SQL intersection

Queries Gone Wild 2

Statistics and Cardinality Estimation

Versions 2008, 2008R2, 2012, and 2014

Post-conference Workshop #06

SQLintersection Fall 2014

Workshop: Friday, November 14, 2014

written and presented by

Joe Sack, http://www.SackHQ.com

Kimberly L. Tripp, http://www.SQLskills.com

SQLintersection
POSTCON06

Queries Gone Wild 2: Statistics / Cardinality in Versions 2008, 2008R2, 2012, and 2014

Joe Sack
Joe@SackHQ.com

Kimberly L. Tripp
Kimberly@SQLskills.com

Authors / Instructors

- **Part 1 – Statistics: Kimberly L. Tripp, http://www.SQLskills.com**
- **Part 2 – Cardinality Estimation: Joe Sack, http://www.sackhq.com**
- **Why do we need multiple instructors?**
  - Different perspectives
  - Different areas of expertise
  - Different problem solving skillsets
  - No one knows everything…
- **This workshop is about knowing:**
  - Internals and really understanding what's going on "under the hood"
  - Where to look
  - What tools are available
  - All sorts of tips/tricks/best practices along the way!

SQL intersection

2

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 SQL Server Immersion Events**
- **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, SQLPASS, ITForum, TechDays, SQLintersection**
- **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**
- **Author/presenter for numerous online training courses at Pluralsight**
- **Owner and technical Manager of the SQLintersection conference**

SQL intersection

3

Workshop Agenda – Part 1

- **Statistics and data distribution**
  - Statistics
    - What do they look like?
    - What are they telling us?
    - How do you see them?
    - When / how do they get created?
    - When / how do they get updated?
  - Data distribution
    - Understanding distribution
    - Problems with uneven distribution (analyzing data skew)
    - Filtered statistics (FS)
    - Automating the creations of FS

SQL intersection

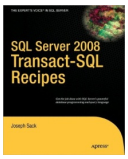
4

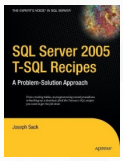
Author/Instructor: Joe Sack

Microsoft CERTIFIED Master | SQL Server® 2005 SQL Server® 2008

MVP Microsoft Most Valuable Professional

- **SQL Server professional, writer, speaker - 17 years**
- **Past roles: SQLskills Principal Consultant, Microsoft Premier Field Engineer, and Program Manager for SQL Microsoft Certified Master**
- **Recent publication: Microsoft White Paper, "Optimizing Your Query Plans with the SQL Server 2014 Cardinality Estimator"**





pluralsight

SQL intersection

5

Workshop Agenda – Part 2

- **Cardinality Estimation**
  - Why CE matters
  - Predicates and selectivity
  - Problem identification
  - Troubleshooting bad estimates
  - The new cardinality estimator

SQL intersection

6

SQLintersection

Queries Gone Wild 2

Part 1 – Statistics and Data Distribution

Kimberly L. Tripp
Kimberly@SQLskills.com

SQL intersection

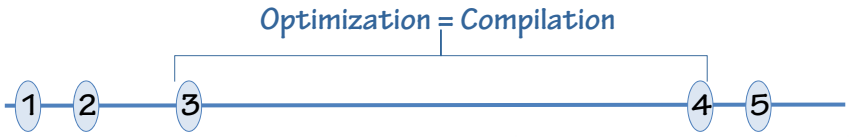
Part 1 – Overview

- **Statistics and data distribution**
  - Statistics
    - What do they look like?
    - What are they telling us?
    - How do you see them?
    - When / how do they get created?
    - When / how do they get updated?
  - Data distribution
    - Understanding distribution
    - Problems with uneven distribution (analyzing data skew)
    - Filtered statistics (FS)
    - Automating the creations of FS

SQL intersection

8

Statement Execution Simplified



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 (not T-SQL anymore)
3. **Cost-based optimization (statistics are used to come up with optimization plan as well as lock granularity)**
4. **Compilation**
5. **Execution: at runtime, if the resources don't exist to support the lock level then escalation occurs to TABLE level (by default)**

SQL intersection

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)**

SQL intersection

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 the tipping point – it's a very low percentage (usually 1-2%) where SQL Server deems a result set to be not selective enough
  - Must consider some form of covering

SQL intersection

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?

SQL intersection

How Would SQL Server Know?

- In every example, knowing the number of rows is necessary to estimate I/Os
- How does SQL Server know how many rows without reading the rows?

Statistics

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

3

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

Overall, this is weird data?!

Histogram

4

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

What Are They Telling You?

- **Histogram has the best detail about the first column (sometimes referred to as the high-order element [LastName])**
  - Anderson 385 rows
  - Barr 385 rows
- **Secondary columns – DENSITY (#3)**
  - Always LEFT-based combinations
  - Rows * All density = average number of rows given that column or combination of columns
    - Index LastName, FirstName will have average for LastName alone and combo
    - Index FirstName, LastName will have average for FirstName alone and combo
    - 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 returned on average]
  - Density for LastName, FirstName combo – what does that tell you?

