

SQLIntersection

PRECON05

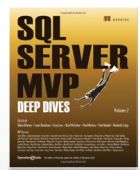
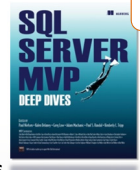
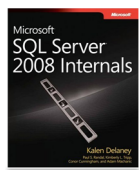
Queries Gone Wild: Real-world Solutions

Kimberly L. Tripp
President/Founder/Co-owner
Kimberly@SQLskills.com



Author/Instructor: Kimberly L. Tripp

- Consultant/Trainer/Speaker/Writer
- President/Founder, SYSolutions, Inc.
 - e-mail: Kimberly@SQLskills.com
 - blog: <http://www.SQLskills.com/blogs/Kimberly>
 - Twitter: @KimberlyLTripp
- Author/Instructor for **in-person** SQL Server Immersion Events
- Author/Instructor for **online** training through Pluralsight
- Instructor for multiple rotations of both the SQL MCM & Sharepoint MCM
- Author/Manager of SQL Server 2005 & 2008 Launch Content
- Author/Speaker at Microsoft TechEd, PASS, ITForum, TechDays, ...
- Author of several SQL Server Whitepapers on MSDN/TechNet: Partitioning, Snapshot Isolation, Manageability, SQLCLR for DBAs
- Author/Presenter for more than 25 online webcasts on MSDN and TechNet
- Co-author MSPress Title: SQL Server 2008 Internals, the SQL Server MVP Project (1 & 2), and SQL Server 2000 High Availability
- Presenter/Technical Manager for SQL Server HOL DVDs
- Owner and Manager of the SQLIntersection conference



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

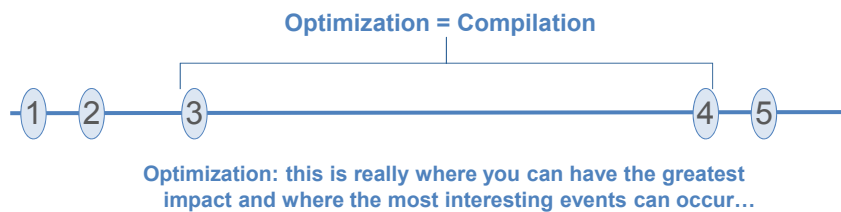
Workshop Overview

- **Query Optimization**
 - Understanding cardinality and selectivity; how SQL Server "predicts" data sets
 - When statistics aren't available, what happens (heuristics)?
- **Statistics**
 - System-created vs. user-defined vs. hypothetical indexes
 - When are they created vs. when are they updated (full-scan vs. sampling)
- **Data Distribution**
 - Evenly distributed data (yeah, right)
 - Problems with uneven distribution
- **Plan Caching**
 - Statement Execution
 - Simple Parameterization vs. Forced Parameterization
- **Stored Procedures**
 - Creation/Optimization
 - Recompilation

3

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

Statement Execution Simplified



1. Parse
2. Standardization/normalization/algebrization
⇒ Query tree (not T-SQL anymore)
3. Cost-based optimization (statistics are used to come up with an optimal plan as well as other things)
4. Compilation
5. Execution



4

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

Cost-Based Optimization

- **Find a reasonable subset of possible algorithms to access data based on:**
 - The query *Sometimes a rewrite helps...*
 - Any joins *Sometimes a derived table (subquery in the FROM clause)...*
 - Any SARGs *Your SARGs need to be well-defined...*
 - 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)**



5

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

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?*



6

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

Demo

Estimates

Estimates and Selectivity
How does SQL Server “know”?



Demo: Key Points

- **Estimates come from:**
 - Statistics – if they exist or if they can be [auto] created, using:
 - The histogram: when the value can be “sniffed” (parameters)
 - The density vector: when the value cannot be “sniffed” (variables)
 - Heuristics – if there are no statistics available and SQL Server cannot auto create them
 - These are internal “magic” numbers (cannot be changed)
 - They often result in very poor plans (LEAVE AUTO_CREATE_STATS ON)
 - Sometimes this is the only option when better estimations cannot be done (comparison between columns [e.g. col1 > col2])
- **Statistics have to be reasonably small to be fast/useful**
- **They’re just estimates**
 - They’re not always guaranteed to be accurate
 - They’re just meant to get us closer to the right value



Workshop Overview

- Query Optimization
- Statistics
 - What do they look like? How can you analyze them?
 - System-created vs. user-defined vs. hypothetical indexes
 - DBCC AUTOPILOT
 - When are they created vs. when are they updated (full-scan vs. sampling)
 - sp_createstats
 - Auto updates, asynchronous updates
 - Dynamic threshold based updates (trace flag 2371)
- Data Distribution
- Plan Caching
- Stored Procedures

9

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

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

10

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

What Are They Telling You?

- **High-order element (LastName)**
 - Histogram (#4) has great details for this column
 - Anderson 385 rows
 - Barr 385 rows
- **Secondary columns – DENSITY (#3)**
 - Always LEFT-based
 - Rows * All density = estimate of rows
 - Based on even distribution
- **Density information**
 - Density for LastName
 - 10,000 Rows * 0.03846154 = 384.6154
 - Density for LastName, FirstName combo
 - 10,000 Rows * 0.0001 = 1



11

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

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

Index Name	AUTOSTATS	Last Updated
[member_ident]	ON	2008-08-26 17:18:12.593
[member_corporation_link]	ON	2008-08-26 17:18:12.673
[member_region_link]	ON	2008-08-26 17:18:12.793
[MemberName]	ON	2008-10-29 11:13:29.220
[_WA_Sys_00000003_0CBAE877]	ON	2008-10-29 11:28:32.313



12

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

How Do You See Them? (2)

- **Query sys.indexes**
 - Use object_id and index_id as input to stats_date
- **Query sys.stats**
 - 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+)**



13

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

What Kinds of Statistics Exist?

