Chicago' not selective enough  
WHERE e.city = 'Glenview' not an easy answer  
WHERE e.city = 'Peoria' selective enough
```

- When is an index on "City" useful?
 - When the data is selective ENOUGH...

**More importantly, how does
SQL Server know?**



Slide 140

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

What Kinds of Statistics Exist?

- **Statistics on indexes**
- **Auto-created statistics**
 - Named `_WA_SYS_`
 - Paul's blog: How are auto-created column statistics names generated? (<http://bit.ly/1gjY28K>)
 - Created automatically when a missing statistics event is encountered
 - Permanent objects in the database, will get auto-updated if database option is set
- **User-created statistics**
 - Created by you...
 - Could have been recommended by DTA and named `_dta_stat_`
- **Hypothetical indexes**
 - Created during DTA's analysis phase
 - Dropped by letting DTA complete successfully
 - Can be created manually for "what if analysis using auto pilot"
 - *See hidden slide, link, and demo script for help on understanding this VERY cool feature!*



Slide 141

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

DBCC AUTOPILOT

- **Check out the Simple Talk article**
Hypothetical Indexes on SQL Server (<http://bit.ly/1fi472d>)



Check out my script samples /
examples for using AUTOPILOT!

```
CREATE INDEX <name> ON <tablename> (<columns>)
WITH STATISTICS_ONLY = -1
GO
DBCC AUTOPILOT(0, <dbid>, <objectid>, <indexid>)
GO
SET AUTOPILOT ON
GO
RUN QUERY TO TEST INDEX USAGE?
(output is showplan xml unless graphical plan on)
GO
SET AUTOPILOT OFF
```



Slide 142

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

What Do They Look Like?

1		2		Statistics Header			
Name	Updated	Rows	Rows Sampled	Steps	Density	Average key length	String Index
MemberName	Oct 10 2008 1:02AM	10000	10000	26	0	21.5526	YES

Density Vector

All density	Average Length	Columns
0.03846154	5.6154	Lastname
0.0001	16.5526	Lastname, Firstname
0.0001	17.5526	Lastname, Firstname, MiddleInitial
0.0001	21.5526	Lastname, Firstname, MiddleInitial, member_no

Histogram

RANGE_HI_KEY	RANGE_ROWS	EQ_ROWS	DISTINCT_RANGE_ROWS	AVG_RANGE_ROWS
ANDERSON	0	385	0	1
BARR	0	385	0	1
CHEN	0	385	0	1
...	...	...	...	...
ZUCKER	0	384	0	1

Overall,
this is
weird
data?!



Slide 143

<http://www.SQLskills.com>

© SQLskills Online Immersion Event. All rights reserved.

How Do You See Them? (1)

- **DBCC SHOW_STATISTICS (tname, statname)**
 - Gives you ALL the statistical details
 - Number of rows and number of rows on which the statistics were based
 - Densities for all LEFT based subsets of column, including the CL key (last – if not already somewhere in the index)
 - Histogram for high order element
- **sp_autostats tname**
 (NOTE: I almost never use this; I use queries against sys.stats but I included here so you'd know it exists)

Index Name	AUTOSTATS	Last Updated
[member_ident]	ON	2008-08-26 17:18:12.5
[member_corporation_link]	ON	2008-08-26 17:18:12.6
[member_region_link]	ON	2008-08-26 17:18:12.7
[MemberName]	ON	2008-10-29 11:13:29.2
[_WA_Sys_00000003_OCBAE8]	ON	2008-10-29 11:28:32.3



Slide 144

<http://www.SQLskills.com>

© SQLskills Online Immersion Event. All rights reserved.

Density Vector

- **Based on a left-based subset of key columns, details the distribution of data**
- **Rows * All density = average number of rows given that column or combination of columns**
 - Index LastName, FirstName will have average for last names alone as well as the combination of last names and first names
 - Index FirstName, LastName will have average for first names alone as well as the combination of first names and last names
 - The density vector value for the combination will be the same
- **Density information**
 - Density for LastName = 0.03846154
[10,000 Rows * 0.03846154 = 384.6154 rows returned on average]
 - Density for LastName, FirstName combo – what does that tell you?



Slide 145

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Histogram

- **Up to 201 total steps with each step's density information**
 - Up to 200 distinct, actual values from the table
 - 1 row for NULLs if the column allows NULL values
- **Histogram has the most detail about the first column (sometimes referred to as the high-order element [LastName])**
 - Anderson 385 rows
 - Barr 385 rows
- **Cannot modify characteristics of statistic structures**
 - Steps chosen / total number of steps
 - How it's built (step compression)
 - Updating can only be done when the entire set is evaluated
 - Update statistics
 - Index rebuild (not reorg)



Slide 146

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.



hidden slide
for the demo

Each of these next few slides
corresponds to a portion of the demo

Statistics Usage

- **Estimations for known values**
 - Histogram step value: EQ_ROWS
 - Histogram value in step range: AVG_RANGE_ROWS
 - Distinct value estimation (TUPLE_CARDINALITY)
 - 1 / All density
- **Unknown values**
 - Density vector average
 - Density * Rows
- **Selectivity for a set (indirect index usage)**
 - Index statistics are often used from the index that the query uses
 - Column-level or index-level statistics can be analyzed independently to determine other possible algorithms (even when not covered)



Slide 147

<http://www.SQLskills.com>

© SQLskills Online Immersion Event. All rights reserved.



hidden slide
for the demo

Each of these next few slides
corresponds to a portion of the demo

Histogram Step Value