SQL intersection

15

hidden slide w/extra details

Are They Really True?

- **For LastName**

```
SELECT AVG(Counts.GroupCounts)
FROM (SELECT COUNT(*) AS GroupCounts
      FROM dbo.member AS m
      GROUP BY m.LastName) AS Counts
```

- **For LastName, FirstName**

```
SELECT AVG(Counts.GroupCounts)
FROM
  (SELECT COUNT(*) AS GroupCounts
  FROM dbo.member AS m
  GROUP BY m.LastName, m.FirstName)
    AS Counts
```

- **What does this tell you about FirstNames?**

SQL intersection

16

What Are the Options?

```
SELECT m.LastName, m.FirstName,
      m.MiddleInitial, m.Phone_no, m.City
   FROM dbo.Member AS m
   WHERE m.FirstName LIKE 'Kim%'
```

- **Table scan (always an option)**
- **No indexes exist for SEEKING (no indexes with FirstName as the high-order element)**
- **What about SCANning the NC index which has FirstNames in it…seems like a gamble?**

SQL intersection

17

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 17:18:12.59
[member_corporation_link]	ON	2008-08-26 17:18:12.67
[member_region_link]	ON	2008-08-26 17:18:12.79
[MemberName]	ON	2008-10-29 11:13:29.22
[_WA_Sys_00000003_0CBAE87]	ON	2008-10-29 11:28:32.31

SQL intersection

18

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+)**

SQL intersection

What Kinds of Statistics Exist?

- **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"

SQL intersection

DBCC AUTOPILOT

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

```
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)
GO
SET AUTOPILOT OFF
```

SQL intersection

21

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 = TABLE_CARDINALITY/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

SQL intersection

22

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

SQL intersection

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)**

SQL intersection

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'
```

SQL intersection

25

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

SQL intersection

26

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

SQL intersection

27

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)

SQL intersection

28

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
```

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.

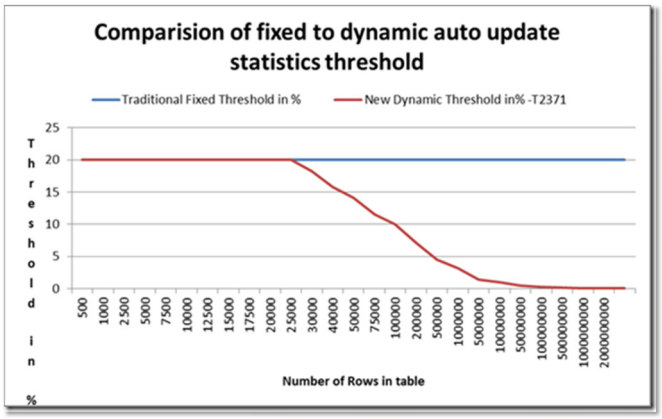
FIXED: Cumulative Update 2 for SQL Server 2012 SP1, Cumulative Update 5 for SQL Server 2012, Cumulative Update 4 for SQL Server 2008 R2 SP2

See KB article: 2778088 http://support.microsoft.com/kb/2778088

SQL intersection

Dynamic Auto-Updating Threshold

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



SQL intersection

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

SQL intersection

Side discussion/FYI

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 do not use auto_update_statistics, add sp_recompile to your stats scripts
  - 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

SQL intersection

Sampling: Always Good?

- **Using *actual* 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 with FULLSCAN, then you might want to turn off auto updating for this index (using STATISTICS_NORECOMPUTE at the index-level or NORECOMPUTE at the statistics-level) and schedule an UPDATE STATISTICS WITH FULLSCAN instead**

```
UPDATE STATISTICS …
WITH FULLSCAN
```

SQL intersection

33

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

SQL intersection

34

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 (tens of thousands) and millions of rows with heavy skew between steps?**
    - Simply put, the averages just aren't going to cut it anymore…

SQL intersection

35

Demo

Understanding the histogram

What information is stored in the histogram?

How does SQL Server use it?

SQL intersection

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 ⇒ high: 30,000+ (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 ⇒ filtered statistics
- **Not an index, just a statistics blob… (smaller, easier to maintain, more accurate over that set)**

SQL intersection

37

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'
```

SQL intersection

38

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       nvarchar(520) = NULL OUTPUT
    -- Fully delimited name of the worktable in tempdb
```

- **Lots of options and relatively low impact (<u>requires</u> an index that has the column specified [@columnname] as the leading column)**

SQL intersection

39

Scripts: 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:**

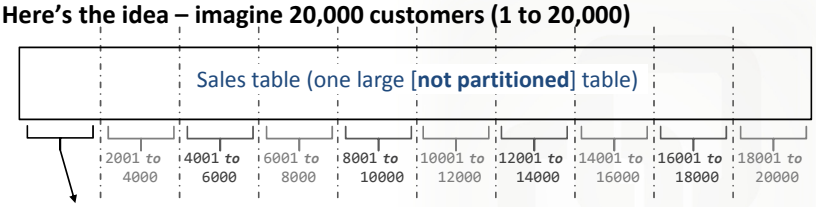
```
EXEC [sp_SQLskills_HistogramTempTables] @management = 'DROP'
```