- **Auto-created statistics**
 - Named _WA_SYS_
 - Blog: [How are auto-created column statistics names generated?](#) (Paul Randal)
 - 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"



14

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

DBCC AUTOPILOT

- Check out the Simple Talk article: [Hypothetical Indexes on SQL Server](#)
 - by [Fabiano Amorim](#)

```
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 (reviewing showplan)
GO
SET AUTOPILOT OFF
```

How Do You See Them? (3)

- Programmatically pull tabular sets from DBCC SHOW_STATISTICS ...
 - STAT_HEADER
 - Number of rows v. rows sampled and number of steps
 - Last updated date
 - DENSITY_VECTOR
 - Left-based density subsets
 - Averages given left-based combinations of index key
 - STAT_HEADER JOIN DENSITY_VECTOR
 - Presents the details from STAT_HEADER and DENSITY_VECTOR as a single row
 - All Density = $\text{TABLE_CARDINALITY} / \text{TUPLE_CARDINALITY}$
(for each left-based subset starting ending at that ordinal position)
 - 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

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 an 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



17

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

Manually Creating Statistics

- For secondary columns of an index: can make SCAN choices more likely for highly selective secondary conditions that don't warrant an index (where only some values are selective and where query frequency doesn't warrant the additional index)
- For columns in search arguments and joins where some are selective and others aren't (and again, you've decided not to index)



18

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

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'
```



19

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

What If the Data Changes?

- **Automatically updated!**
 - If auto update statistics is ON (for both the DB and the statistic – didn't set "norecompute")
 - If roughly 500 + 20% of the data changes
 - Complete details in TechNet whitepaper
- **Manually update statistics**
 - 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 (and less than 20% change)
 - If so, consider turning off auto update stats at the statistic-level by adding STATISTICS_NORECOMPUTE on the index definition or NORECOMPUTE on the statistics definition
 - This is a lot better and offers more granular control than turning it off at the database level
 - Executing UPDATE STATISTICS



20

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

Auto Update Statistics

- **SQL Server 7.0**
 - Invalidated when sysindexes.rowmodctr reached
 - Updated when invalidated
- **SQL Server 2000**
 - Invalidated when sysindexes.rowmodctr reached
 - ✓ Updated when needed
- **SQL Server 2005**
 - Invalidated when sysrowsetcolumns.rcmodified reached
 - ✓ Updated when needed
- **SQL Server 2008**
 - Invalidated when sysrscols.rcmodified reached
 - Updated when needed



21

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

Statement Execution: Statistics Events



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 created
- 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)



22

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

Asynchronous Statistics Updates

- 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 (but you could get into a vicious cycle where you optimize with old statistics, trigger the update and by the time you execute again, they’re invalid)

```
ALTER DATABASE databasename  
SET AUTO_UPDATE_STATISTICS_ASYNC ON
```

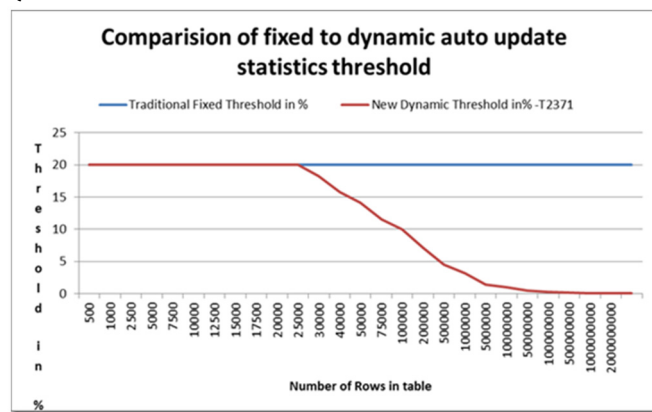


23

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

Dynamic Auto Updating Threshold

- Server-wide trace flag 2371
- Released in SQL Server 2008 R2 SP1
- Blogged by:
Juergen
Thomas
(SQLCat)
7 Sep 2011



24

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

Updating Statistics

- **Manually: but automated through a job**
 - Executing sp_updatestats
 - Sledgehammer maintenance. Only one row has to have been modified to execute this... not very granular
 - Ola Hallengren's code
 - <http://ola.hallengren.com/>
 - Roll your own?
 - Programmatically evaluate the stat_header (just an idea)
 - If stats_date older than 1 week OR 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



25

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

Where Are We At?

- **Generally, statistics are helpful**
- **The better your statistics, the more likely you'll have better plans**
- **You need to understand where statistics will and will not be used**
 - And, unfortunately, I have more bad news/information about this and filtered objects...
- **The rest of this section I'm going to focus on where/when statistics start to fall short:**
 - Sampling
 - The Histogram
 - Estimations for multiple columns



26

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

Workshop Overview

- **Query Optimization**
- **Statistics**
- **Data Distribution**
 - Concerns around sampling (to generate statistics)
 - Concerns around the histogram (limitations)
 - Data Distribution
 - Assessing level of data skew
 - Filtered statistics
 - Automating the creation of filtered statistics
 - Problems with filtered objects (interval subsumption and query execution)
 - Concerns Around Estimations Between Columns?
- **Plan Caching**
- **Stored Procedures**

27

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

Concerns Around Sampling

- Using showplan tooltip – estimate v. actual rows
- If query performance is poor AND the actual is significantly OFF from the estimate then you might want to verify the statistics creation (rows v. rows scanned)
- If statistics were based on a sampling and performance is improved after statistics have been updated, then you might want to turn off auto update for this index (using STATISTICS_NORECOMPUTE) and schedule an update statistics with FULLSCAN

UPDATE STATISTICS ...
WITH FULLSCAN



28

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

Concerns Around the Histogram

- **Stores ACTUAL values from the FIRST (and only first) column of the key**
 - Sometimes referred to as the leading column of the index
 - Sometimes referred to as the high-order element of the index
- **Never stores more than 201 steps (up to 200 distinct/actual values plus 1 NULL values row – if the column allows NULLs)**
- **Even when your table has more than 200 distinct values, you may have fewer than 200 steps**
- **Values chosen aren't evenly distributed**
 - Internally, during statistics creation, SQL compresses steps to make room for more. They try to track "interesting" values/anomalies but the more compression that occurs, the more they lose outliers
- **Has the best information – but how good is it?**
 - This is really the issue. And, unfortunately, it depends – mostly on how skewed the data distribution is...



Important: Describes the entire table... even when partitioned

29

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

Data Distribution Matters

- **Even distribution is easy**
 - Think about "line items per sales order"
 - That's probably *fairly* consistent at 2 or 3
- **Un-even distribution is HARDER**
 - Think about sales per product
 - This is all over the place – and varies over time as well...
 - Think about sales per customers
 - Again, all over the place – and, again, varies over time...
- **The histogram does a MUCH better job having steps and average distribution per step but what if there are well over 200 distinct values (10s of thousands) and millions of rows with heavy skew between steps?**
 - Simply put, the averages just aren't going to cut it anymore...



30

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

Demo

Understanding the Histogram

What information is stored in the histogram?
How does SQL Server use it?



Demo: Key Points

- **Very useful to help the optimizer assess the usefulness of an existing index (whose histogram is not as accurate due to skew [and probably table size])**
- **Take a *slightly* skewed example: sales by customer**
 - Sales: 30,923,776
 - Customers: 18,484
 - Average = $30,923,776 / 18,484 = 1,673$
 - All density (from density_vector of statistics) = $5.410084E-05$ multiplied by rows = 1,673
 - Skew $\Rightarrow$ high: 30K+ (only ~6), Low: ~500 (thousands)
 - Not every value can possibly be represented in 200 steps (for histogram)
 - Occasionally, they'll be wrong – average might not be good enough... enter $\Rightarrow$ filtered statistics
- **Not an index, just a statistics blob...**



(smaller, easier to maintain, more accurate over that set)

Assessing Level Of Data Skew

- Can we access the histogram – programmatically?