```
SELECT [m].*
FROM [dbo].[Member] AS [m]
WHERE [m].[LastName] = 'Chen'
```

```
DBCC SHOW_STATISTICS
('Member', 'MemberName')
WITH HISTOGRAM;
```

	RANGE_HI_KEY	RANGE_ROWS	EQ_ROWS	DISTINCT_RANGE_ROWS	AVG_RANGE_ROWS
1	ANDERSON	0	385	0	1
2	BARR	0	385	0	1
3	CHEN	0	385	0	1
4	CHEN	0	385	0	1

Clustered Index Scan (Clustered)	
Scanning a clustered index, entirely or only a range.	
Physical Operation	Clustered Index Scan
Logical Operation	Clustered Index Scan
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Actual Number of Rows	385
Actual Number of Batches	0
Estimated I/O Cost	0.107569
Estimated Operator Cost	0.118726 (100%)
Estimated Subtree Cost	0.118726
Estimated CPU Cost	0.011157
Estimated Number of Executions	1
Number of Executions	1
Estimated Number of Rows	385
Estimated Row Size	189 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	False
Node ID	0
Predicate	[(Credit].[dbo].[member].[lastname] as [m].[lastname] like 'Chen']
Object	[(Credit].[dbo].[member].[member_idnt].[m]]



Slide 148

<http://www.SQLskills.com>

© SQLskills Online Immersion Event. All rights reserved.



Each of these next few slides corresponds to a portion of the demo

Histogram Value In Step Range

```
SELECT [m].*
FROM [dbo].[member] AS [m]
WHERE [m].[corp_no] = 404;
```

```
DBCC SHOW_STATISTICS
('Member', 'member_corporation_link')
WITH HISTOGRAM;
```

	RANGE_HI_KEY	RANGE_ROWS	EQ_ROWS	DISTINCT_RANGE_ROWS	AVG_RANGE_ROWS
137	402	0	8	0	1
138	403	0	4	0	1
139	407	14	5	2	7
140	408	0	7	0	1

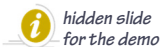
4 rows for value 403 | 5 rows for value 407 | 14 rows between 403 and 407 but not including 403 or 407
14 rows between 403 and 407 over 2 distinct values = 7 rows "on average"
404 / 405 / 406 will estimate 7... is that correct?

Index Seek (NonClustered)	
Scan a particular range of rows from a nonclustered index.	
Physical Operation	Index Seek
Logical Operation	Index Seek
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Actual Number of Rows	0
Actual Number of Batches	0
Estimated Operator Cost	0.0032897 (14%)
Estimated I/O Cost	0.003123
Estimated CPU Cost	0.000164
Estimated Subtree Cost	0.0032897
Number of Executions	1
Estimated Number of Executions	1
Estimated Number of Rows	158



Slide 149

© SQLskills Online Immersion Event. All rights reserved.



Each of these next few slides corresponds to a portion of the demo

Distinct Value Estimation