SQL intersection

40

Filtered Statistics For The Entire Table? (1)

- **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 20,000 customers (1 to 20,000)**




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


Filtered Statistics For The Entire Table? (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 20,000)
  - 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]`

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!**

SQL intersection

43

Scripts: 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'
```

SQL intersection

44

hidden slide w/extra details

How Many Filtered Indexes/Stats?
Per Table

- **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**
  - Columns: 30,000
  - Nonclustered indexes: 999
  - Statistics: 10,000 (30,000 in R2)

SQL intersection

45

hidden slide w/extra details

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...

SQL intersection

46

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)

SQL intersection

47

hidden slide w/extra details

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.
```

SQL intersection

48

Client Consistency Requirements

hidden slide w/extra details

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

SQL intersection

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.**

SQL intersection

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)

51

Demo

Interval subsumption

When won't SQL Server use a filtered object?

Is there a workaround?

The Bad News…

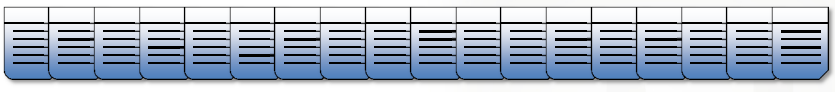
- **Filtered indexes/filtered stats: stats may get HORRIBLY out of date:**
  - SQL Server ONLY tracks a colmodctr NOT related to the filtered set or size. As a result, the stats (even filtered stats) are only updated when 20% of the table has been modified…
  - For better performance, you need to automate 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)
- **Stored procedures and sp_executesql will not use filtered objects unless you recompile the statement [add OPTION (RECOMPILE)]**
- **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!***

SQL intersection

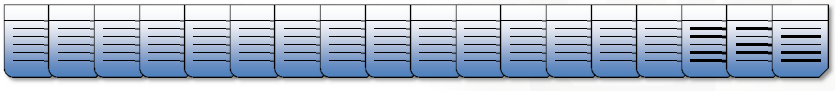
53

Evenly Distributed Data?