```
INSERT TemporaryWorktable
EXEC ("DBCC SHOW_STATISTICS ... WITH HISTOGRAM")
```

Should be easy enough? (er, famous last words)

- RANGE_HI_KEY (if you want to analyze it against the base table, needs to be the same data type as the base table)
- LOTS of dynamic string execution and different lookups to reconstruct data types (including collations)
 - Which, by the way – has a slightly different syntax when in a CAST clause vs. in a column definition

- End result: [sp_SQLskills_AnalyzeColumnSkew]

- Plus, [sp_SQLskills_AnalyzeAllLeadingIndexColumnSkew]

- Both need to be added to master and marked as system objects:

```
EXEC [sys].[sp_MS_marksystemobject]
'sp_SQLskills_AnalyzeAllLeadingIndexColumnSkew'
```



33

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

Assessing Level of Data Skew

- Parameters and uses for this:

```
[sp_SQLskills_AnalyzeColumnSkew]
@schemaName sysname = NULL
, @objectName sysname = NULL
, @columnName sysname = NULL
, @difference int = 1000
-- Min diff between average and largest difference in that step
, @factor decimal(5, 2) = 2.5
-- Min factor of the difference against the average
, @numofsteps tinyint = 10
-- Min number of steps that have to meet this/these
, @percentofsteps tinyint = 10
-- Minimum PERCENT of steps that have to meet this/these
, @keeptable char(5) = 'FALSE'
-- If TRUE keeps ALL of the worktables
, @tablename varchar(520) = NULL OUTPUT
-- Fully delimited name of the worktable in tempdb
```

- Lots of options and relatively low impact (requires an index that has the column specified [@columnName] as the leading column)



34

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

Demo

Skewed Data

Understanding uneven distribution
Using statistics to determine skew



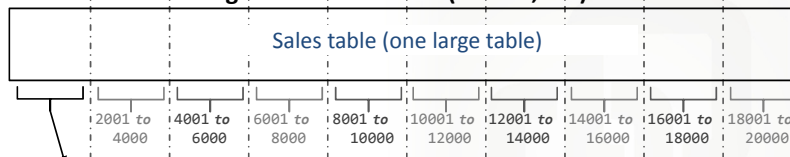
Demo: Key Points

- **Not everyone will have skewed data**
 - **Might already suspect you have a problem**
 - Poor estimates where you've tried:
 - Updating statistics more frequently
 - Updating statistics with FULLSCAN more frequently
 - Adding OPTION (RECOMPILE) to your code
 - But... it still didn't work – the estimates were still *off*
- NOTE: There are still other things it could be. But, here's where this proc can really help you to understand what your data really looks like!
- **Target only specific (probably large [over 50-60GB]) tables**
 - NOTE: It's not the size of the table that's important here – even a small table that's 100-200MB can show signs of skew. But, the performance problems that result from a small table just aren't as noticeable.
 - **Don't forget to clean up the worktables in tempdb:**
EXEC [sp_SQLskills_HistogramTempTables] @management = 'DROP'



Filtered Statistics For The Entire Table? (1 of 2)

- **We've determined there's skew – now what?**
 - Manually create a whole bunch of filtered stats?
 - How do we break down the data?
 - How do we account for new rows?
 - How do we maintain this?
- **Here's the idea – imagine 20K customers (1 to 20,000)**



```
CREATE STATISTICS FilteredStat
ON [schemaname].[tablename] ([columnname])
WHERE [columnname] >= 1
AND [columnname] < 2000
```



37

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

Filtered Statistics For The Entire Table? (2 of 2)

- **So, that seemed simple**
 - Not all data is as simple as an ever-increasing integer
 - Most data actually changes... (what happens with new rows over 20K)
 - How would we automate this process?
- **End result:**
[sp_SQLskills_CreateFilteredStats]
- **Needs to be added to master and marked a system object:**
EXEC [sys].[sp_MS_marksystemobject] 'sp_SQLskills_CreateFilteredStats'
- **Requires a couple of other procedures:**
 - [sp_SQLskills_CreateFilteredStatsString]
 - [sp_SQLskills_DropAllColumnStats]



38

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

Automating Filtered Statistics Creation

- Parameters and uses for this:

```
[sp_SQLskills_CreateFilteredStats]
    @schemaname sysname = NULL
    , @objectname sysname = NULL
    , @columnname sysname = NULL
    , @filteredstats tinyint = 10
    -- Number of filter chunks to define
    , @everincreasing bit = 0
    -- depending on whether or not your table is ever increasing
    , @maxforfs sql_variant = NULL
    -- Projected value to extend out until the next redistribution
    , @fullscan varchar(8) = NULL
    -- Generate the statistic using a FULLSCAN or SAMPLE
    , @samplepercent tinyint = NULL
    -- If @fullscan = SAMPLE, do you want to override SQL's sample
```

- Lots of options and relatively low impact to create (will leverage an existing index; fullscan might require that index to end up in cache)
- Might not give you any benefits and it will need to be maintained!



39

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

Demo

Filtered Statistics

What are they?

What are the benefits?



Demo: Key Points

- **Must supply values, cannot use only the defaults**
- **You'll need to estimate a new max value based on:**
 - How long you want these filtered statistics to live
 - How much data churn/change you see in these values
- **When this is FIRST run, All column-level statistics on the column you're about to create filtered statistics on – are deleted. This shouldn't be a problem because:**
 - We're only targeting leading index columns (so, SQL can use the index stats; the column-level were redundant already)
- **If you want to clean up the newly created filtered stats:**

```
EXEC [sp_SQLskills_DropAllColumnStats]
    @schemaname= N'dbo'
    , @objectname= N'factinternetsales'
    , @columnname= N'customerkey'
    , @dropall= N'TRUE'
```



41

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

How Many Filtered Indexes/Stats? Per Table

 *hidden slide
w/extra details*

- **SQL Server 2000**
 - Columns: 1,024
 - Nonclustered indexes: 249
 - Statistics: 249 (shared with nonclustered)
- **SQL Server 2005**
 - Columns: 1,024
 - Nonclustered indexes: 249
 - Statistics: 2,000 (referenced in sys.stats)
- **SQL Server 2008/R2 & 2012**
 - Columns: 30,000
 - Nonclustered indexes: 999
 - Statistics: 10,000 (30,000 in R2)



42

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

Valid Index IDs

- **Table always has at least one index id**
 - For a heap the index id is always 0
 - For a clustered table the index id is always 1
 - Nonclustered indexes *can* start at 2
- **999 nonclustered indexes per table**
 - Index ids start with 2 but statistics use the same counter range
 - Index ids 251 through 255 are reserved (from prior use cases)
 - NOTE: 255 was used in earlier releases but 251-254 have never been used (at least not that I know of ☺)
- **30,000 statistics per table (SQL Server 2008 R2+)**
- **Index and statistics ids run from 2 to 250 and from 256 to 31005**
 - NOTE: The datatype used is int (sys.indexes.index_id and sys.stats.stats_id). Technically, they *could* support more...

This Is So Cool...

But, it won't always work... sigh.

- **Session setting requirements for filtered indexes do NOT apply to the creation OR usage of filtered statistics**
- **Interval subsumption problems**
 - Might be able to do a query rewrite but this is kind of a nightmare
- **Might be able to add recompile**
- **Might be able to add a plan guide**
- **To be honest, there are better – architectural ways – to handle very large tables with skew problems**
 - Don't have very large tables
 - Create multiple tables and union them together as a view
 - Partitioned Views (UNION ALL and constraints)

Session Settings Requirements

APPLY only to Filtered Indexes (NOT Filtered Statistics)

- See BOL topic: Set Options that Affect Results
- Session settings control behavior – and the result of some computations
- Data in these persisted structures must be consistent (which is why these apply to filtered indexes)
- Session settings that must be on:
 - ANSI_NULLS
 - ANSI_WARNINGS
 - QUOTED_IDENTIFIER
 - CONCAT_NULL_YIELDS_NULL
 - ANSI_PADDING
 - ARITHABORT
- Session setting that must be off:
 - NUMERIC_ROUNDABORT

Msg 1934, Level 16,
State 1, Line 1
CREATE INDEX failed because the following SET options have incorrect settings: 'QUOTED_IDENTIFIER'. Verify that SET options are correct for use with indexed views and/or indexes on computed columns and/or filtered indexes and/or query notifications and/or XML data type methods and/or spatial index operations.



45

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

Client Consistency Requirements

APPLY only to Filtered Indexes (NOT Filtered Statistics)

- Consistency with table(s), view and the clustered index (on the view) creation OR table and the filtered index
 - All tables on which the view is based, the view itself and the index must be created with the correct session settings set or the index cannot be created on the view
- Consistency with base table access
 - All INSERT, UPDATE and DELETE statements must be executed with correct session settings or the insert, update or delete will fail
- Consistency with query access
 - All queries that SELECT against views with indexes (or tables with filtered indexes) must access them with the correct session settings set otherwise the data will need to be recalculated, rejoined or recomputed



46

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

Interval Subsumption (1 of 2)

Filter use (per query)

- **Filters (indexes/statistics) cannot be used when the query's predicate is not a subset of a SINGLE filter interval:**
 - Filtered index: January data
 - WHERE [date] >= '20110101' AND [date] < '20110201'
 - Filtered index: February data
 - WHERE [date] >= '20110201' AND [date] < '20110301'
 - Predicates and their usage:
 - WHERE [date] = '20110115' **YES**
 - WHERE [date] BETWEEN '20110105' AND '20110115' **YES**
 - WHERE [date] BETWEEN '20110115' AND '20110215'**NO, cannot use multiple filtered objects**
- **BAD NEWS:** This is why there's no such thing as partition-level statistics. SQL Server cannot combine the filtered indexes and use them individually. So, you need a table-level index (that's partition-aligned) but the statistics are less accurate.



47

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

Interval Subsumption (2 of 2)

Filter use (per query)

- **Filtered statistics examples:**
 - Filtered statistic over a range:
 - WHERE ([CustomerKey]>= 11000 AND [CustomerKey] < 11566)
 - Filtered statistic over a range:
 - WHERE ([CustomerKey]>= 11566 AND [CustomerKey] < 12363)
 - Predicates and their usage:
 - WHERE [c].[CustomerKey] IN (11509, 11503, 11123) **YES**
YES, all values are in only ONE filtered statistic
 - WHERE [c].[CustomerKey] IN (11509, 12345)
NO, cannot use multiple filtered objects
WORKAROUND: Use dynamic string execution to UNION ALL the individual values
 - Predicates and their usage:
 - WHERE [c].[CustomerKey] BETWEEN 11500 AND 11600
NO, cannot use multiple filtered objects
GOOD NEWS: The larger the range, the less you need to use the more accurate values (averages are fine when they're over larger ranges)



48

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

Demo

Interval Subsumption

When won't SQL Server use a filtered object?
Is there a workaround?



More Bad News...

- **Filtered indexes/filtered stats: stats may get HORRIBLY out of date:**
 - SQL Server ONLY tracks a colmodctr NOT related to the filtered set or size. As a result, the stats (even filtered stats) are only updated when 20% of the table has been modified...
 - For better performance, you need to automate and control their updates (do NOT rely on "auto update statistics")
 - Good news is that they're relatively small and creating a job to automated them is relatively easy!
 - See this blog post: <http://www.sqlskills.com/BLOGS/KIMBERLY/post/Filtered-indexes-and-filtered-stats-might-become-seriously-out-of-date.aspx> (<http://bit.ly/1knEE2>)
- **Forced parameterization (the database option) will generalize statements and many filters won't be eligible during optimization**
- ***Similar summary, don't go wild with this feature; it has powerful – but specific – uses!***



Concerns Around Estimations Between Columns?

- **Scenario**
 - You have 90 rows in a table
 - 10 rows have a value of x for column 6
 - Where (physically) in this table are these 10 rows?
- **They could be evenly distributed throughout the table...**



- **But, what if they're not?**



51

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

Scenario: Unevenly Distributed Data

- **Scenario**
 - AdventureWorksDW2012: FactInternetSales has 60,398
 - 2,541 rows have a null for ShipDateKey
 - $60,398 / 2,541 = 23.7$ (1 in 23.7 rows is NULL)
 - Nonclustered index on ShipDateKey
 - Nonclustered index on OrderDateKey
- **What does SQL Server do?**