```
SELECT DISTINCT [m].[corp_no]
FROM [dbo].[member] AS [m];
```

```
DBCC SHOW_STATISTICS
('Member', 'member_corporation_link')
WITH DENSITY_VECTOR;
```

	All density	Average Length	Columns
1	0.0025	0.6008	corp_no
2	0.0001	4.6008	corp_no, member_no

```
All density * rows = average
1 / All density =
tuple_cardinality
1 / 0.0025 = 400
```

Physical Operation	Stream Aggregate
Logical Operation	Aggregate
Actual Execution Mode	Row
Estimated Execution Mode	Row
Actual Number of Rows	400
Actual Number of Batches	0
Estimated I/O Cost	0
Estimated Operator Cost	0.0052 (16%)
Estimated CPU Cost	0.0052
Estimated Subtree Cost	0.0320746
Estimated Number of Executions	1
Number of Executions	1
Estimated Number of Rows	400
Estimated Row Size	11 B
Actual Rebinds	0
Actual Rewinds	0
Node ID	0
Output List	
[Credit].[dbo].[member].corp_no	
Group By	
[Credit].[dbo].[member].corp_no	



[QLskills.com](http://www.QLskills.com)
rights reserved.



Each of these next few slides

corresponds to a portion of the demo

Unknown Values (1)

```
DECLARE @Lastname varchar(15) = 'Chen';
SELECT [m].*
FROM [dbo].[Member] AS [m]
WHERE [m].[LastName] = @Lastname;
```

```
DBCC SHOW_STATISTICS
('Member', 'MemberName')
WITH DENSITY_VECTOR;
```

	All density	Average Length	Columns
1	0.03846154	5.6154	lastname
2	0.0001	16.5526	lastname, firstname
3	0.0001		
4	0.0001		

All density * rows = average
0.03846154 * 10000 = 384.615

Clustered Index Scan (Clustered)	
Scanning a clustered index, entirely or only a range.	
Physical Operation	Clustered Index Scan
Logical Operation	Clustered Index Scan
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Actual Number of Rows	385
Actual Number of Batches	0
Estimated I/O Cost	0.107569
Estimated Operator Cost	0.118726 (100%)
Estimated Subtree Cost	0.118726
Estimated CPU Cost	0.011157
Estimated Number of Executions	1
Number of Executions	1
Estimated Number of Rows	384.615
Estimated Row Size	189 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	False
Node ID	0
Predicate	
[Credit].[dbo].[member].[lastname] as [m].[lastname]=	
[@Lastname]	
Object	
[Credit].[dbo].[member].[member_idnt] [m]	



Slide 151

<http://www.SQLskills.com>

© SQLskills Online Immersion Event. All rights reserved.



Each of these next few slides

corresponds to a portion of the demo

Unknown Values (2)

```
DECLARE @Lastname varchar(15) = 'Fish';
SELECT [m].*
FROM [dbo].[Member] AS [m]
WHERE [m].[LastName] = @Lastname;
```