- **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?**



SQL intersection

54

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
```

SQL intersection

55

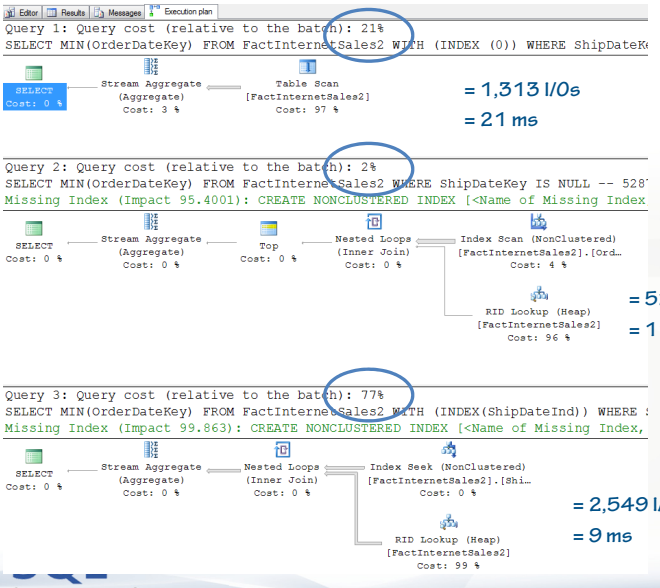
Even Distribution: Always Even??

- **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!**

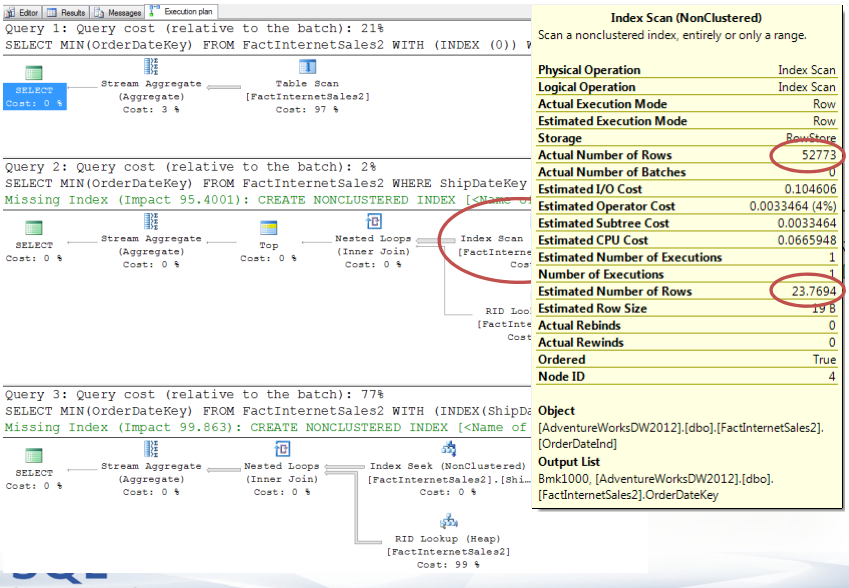
SQL intersection

56

Evenly Distributed Data??



Evenly Distributed Data??



Always Better: Indexes

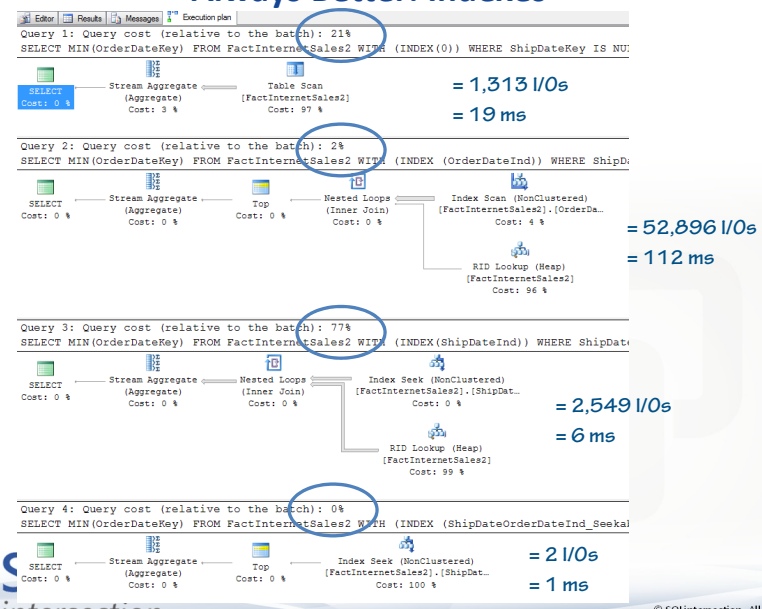
- **Statistics cannot be used to directly access data but:**
  - They can only HELP the optimizer determine the best plan
  - They can help determine best join order
- **Statistics are just estimates: they help MOST of the time but:**
  - There are limitations
  - They take time to create, update, store…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

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

SQL intersection

59

Always Better: Indexes



SQL intersection

60

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!**

SQL intersection

Part 1 – Review

- **Statistics and data distribution**
  - Statistics
    - What do they look like?
    - What are they telling us?
    - How do you see them?
    - When / how do they get created?
    - When / how do they get updated?
  - Data distribution
    - Understanding distribution
    - Problems with uneven distribution (analyzing data skew)
    - Filtered statistics (FS)
    - Automating the creations of FS

SQL intersection

SQLintersection

Queries Gone Wild 2

Part 2 – Cardinality Estimation

Joe Sack
Joe@SackHQ.com

SQL intersection

Part 2 – Overview

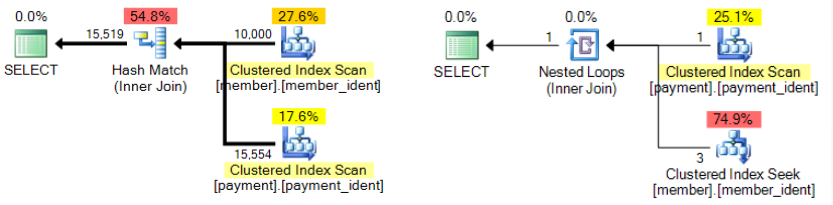
- **Cardinality Estimation**
  - Why CE matters
  - Predicates and selectivity
  - Problem identification
  - Troubleshooting bad estimates
  - The new cardinality estimator

SQL intersection

64

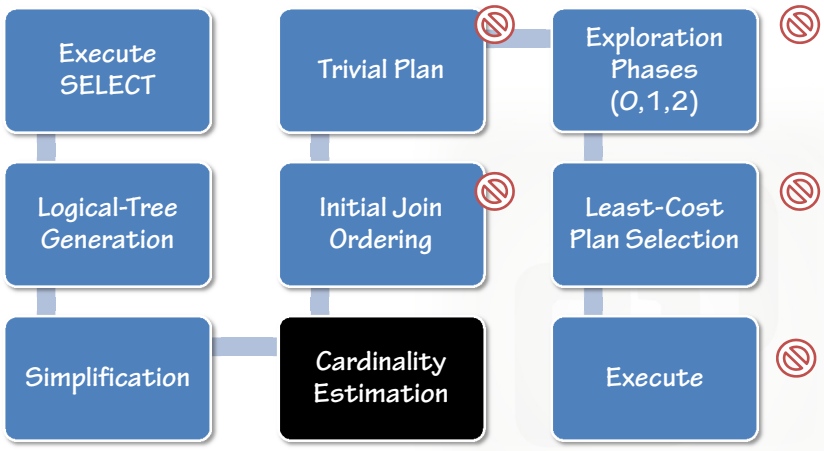
Which is the "Good" Plan?