```
SELECT MIN(fis.OrderDateKey)
FROM dbo.FactInternetSales2 AS fis
WHERE fis.ShipDateKey IS NULL
```



52

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

Even Distribution: Always Even??

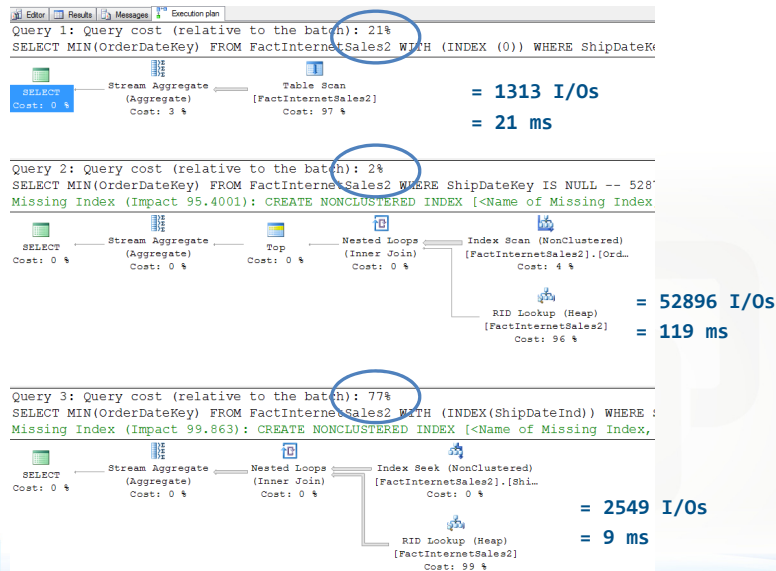
- **Table scan is always an option**
- **Use an index on ShipDateKey to look up the actual date for all orders where ShipDateKey is NULL**
 - This means a bookmark lookup must be run for EVERY NULL so that we can get the OrderDateKey
 - The worktable then needs to be sorted to find the lowest order date
- **Use an index on OrderDateKey as only 1 on 23.7 sales have a NULL for ShipDateKey**
 - SQL Server estimates that they'll find a NULL within 23.7 rows and they won't need a worktable
 - This sounds better...
- **But the rows are not evenly distributed!**



53

© SQLintersection. All rights reserved.
http://www.SQLintersection.com

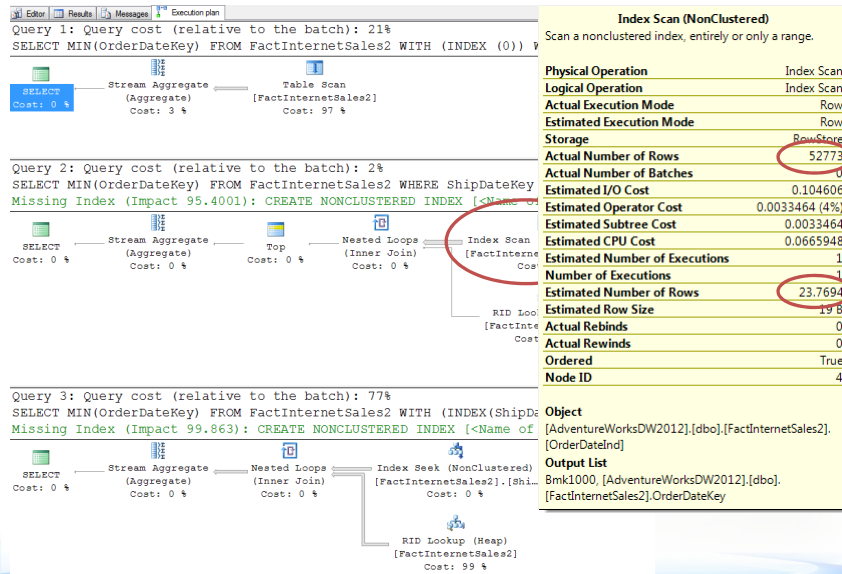
Evenly Distributed Data??



54

© SQLintersection. All rights reserved.
http://www.SQLintersection.com

Evenly Distributed Data??



55

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

Always Better: Indexes

- **Statistics cannot be used to directly access data but:**
 - They can only HELP the optimizer determine the best plan
 - They can help determine best join order
- **Statistics are just estimates: they help MOST of the time but:**
 - There are limitations
 - They take time to create, update, store...
- **Indexing can often be A LOT better...**