```
DBCC SHOW_STATISTICS
('Member', 'MemberName')
WITH DENSITY_VECTOR;
```

	All density	Average Length	Columns
1	0.03846154	5.6154	lastname
2	0.0001	16.5526	lastname, firstname
3	0.0001		
4	0.0001		

All density * rows = average
0.03846154 * 10000 = 384.615

Clustered Index Scan (Clustered)	
Scanning a clustered index, entirely or only a range.	
Physical Operation	Clustered Index Scan
Logical Operation	Clustered Index Scan
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Actual Number of Rows	0
Actual Number of Batches	0
Estimated I/O Cost	0.107569
Estimated Operator Cost	0.118726 (100%)
Estimated Subtree Cost	0.118726
Estimated CPU Cost	0.011157
Estimated Number of Executions	1
Number of Executions	1
Estimated Number of Rows	384.615
Estimated Row Size	189 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	False
Node ID	0
Predicate	
[Credit].[dbo].[member].[lastname] as [m].[lastname]=	
[@Lastname]	
Object	
[Credit].[dbo].[member].[member_idnt] [m]	



Slide 152

<http://www.SQLskills.com>

© SQLskills Online Immersion Event. All rights reserved.

Indirect Index Usage (Set Selectivity)

```
SELECT [m].[LastName], [m].[FirstName],  
       [m].[MiddleInitial], [m].[Phone_no], [m].[City]  
FROM [dbo].[Member] AS [m]  
WHERE [m].[FirstName] LIKE 'Kim%'  
OPTION (QUERYTRACEON 3604, QUERYTRACEON 9204, RECOMPILE);
```

Output to the client ↗

Statistics used ↗

Re-evaluate ↗

- Table scan (always an option)
- No indexes exist for SEEKING (no index with first name as the high-order element)
- What about scanning the NC index which has first names in it?
- What would be the best algorithm?



hidden slide
for the demo

Each of these next few slides
corresponds to a portion of the demo



Slide 153

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

How Do You See Them? (2)

- **Query [sys].[indexes]**
 - Use OBJECT_ID and INDEX_ID as input to STATS_DATE
- **Query [sys].[stats] (preferred)**
 - Shows ALL statistics (index-level, column-level, hypothetical, etc.)
 - Use OBJECT_ID and STATS_ID as input to STATS_DATE
- **Use STATS_DATE ([object_id], [index_id]) in a query**
 - Nice quick way to see JUST the date the statistics were last updated
 - Nice to check periodically and automatically
- **Use [sys].[dm_db_stats_properties] (2008R2 SP2+, 2012 SP1+)**
 - Great for automation routines



hidden slide
for the demo

Each of these next few slides
corresponds to a portion of the demo



Slide 154

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Statement Execution: Statistics Events

Optimization = Compilation



Missing Statistics Event

If **auto_create_stats** is enabled then SQL Server (the QO) will WAIT while statistics are created

Invalidated Statistics Event

- If **auto_update_stats_async** is disabled AND **auto_update_statistics** is enabled then SQL Server will WAIT while statistics are updated
- If **auto_update_stats_async** is enabled then SQL Server will optimize based on the invalidated statistics AND kick off an update (which will be used by subsequent users)
- If both methods for updating statistics are disabled then the query will optimize using the invalidated statistic and a warning will be generated (visible in [xml] showplan)



Slide 155

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

When Are They Created?

- **Automatically**
 - For all Indexes
 - When “auto create statistics” is ON AND when the optimizer thinks that statistics would be a good idea (often when a column is in a SARG or a join and does not have an index with that column as the high-order element)
- **Manually**
 - sp_createstats
 - Using CREATE STATISTICS
- **Tip: Leave Auto Create Statistics ON**



Slide 156

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Manually Creating Statistics

- **For secondary columns of an existing index:**
 - Gives the optimizer more options for using existing indexes
 - Scanning an index for highly selective secondary values
 - Can make index usage more likely for secondary conditions; helping understand set selectivity where creating an index is not ideal:
 - Don't warrant a permanent index because the queries are neither frequent enough or critical enough
 - Aren't selective enough across many values / only some values can benefit
- **For columns in search arguments or joins where some are selective and others aren't (and again, you've decided not to index)**
 - Can help to determine position of a table in a join



Slide 157

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

sp_createstats