```
SELECT
    [member].[member_no],
    [member].[lastname],
    [payment].[payment_no],
    [payment].[payment_dt],
    [payment].[payment_amt]
FROM [dbo].[member]
INNER JOIN [dbo].[payment] ON
    [member].[member_no] = [payment].[member_no];
```



SQL intersection

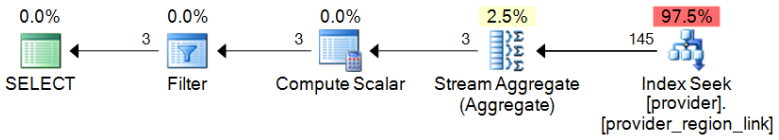
Downstream Impact of Bad Assumptions



SQL intersection

Cardinality Estimates

- **Estimate of rows processed per operator in a query plan**
  - Row estimates calculated from leaf-level to root
  - Non-leaf level operators look at descendent operator estimates and apply additional estimation activities, for example, filtering
- **Cardinality estimation relies on statistics, constraints, and heuristics**

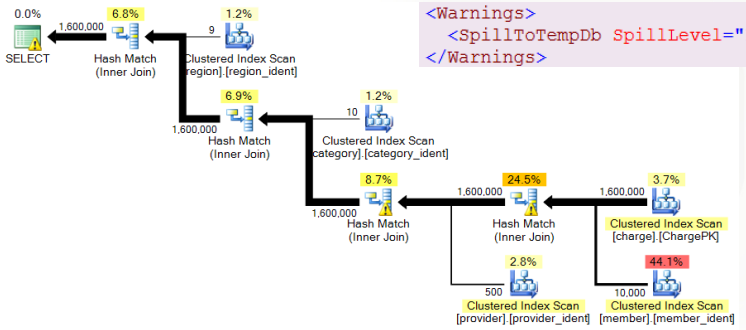


SQL intersection

67

Under-estimates and Spills

- **Significant under-estimates?**
  - Hash and Sort operations example
  - Memory grants may be insufficient
  - Risk of spills to disk, excessive I/O




```
<Warnings>
  <SpillToTempDb SpillLevel="1" />
</Warnings>
```

SQL intersection

68

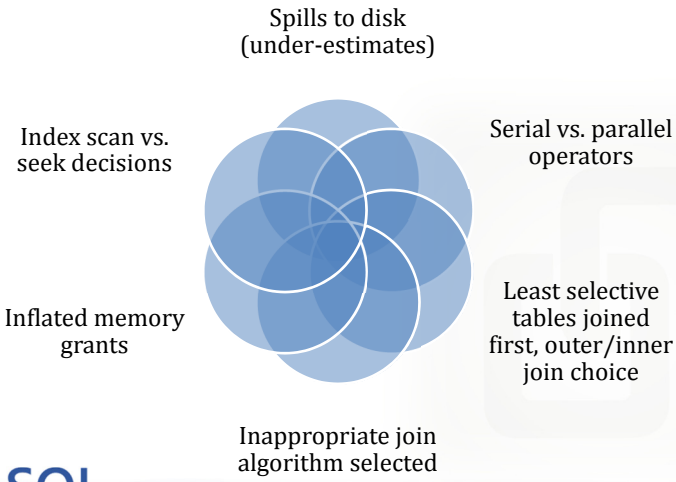
Over-estimates and Concurrency

- **Significant over-estimates?**
  - Hash and Sort operation impact example
  - Memory grants may be unnecessarily inflated
  - Concurrent requests wait for inflated memory grants
  - Concurrency throttled unnecessarily

	requested_memory_kb	granted_memory_kb	required_memory_kb
1	94512	NULL	36496
2	94608	94608	36496
3	95008	NULL	36496

sys.dm_exec_query_memory_grants

Impact of skewed estimates



Demo

Predicates and Selectivity

Cardinality Estimation and Predicates

- **Cardinality estimates are row count estimates calculated for each operator within a query execution plan**
- **The cardinality estimation process is interested in the predicates you have defined in a T-SQL query**
  - Predicates are expressions that evaluate to TRUE, FALSE or UNKNOWN
    - Filter predicates are used in a search condition via a WHERE and/or HAVING clause of a T-SQL query
    - Join predicates are used in JOIN conditions of the FROM clause
- **The cardinality estimation process is interested in determining the selectivity of a predicate**

72

Predicates

```
SELECT
    [AddressID],
    [AddressLine1],
    [AddressLine2]
FROM [Person].[Address]
WHERE [StateProvinceID] = 9 AND
      [City] = N'Burbank' AND
      [PostalCode] = N'91502';
```

Three predicates.

Predicates: Expressions that evaluate to TRUE, FALSE, UNKNOWN.

SQL intersection

73

Filter Predicates