```
CREATE INDEX ShipDateOrderDateInd_SeekableForMin
ON FactInternetSales2 (ShipDateKey, OrderDateKey)
```



56

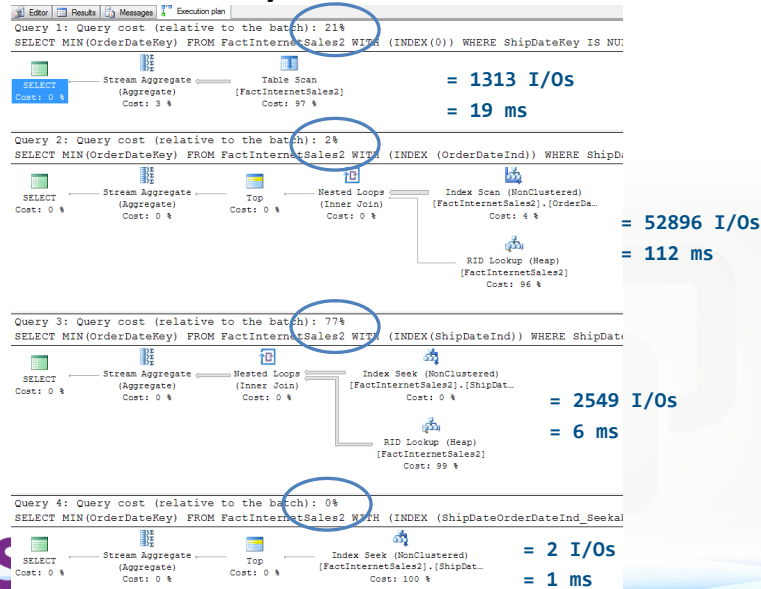
© SQLintersection. All rights reserved.
<http://www.SQLintersection.com>

SQLintersection

Orlando, Florida – April 2014

www.SQLintersection.com

Always Better: Indexes



Always Better: The RIGHT Indexes

- Every SINGLE QUERY can be indexed to make THAT SINGLE query fast
- You can't index EVERY query unless you're read-only/decision support, and even then you're limited by disk space (but that's about it...lots of indexes – yeah!)
- But a better choice is prioritizing and determining which queries really need indexes!

Workshop Overview

- Query Optimization
- Statistics
- Data Distribution
- Statement Execution and Plan Caching
 - Literals/Parameters/Variables
 - Parameter sniffing
 - Adhoc vs. sp_executesql
- Stored Procedures

59

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

How/When Does the Estimate Happen?

(1 of 3)

- A few concepts about statement execution
- Statements can have:
 - Literals
 - Variables
 - Parameters
- Depending on how you write and execute your statements – the processing and analyzing your queries for their distribution, might vary
 - Some statements can “sniff” the values
 - Some statements cannot
- Adhoc statements are most likely going to be able to “sniff” the values but there are exceptions

60

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

How/When Does the Estimate Happen? (2 of 3)

- **Estimation comes from “sniffing” the value**
 - **Result:** estimate comes from HISTOGRAM
 - **Pro:** estimate is usually more accurate
 - **Pro:** *that* execution gets a plan designed for *that* value
 - **Con:** If/when this plan is saved – subsequent executions are prone to “parameter sniffing problems” (PSP)
- **Value cannot be “sniffed”**
 - **Result:** estimate comes from the DENSITY_VECTOR
 - **Depends:** estimate is an “average”
 - **Depends:** the plan generated is designed for the “average” value – not *that* value
 - **Pro:** If/when this plan is saved – subsequent executions are NOT prone to PSP
 - **Con:** When your data is NOT [relatively] evenly distributed, this plan might not be good for anyone



61

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

How/When Does the Estimate Happen? (3 of 3)

- **Adhoc statements**
 - Simple parameterization – almost all statements will be compiled just for that execution; they will not be parameterized/saved (see rules for parameterization in whitepaper)
 - Forced parameterization – most statements will be parameterized/saved (you’ll see this in decreased CPU/compilations and potential for PSP)
- **sp_executesql (or, prepared statements)**
 - A fantastic way to reduce the CPU/cache overhead that adhoc has but should only be used when a plan is stable and consistent
- **Stored procedures**
 - Can be “sniffed” but there are exceptions to what SQL Server will store in their plans
 - Special considerations for when these items are IN stored procedures:
 - variables/parameters
 - dynamic strings in stored procedures
 - sp_executesql in stored procedures



62

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

Why Do We Care About This?

- **Filtered indexes and filtered statistics cannot be used by forced statement caching, prepared statements, or stored procedures – without either:**
 - Forcing a recompile (**when you can change the code**)
 - Best way: add OPTION (RECOMPILE) on the end of the statement that needs the filtered statistic
 - Creating a plan guide (**when you can't change the code**)
 - Type of plan guide depends on what you're guiding...
 - Procedure -> object plan guide
 - Adhoc statement -> SQL plan guide
 - Adhoc statement when the database is FORCED parameterization then use a plan guide template (changing the statement that needs the filtered object to SIMPLE)



63

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

Demo

Plan Guides

- Object plan guide
- SQL plan guide
- Plan guide template



Demo: Key Points

- **There's always a way to get SQL Server to do what you want!**
- **Understanding statement execution, parameter sniffing, lots of things that change the behavior**
 - Database option for PARAMETERIZATION: SIMPLE | FORCED
 - How you're executing the statement
 - Adhoc
 - `sp_executesql` (or, prepared from the app)
 - In a procedure
 - With parameters
 - With variables
 - As a dynamic string (using `EXEC (@execstring)` -> acts like and works like adhoc (so, the database option will affect it)
 - Using `sp_executesql` (works as above)



65

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

Workshop Overview

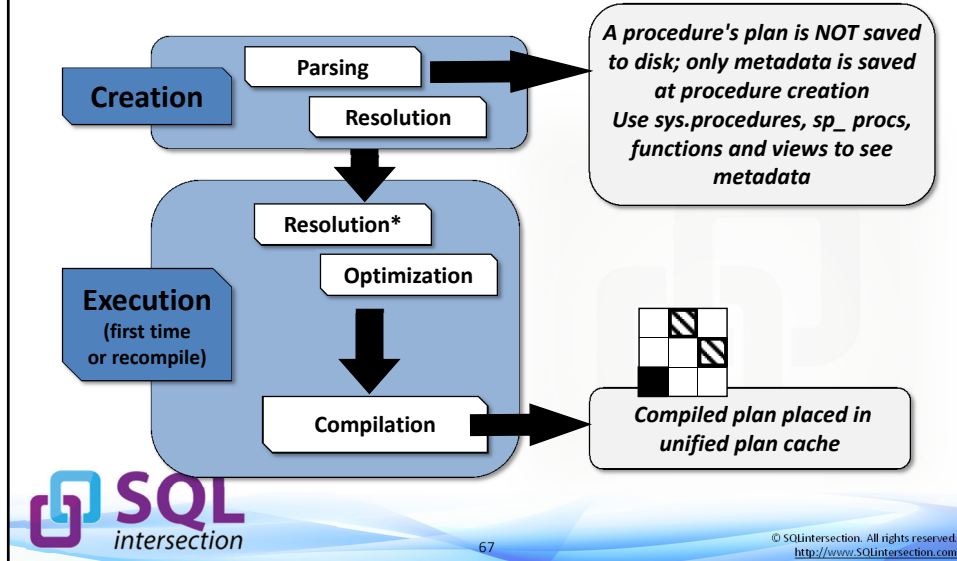
- **Query Optimization**
- **Statistics**
- **Data Distribution**
- **Statement Execution and Plan Caching**
- **Stored Procedures**
 - Creation
 - Optimization
 - Plan invalidation
 - Recompilation
 - Optimization strategies

66

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



Processing Stored Procedures



Stored Procedure Optimization

- Plan is generated when no plan already exists in cache
- Plans are never saved on disk and can "fall out" of cache
 - Forced out through:
 - Server restart
 - DBCC FREEPROCCACHE
 - DBCC FLUSHPROCINDB
 - sp_recompile
 - Aged out through non-use
 - Schema of base object changes
 - Statistics of base objects change
 - See next slide for details specific to version
- When a plan exists in cache, all subsequent executions use that plan
- Plans are not regenerated for subsequent executions (even when statistics or indexes are added – use sp_recompile)

Plan Invalidation A Painful History

- In SQL Server 2012, statistics updates only cause plan invalidation IF the data has changed (what defines data change? At least ONE row has changed.)
- In prior versions (SQL Server 2008R2, 2008, 2005)
 - If database option: `auto_update_stats` is ON
 - Updating statistics runs and always caused plan invalidation
 - If database option: `auto_update_stats` is OFF
 - Updating statistics always ran but does NOT cause plan invalidation
- Recommendation
 - In earlier versions, use `sp_recompile` after updating stats against a table
 - In SQL 2012, use `sp_recompile` after adding indexes or adding/updating statistics
- For more information, check out these blog posts:
 - *What caused that plan to go horribly wrong – should you update statistics?*
<http://www.sqlskills.com/blogs/kimberly/what-caused-that-plan-to-go-horribly-wrong-should-you-update-statistics/>
 - *Statistics and Recompilations:* <http://erinstellato.com/2012/01/statistics-recompilations/> and Part II: <http://erinstellato.com/2012/02/statistics-recompilations-part-ii/>



69

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

Procedure Caching Isn't That the Point?

- Reusing plans can be good
 - When different parameters don't change the optimal plan, then saving and reusing is excellent!
 - SQL Server saves time in compilation
- Reusing plans can be VERY bad
 - When different parameters wildly change the size of the result set and the optimal plans vary, then reusing the plan can be horribly bad
 - Don't worry, I'll show you how to see this...
 - Don't worry, I'll show you the plethora of options to control/fix this!
 - If indexes are added to base tables, existing plans may not leverage them
 - Don't worry, there's a simple solution here:
 - `EXEC sp_recompile <tablename>`



70

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

Recompilation Issues

RECOMPILATION = OPTIMIZATION

OPTIMIZATION = RECOMPILATION

- When do you want to recompile?
- What options do you have for recompilation – and at what granularity?
- How do you know you need to recompile?
- Do you want to recompile the entire procedure or only part of it?
- Can you test it?



71

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

When to Recompile?

- When the plan for a given statement within a procedure is not consistent in execution plan, due to parameter changes
- Cost of recompilation might be significantly less than the execution cost of a bad plan!
- Why?
 - MUCH faster execution with a better plan
 - Some plans just don't work for a wide variety of your execution cases, in fact, some plans should NEVER be saved
- Do you want to do this for every procedure?
 - No, but start with the highest priority/expensive procedures that aren't performing well first – and test!!



72

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

Options for Recompilation

- **CREATE ... WITH RECOMPILE**
- **EXECUTE ... WITH RECOMPILE**
- **sp_recompile objname**
- **Statement-level recompilation**
 - The 2000+ way (still has benefits)
 - Dynamic string execution (statement isn't stored with proc plan)
 - Modularized code (breaks the statement/block into its own proc)
 - The new way
 - 2005+: **OPTION(RECOMPILE)**
 - 2005+: **OPTION (OPTIMIZE FOR (@variable_name = constant, ...))**
 - 2008+: **OPTION (OPTIMIZE FOR UNKNOWN)**



73

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

CREATE ... WITH RECOMPILE

- **Causes the entire procedure's plan to be recompiled on every execution**
- **Since SQL Server 2005 introduced statement-level recompilation, you should only use this for SMALL procedures**
- **When procedure returns widely-varying results**
- **When the plan is not consistent**
- **For backward compatibility**
 - 2000: has only complete procedure recompiles (KB 263889 Compile Locks)
 - 2005: statement-level recompilation
- **Always target the smallest amount possible to recompile!**
- **NOTE: If a procedure is created WITH RECOMPILE, the statement(s) won't show up in dm_exec_query_stats OR dm_exec_procedure_stats (very limited troubleshooting capabilities; avoid using)**



74

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

EXECUTE WITH RECOMPILE

- **Recompiles the plan only for the execution**
 - Does not affect the plan in cache
 - Does not place this plan in cache for subsequent executions
- **Excellent for testing**
- **Possibly for “atypical” execution scenarios**
- **Verify plans for a variety of test cases**