```
sp_createstats @indexonly = 'indexonly'  
              , @fullscan = 'fullscan'  
              , @norecompute = 'norecompute'
```

- **@indexonly: only create statistics for secondary columns of indexes**
 - This can help to make non-clustered indexes more useful
- **@fullscan: requires more time but will create more accurate statistics**
 - If off-hours this is a good idea
- **@norecompute: the statistics will NOT get automatically updated as distribution of data changes**
 - Generally not recommended
- **Recommendation:**
`sp_createstats 'indexonly', 'fullscan'`



Slide 158

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

RECOMMENDATION: Updating Statistics

- **Manually: but automated through a job**
 - Executing sp_updatestats
 - Sledgehammer maintenance. Only one row has to have been modified.
 - Ola Hallengren's code
 - <http://ola.hallengren.com/>
 - Use @UpdateStatistics parameter / settings
 - Roll your own?
 - Programmatically evaluate the stat_header data
 - Use sys.dm_db_stats_properties: If 2-5% of the data has changed since last stats update OR (stats_date older than 1 week AND last update was sampled (sampled < rows)) then UPDATE STATS with FULLSCAN
- **Auto update stats: only as a safety measure**
 - As a fallback if your code doesn't catch EVERY statistic
- **Asynchronous update stats**
 - Unlikely to cause a problem



Slide 159

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Summary: How Does Auto-Updating Work?

- **You don't really want auto-updating to be your method of updating statistics**
 - 1) It could happen during production
 - 2) It's likely to be later than desired (causing performance problems along the way)
- **But, LEAVE IT ON (auto_update_statistics) as a safety measure**
- **Auto-updating in all versions prior to 2016**
 - Percentage-based threshold unless trace flag 2371
- **Auto-update in version 2016 and higher**
 - Dynamic auto-updating tied to table size (not percentage); same as having turned on 2371
- **Incremental stats – good, but not great**
 - **Positive:** less data has to be read for statistics updates (especially useful in ever-increasing tables)
 - **Negative:** statistic the optimizer uses for optimization is still limited to 200 steps and can become lossy for earlier partitions



Slide 160

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

What If the Data Changes?

- **Automatically updated statistics**

- If auto update statistics is ON (for both the DB and the statistic)
- Statistics are invalidated when:
 - In versions PRIOR to SQL Server 2016: If a minimum of 500 + 20% of the data changes
 - In SQL Server 2016 OR in prior versions with a TF: the threshold is dynamic and tied to the number of rows in the table (more details coming up)

- **Manually update statistics**

- Executing UPDATE STATISTICS
 - Might want to decrease the frequency of updating for highly volatile tables where distribution isn't changing significantly and you see a lot of "statistics" events
 - Might want to increase the frequency of updating for large table where distribution is changing significantly but you're not reaching 20%
- Consider turning off auto update stats at the statistic-level instead of the database-level (more granular control); use:
 - STATISTICS_NORECOMPUTE on the index definition
 - NORECOMPUTE on the statistics definition



Slide 161

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Auto Update Statistics

 hidden slide
w/extra details

- **SQL Server 7.0**
 - Invalidated when sysindexes.rowmodctr reached
 - Updated when invalidated (yikes!)
- **SQL Server 2000**
 - Invalidated when sysindexes.rowmodctr reached
 - Updated when needed
- **SQL Server 2005**
 - Invalidated when sysrowsetcolumns.rcmodified reached (column modification counter this is both good and bad...)