```
SELECT  [od].[SalesOrderID], [od].[SalesOrderDetailID]
FROM    [Sales].[SalesOrderDetail] AS [od]
        INNER JOIN [Production].[Product] AS [p]
            ON [od].[ProductID] = [p].[ProductID]
WHERE   [p].[Color] = 'Red'
        AND [od].[ModifiedDate] = '2008-06-29 00:00:00.000';
```

Filter predicates: used in a search condition via WHERE and HAVING

SQL intersection

74

Join Predicates

```
SELECT  [od].[SalesOrderID], [od].[SalesOrderDetailID]
FROM    [Sales].[SalesOrderDetail] AS [od]
        INNER JOIN [Production].[Product] AS [p]
            ON [od].[ProductID] = [p].[ProductID]
WHERE   [p].[Color] = 'Red'
        AND [od].[ModifiedDate] > '2008-06-29 00:00:00.000';
```

Join predicates: used in JOIN conditions of the FROM clause

SQL intersection

75

Selectivity of a Predicate

$$\frac{\#\ of\ Rows\ Passing\ the\ Predicate}{Total\ Rows}$$

1.0 = low selectivity

0.0 = high selectivity

How many products have a "red" color?

$$\frac{432\ Rows}{1{,}600{,}000\ Rows}$$

0.00027 selectivity

SQL intersection

76

Cardinality Estimator Questions

- **What is the selectivity of:**

- **A query with a single filter predicate?**
- **A query with multiple filter predicates?**
- **A join predicate between two tables?**

More Cardinality Estimator Questions

- **Given a GROUP BY, DISTINCT query:**

- **How many distinct values do we expect from a specific column?**
- **How many distinct values do we expect from a set of columns?**
  - (How many unique combinations)?

Statistics and CE

- **Statistics drive cardinality estimates**
- **If the statistics don't reflect reality, your query execution plan is unlikely to include the most optimal choices**
- **Understanding how to analyze statistics objects (index or standalone) is a critical skill when used for tuning queries in SQL Server**
- **You'll want to be very familiar with DBCC SHOW_STATISTICS and other methods for finding out which statistics were used for a specific query**

SQL intersection

79

Demo

Predicates and Selectivity

SQL intersection

Histogram Direct Hit

```
SELECT [BusinessEntityID]
FROM [HumanResources].[Employee]
WHERE [JobTitle] = N'Sales Representative';
```

Plan Tree

Operation	Object	Est Cost	Est Subtree Cost	Actual Rows	Est Rows
SELECT		100.0%	100.0%	14	14
Clustered Index Scan	[AdventureWorks2012].[HumanResources].[Em...	100.0%	100.0%	14	14

```
DBCC SHOW_STATISTICS ('HumanResources.Employee'
                      _WA_Sys_00000006_49C3F6B7');
```

	RANGE_HI_KEY	RANGE_ROWS	EQ_ROWS	DISTINCT_RANGE_ROWS	AVG_RANGE_ROWS
49	Quality Assurance Technician	0	4	0	1
50	Recruiter	0	2	0	1
51	Research and Development Engineer	0	2	0	1
52	Research and Development Manager	0	2	0	1
53	Sales Representative	0	14	0	1
54	Scheduling Assistant	0	4	0	1

SQL intersection

81

Histogram *Intra*-Step Hit

```
SELECT [BusinessEntityID]
FROM [HumanResources].[Employee]
WHERE [JobTitle] = N'Production Technician - WC39';
```

Plan Tree

Operation	Object	Est Cost	Est Subtree Cost	Actual Rows	Est Rows
SELECT		100.0%	100.0%	0	25
Clustered Index Scan	[AdventureWorks2012].[HumanResources].[Em...	100.0%	100.0%	0	25

	RANGE_HI_KEY	RANGE_ROWS	EQ_ROWS	DISTINCT_RANGE_ROWS	AVG_RANGE_ROWS
41	Production Technician - WC20	0	22	0	1
42	Production Technician - WC40	25	26	1	25
43	Production Technician - WC45		15		
44	Production Technician - WC60		26		
45	Purchasing Assistant	0	2	0	1

SQL intersection

82

Density Vector: Unknown Runtime Value

```
DECLARE @JobTitle nvarchar(50) = N'Sales Representative';

SELECT [BusinessEntityID]
FROM [HumanResources].[Employee]
WHERE [JobTitle] = @JobTitle;
```

Plan Tree

Operation	Object	Est Cost	Est Subtree Cost	Actual Rows	Est Rows
SELECT		100.0%	100.0%	14	4
Clustered Index Scan	[AdventureWorks2012].[HumanResources].[Em...	100.0%	100.0%	14	4

All density	Average Length	Columns
0.01492537	49.82069	JobTitle

0.01492537 * 290 rows = 4.3283573

Avg. Frequency = Density * # of Rows

SQL intersection

Density Vector: Distinct Value Estimation