```
EXEC dbo.GetMemberInfo  
  'Tripp' WITH RECOMPILE  
or   'T%' WITH RECOMPILE  
or   '%T%' WITH RECOMPILE
```
- **Do the execution plans match?**
 - Yes: create the procedure normally
 - No: determine what should be recompiled



75

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

Statement-Level Recompilation

- **In 2005 and 2008+: “inline” recompilation for statements**
 - **OPTION (RECOMPILE)**
 - Excellent when parameters cause the execution plan to widely vary
 - Bad because EVERY execution will recompile – but only for the statements
 - **OPTION (OPTIMIZE FOR (@variable = literal, ...))**
 - Excellent when large majority of executions generate the same optimization time
 - You don't care that the minority may run slower with a less than optimal plan?
 - 2008+ only: **OPTION (OPTIMIZE FOR UNKNOWN)**
 - Use the “all density” (average) instead of the histogram
- **The old way: modularizing your code**
 - Doesn't hurt!
 - Better for statement blocks/code reuse
 - SQL Server never steps into a procedure until it's executed



76

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

Modularization

- **Be careful of block statements/conditional logic**
 - SQL Server optimizes the process of optimization and this may lead to a less-than-optimal plan (no, really!)
- **Breaking the stored procedure into smaller chunks**
 - Can handle the block of statements on a more granular basis
 - Subprocedures can be optimized/compiled
 - `CREATE subprocedure WITH RECOMPILE`
 - `EXECUTE subprocedure WITH RECOMPILE`



77

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

Multi-Purpose Procedures

- **Stored procedures with n parameters where any combination of these parameters can be supplied**
- **Often the code looks like this:**