 - Updated when needed
- **SQL Server 2008+**
 - Invalidated when sysrscols.rcmodified reached
 - Updated when needed

If someone challenges you on the overhead of auto update statistics it could be because of the original (poor) design.



Slide 162

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Asynchronous Statistics Updates

 hidden slide
w/extra details

- By default statistics are updated before query compilation (as part of compilation/optimization) and this can cause a delay in execution
- Can turn “Async Stats Update” on to have the current execution trigger the update but not wait for the update (unlikely to be a problem at ~20%; otherwise, likely to have been a problem earlier)
- **ALTER DATABASE databasename
SET AUTO_UPDATE_STATISTICS_ASYNC ON**
- **BUG:** When you enable the Auto Update Statistics Asynchronously statistics option in a database of Microsoft SQL Server 2012, Microsoft SQL Server 2008, or Microsoft SQL Server 2008 R2, and then you run queries on the database, a memory leak occurs.
- **FIXED:** Cumulative Update 2 for SQL Server 2012 SP1, Cumulative Update 5 for SQL Server 2012, Cumulative Update 4 for SQL Server 2008 R2 SP2
- See KB article: 2778088 <http://support.microsoft.com/kb/2778088>

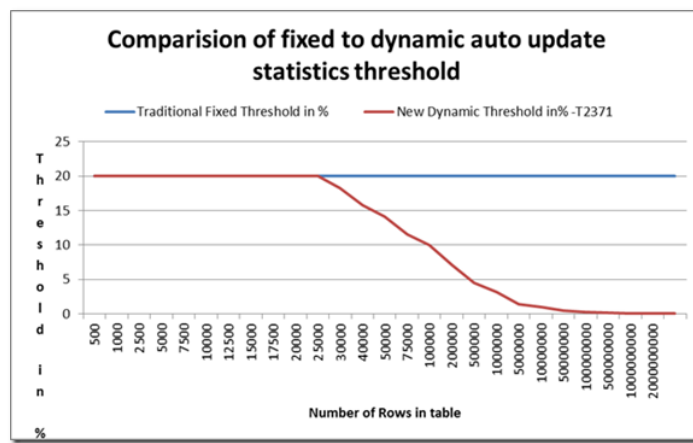


Slide 163

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Dynamic Auto-Updating Threshold

- Server-wide trace flag 2371 prior to SQL Server 2016
- NEW behavior in SQL Server 2016
- Originally released in SQL Server 2008 R2 SP1
- Blogged by:
Juergen
Thomas
(SQLCat)
7 Sep 2011



Slide 164

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Incremental Stats Updates

- **New feature for SQL Server 2014 partitioned tables**
- **Statistics update triggered when 20% of the partition changes**
 - Build partition-level statistics
 - Compress table-level statistics
 - Merge partition-level statistics in with table-level statistics
 - Resulting table-level statistics are used for estimation
- **Positive: statistics updates are triggered a lot earlier for partitions that are updates (especially useful in ever-increasing tables)**
- **Positive: less data has to be read for statistics updates (especially useful in ever-increasing tables)**
- **Negative: statistic the optimizer uses for optimization is still limited to 200 steps and can become lossy for earlier partitions**
 - Article: SQL Server 2014 Incremental Statistics on SQLperformance.com
 - <http://bit.ly/1uhEbr1>



Slide 165

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Review: Plan Problems Related to Statistics

- Statement selectivity
- What kinds of statistics exist
- How does SQL Server use statistics
- Creating additional statistics
- Updating statistics



Slide 166

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Questions!



Review: What To Do Next?



What to do next?

- **Lots of content...**
 - Reviewing sooner rather than later will help!
 - Using a mix of mediums will help you see things in different ways and likely for some of you one will be better than another – multiple will be the way to make it stick!
- **A possibly review strategy?**
 - Print the slides + the whiteboard
 - Watch the course more interactively while taking notes and playing with demos
 - Small chunks – a module every day or two for a couple of weeks
 - Go through the recommended videos
 - Offer some “short courses” or “brown bag” lunch sessions where you deliver 30-45 minute topics to team members

If you want to learn something, read about it. If you want to understand something, write about it. If you want to master something, teach it.
– Yogi Bhajan
 - Always keep learning!



Slide 169

<http://www.SQLskills.com>