```
SELECT DISTINCT [JobTitle]
FROM [HumanResources].[Employee];
```

Plan Tree

Operation	Object	Est Cost	Est Subtree Cost	Actual Rows	Est Rows
SELECT		100.0%	100.0%	67	67
Sort (Distinct Sort)		65.2%	100.0%	67	67
Clustered Index Scan	[AdventureWorks2012].[HumanResources].[Em...	34.8%	34.8%	290	290

All density	Average Length	Columns
0.01492537	49.82069	JobTitle

```
-- Reciprocal of DV
SELECT 1/0.01492537;
-- Returns 67.000014070
```

SQL intersection

Demo

Problem Identification

Query Execution Plan

- **Estimated plan**
  - Query not executed
  - Plan represents how the statements would be executed
    - Pulling plan from cache shows estimated not actual!
  - Without actual rows, we don't know if there is a cardinality estimate issue
    - Not always possible with long running or data modifying queries
- **Actual plan**
  - Includes estimated AND actual number of rows by operator and executions by operators
  - This comparison is critical for identifying cardinality estimate issues which may impact query plan quality

86

Capturing an Actual Plan

- **SET STATISTICS XML**
- **SET STATISTICS PROFILE (deprecated, but still available)**
- **Graphical Showplan**
  - "Include Actual Execution Plan"
    - Parsed and executed
  - Estimated vs. Actual rows?
    - F4 properties for each operator
    - Or the ToolTips

SQL intersection

87

SQL Sentry Plan Explorer

- **Free tool**
- **Compact plan representation**
- **Efficient cardinality estimate issue identification**
  - Plan Tree tab

Operation	Object	Est Cost	Est Subtree Cost	Actual Rows	Est Rows
SELECT		100.0%	100.0%	15,554	1
Nested Loops (Inner Join)		0.0%	100.0%	15,554	1
Table Scan	[Credit].[dbo].[payment].(Heap)	25.1%	33.4%	15,554	1
Clustered Index Seek	[Credit].[dbo].[member].[member_iden...	74.9%	99.5%	15,554	3

SQL intersection

88

SQL Server 2012 Supplemental Info

- **New "inaccurate_cardinality_estimate" event (beware overhead)**

```
CREATE EVENT SESSION [CE] ON SERVER
ADD EVENT sqlserver.inaccurate_cardinality_estimate
WITH
     (MAX_MEMORY=4096 KB,
     EVENT_RETENTION_MODE=ALLOW_SINGLE_EVENT_LOSS,
     MAX_DISPATCH_LATENCY=30 SECONDS,
     MAX_EVENT_SIZE=0 KB,
     MEMORY_PARTITION_MODE=NONE,
     TRACK_CAUSALITY=OFF,
     STARTUP_STATE=OFF);
GO
```

- **ConvertIssue plan attribute**
- **Additional row count statistics in sys.dm_exec_query_stats**
- **We'll talk about SQL Server 2014 diagnostic options later…**

SQL intersection

89

Demo

Problem Identification

SQL intersection

Troubleshooting Bad Estimates

What Variance is Problematic?

- **100 rows? 1,000 rows? 1,000,000 rows?**
- **Not all skews matter, so ask these questions:**
  - Does the row estimate skew affect plan choices?
    - Requires a solid understanding of query execution plans
  - Do the plan choices result in excessive resource consumption or excessive execution time?

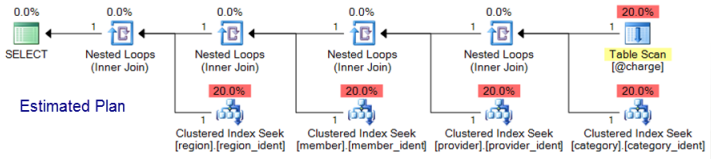
Before Jumping In…

- **Is performance or concurrency degraded?**
  - Not all skews translate to poor performance
    - E.g. a hash join being used due to a skew, but ultimately the hash join operation is more efficient for that particular query
- **False positive on (executions * estimated rows)?**
- **Sometimes the resolution involves significant refactoring of the T-SQL code and associated schema**
  - Keep the cost of effort versus the desired benefit in mind
  - No magic number for defining when a skew is problematic

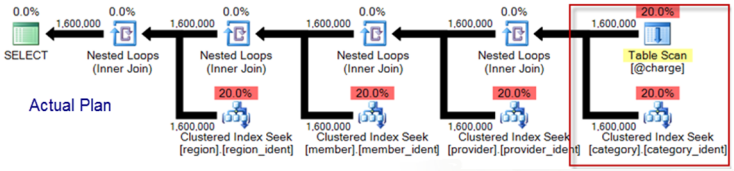
SQL intersection

Issue Prioritization

- **Poor assumptions propagate "up" the tree (right-to-left)**



- **So troubleshoot the root cause, not the side effects**



SQL intersection

Missing or Stale Statistics (1)

- **Missing or out-of-date statistics**
  - Of cardinality estimate issues, this is often the easiest to address
- **Next steps?**
  - Check database options
    - is_auto_create_stats_on
    - is_auto_update_stats_on
- **Check "no recompute" settings**
  - sys.stats catalog view with no_recompute = 1
    - Can be changed via sp_autostats , UPDATE STATISTICS, CREATE STATISTICS, CREATE INDEX, ALTER INDEX

SQL intersection

Missing or Stale Statistics (2)

- **Next steps?**
  - Check STATS_DATE to validate last update
  - Manual statistics updates may be needed where auto-updates are inadequate
    - For example, for very large tables where the automatic update threshold is significantly higher
- **TF 2371 for changes to automatic statistics updates, decreasing dynamic threshold as table rows grow**
  - Trade-off is increased recompilation, new plans
  - See "Changes to automatic update statistics in SQL Server – traceflag 2371" from the Microsoft SAP on SQL Server Blog, http://bit.ly/SAP2371

SQL intersection

Demo

Stale Stats

Statistics Sampling Issue

- **Precision of statistics histogram may be inadequate**
  - Very large table or significant data skews
- **Next steps?**
  - Compare DBCC SHOW_STATISTICS "rows sampled" versus "rows"
  - Evaluate impact based on using higher percent sampling or FULLSCAN during manual statistics update
    - Overhead trade-offs for updating statistics on very large tables
  - Explore filtered statistics for very large tables
    - Beware of update threshold, which is based on overall table thresholds and NOT the filter itself

98

Hidden Column Correlation

- **Are columns correlated?**
  - Query Optimizer assumption is that columns are independent
  - Example – 10,000 row member table