```
WHERE ...  
  (column1 = @variable1 OR @variable1 IS NULL)  
AND (column2 = @variable2 OR @variable2 IS NULL)  
... AND (columnN = @variablen OR @variablen IS NULL)
```
- **These are very difficult for the optimizer to “generalize” and what often happens is a generally bad plan**
- **How do you fix it?**
 - Concatenate ONLY when the variable is NOT NULL
 - Define the data type of the variable in the string. Use: `column = convert(datatype, @variable)` in actual string (to reduce implicit conversions and plan cache bloat)
 - Be careful using `sp_executesql`



78

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

Ad Hoc Statements and Statement Auto-Parameterization

- **EXEC (@str) = ad hoc statements, these can:**
 - Be cached when there are no parameters: subsequent executions must be an exact textual match (case-sensitive regardless of db setting and spaces)
 - Be parameterized and deemed safe
 - Subsequent executions will use the plan in cache
 - Be parameterized and deemed unsafe (MOST LIKELY)
- **sp_executesql = forced statement caching**
 - The first execution will be “sniffed” and the plan will be placed in cache for subsequent executions, regardless of whether or not the plan is good/consistent for all values
 - Great when the plan is consistent!
 - Can be horrible when it’s not!!



79

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

Summary: Stored Procedure Pitfalls/Performance

- **Stored procedures and sp_executesql have the same potential for executing a bad plan but stored procedures have more options for centralized control**
- **Forcing a recompile can be warranted/justified**
- **Always recompile the smallest amount possible!**
- **But, can you have too many recompiles?**
 - Yes: however, it’s not as bad as 2000 because only the statement is recompiled (as of 2005+), not the entire procedure
 - Yes: however, sticking to these best practices can help to DRASTICALLY minimize that!



80

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

Workshop Review

- **Query Optimization**
 - Understanding cardinality and selectivity; how SQL Server "predicts" data sets
 - When statistics aren't available, what happens (heuristics)?
- **Statistics**
 - System-created vs. user-defined vs. hypothetical indexes
 - When are they created vs. when are they updated (full-scan vs. sampling)
- **Data Distribution**
 - Evenly distributed data (yeah, right)
 - Problems with uneven distribution
- **Plan Caching**
 - Statement Execution
 - Simple Parameterization vs. Forced Parameterization
- **Stored Procedures**
 - Creation/Optimization
 - Recompilation

81

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

Resources

- **Whitepaper: Plan Caching and Recompilation in SQL Server 2012**
 - <http://msdn.microsoft.com/en-us/library/dn148262.aspx>
- **Demo code/samples:**
 - SQLskills, Resources, Demo Scripts and Sample Databases
- **Automated maintenance scripts to help you create more efficient and more optimal maintenance:** <http://ola.hallengren.com/>
- **Please check out the code, let me know what you think – send me feedback. My code for this workshop is essentially v2**
 - After I get even more feedback and I'll make a few more tweaks...
 - I'll end up blogging about it and then I'll update the code on my blog
 - You can always reach me at: Kimberly@SQLskills.com



82

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



Questions?

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

Session: PRECON05

Queries Gone Wild – Real-world Solutions

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



Thank you!