 © SQLskills Online Immersion Event. All rights reserved.


PLURALSIGHT

- **Get that Pluralsight code from Paul!**
- **Then, watch this course first**
SQL Server: Optimizing Ad Hoc Statement Performance
 - <http://pluralsight.com/training/Courses/Description/sqlserver-optimizing-adhoc-statement-performance>
 - Just over 7 hours on ad hoc statement execution and plan caching
- **Then, watch this course**
SQL Server: Optimizing Stored Procedure Performance (Part 1)
 - <http://pluralsight.com/training/Courses/Description/sqlserver-optimizing-stored-procedure-performance>
 - Just over 7 hours on procedure caching and recompilation
- **Then, watch this course**
SQL Server: Optimizing Stored Procedure Performance – Part 2
 - <https://www.pluralsight.com/courses/sqlserver-optimizing-stored-procedure-performance-part2>
 - Just under 4 hours on session settings and caching



Slide 170

<http://www.SQLskills.com>
 © SQLskills Online Immersion Event. All rights reserved.

Plan Caching and Recompilation Resources

- **Plan Caching and Recompilation in SQL Server 2012**
 - <http://msdn.microsoft.com/en-us/library/dn148262.aspx>
- **Plan Caching in SQL Server 2008**
 - <http://msdn.microsoft.com/en-us/library/ee343986.aspx>
- **Batch Compilation, Recompilation, and Plan Caching Issues in SQL Server 2005**
 - <http://www.microsoft.com/technet/prodtechnol/sql/2005/recomp.msp>
- **PSS SQL Server Engine Blog**
 - <http://blogs.msdn.com/psssql/default.aspx>
- **KB 243586: Troubleshooting Stored Procedure Recompilation**



Slide 171

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Plan Cache Pollution Resources

- **Blog posts:**
 - [Plan cache and optimizing for adhoc workloads](#)
 - [Plan cache, adhoc workloads and clearing the single-use plan cache bloat](#)
 - [Clearing the cache - are there other options?](#)
 - [Statement execution and why you should use stored procedures](#)
 - Category: Optimizing Procedural Code
 - <http://www.sqlskills.com/BLOGS/KIMBERLY/category/Optimizing-Procedural-Code.aspx>
- **MSDN Article: How Data Access Code Affects Database Performance, Bob Beauchemin**
 - <http://msdn.microsoft.com/en-us/magazine/ee236412.aspx>



Slide 172

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Optimizing Procedural Code on PASStv

- **Start with this blog post (and sample code):**
 - SQLskills SQL101: Stored Procedures
 - <http://www.sqlskills.com/blogs/kimberly/sqlskills-sql101-stored-procedures/>
 - THEN, Building High Performance Stored Procedures
 - <http://www.sqlskills.com/blogs/kimberly/high-performance-procedures/>
- **Then, watch my PASS Summit 2014 presentation through PASStv**
 - <http://www.sqlpass.org/summit/2014/passtv.aspx>
 - See, Day 1: 4:30pm
 - Dealing with Multipurpose Procs and PSP the RIGHT Way! - Kimberly Tripp
- **Then, check out this post: Stored Procedure Execution with Parameters, Variables, and Literals**
 - <https://www.sqlskills.com/blogs/kimberly/stored-procedure-execution-with-parameters-variables-and-literals/>



Slide 173

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

Statistics Resources

- **Whitepaper: [Statistics Used by the Query Optimizer in Microsoft SQL Server 2008](#)**
- **2013 – PASStv from PASS Summit: VLDBs and Data Skew (i.e. filtered statistics):**
<http://www.sqlpass.org/summit/2013/PASStv.aspx?watch=li5HwaZF8tc>
(NOTE: skip ahead 4 mins because they didn't tell me that they were *already* recording BEFORE the official session start time... and, well, the first 4 mins are not related to SQL)
- **Joe Sack's whitepaper: *Optimizing Your Query Plans with the SQL Server 2014 Cardinality Estimator*: <https://msdn.microsoft.com/en-us/library/dn673537.aspx>**



Slide 174

<http://www.SQLskills.com>
© SQLskills Online Immersion Event. All rights reserved.

THANK YOU!