```
SELECT  [lastname] ,
        [firstname]
FROM [dbo].[member]
WHERE [city] = 'Minneapolis'AND
      [state_prov] = 'MN';
```

Actual Rows	Est Rows
1,000	100
1,000	100

No multi-column stats

Actual Rows	Est Rows
1,000	1,000
1,000	1,000

With Multi-Column Stats

- **How to resolve?**
  - Create multi-column statistics
  - If appropriate, create multi-column indexes

99

Demo

Column Correlation

Comparison of Intra-Table Columns

- **Cardinality estimate issues can occur when comparing intra-table columns**

```
SELECT  [statement_no] ,
        [member_no] ,
        [statement_amt] ,
        [statement_code]
FROM    [dbo].[statement]
WHERE   [statement_dt] < [due_dt];
```

Operation	Object	Est Cost	Est Subtree Cost	Actual Rows	Est Rows
SELECT		100.0%	100.0%	20,000	6,000
Clustered Index Scan	[Credit].[dbo].[statement].[statement_ident].(Clustered)	100.0%	100.0%	20,000	6,000

- **How to resolve?**
  - Computed columns and ensuring that associated statistics are generated
  - Self-joins, common table expressions
  - Normalization (e.g. order header vs. detail)

SQL intersection

101

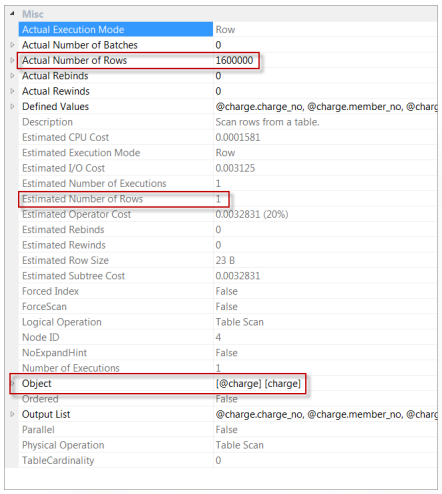
Demo

Intra-Table Column Comparison

SQL intersection

Table Variable Usage

- **Small number of rows?**
  - May be a non-issue
- **Large or volatile result set**
  - Can significantly impact query plan quality
- **Next steps**
  - Compare cardinality using a temporary table or permanent table instead



SQL intersection

103

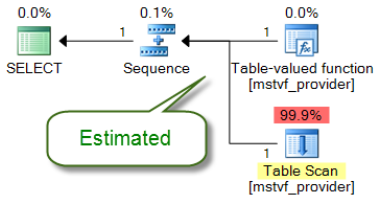
Demo

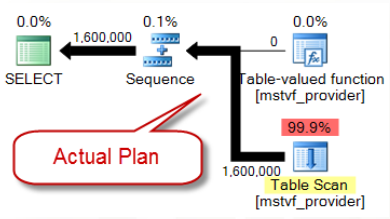
Table Variable

SQL intersection

Scalar and MSTV UDFs

- **Black-box from a cardinality estimate perspective**
  - Multi-statement table-valued function
  - Scalar UDFs





- **Resolutions**
  - Inline versus multi-statement?
  - Is a function even necessary?

Demo

MSTVF

Parameter Sniffing

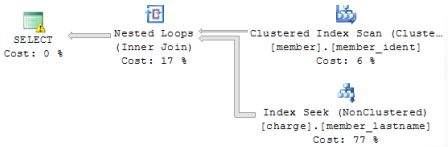
- **Initial compilation of a plan based on the first value passed**
  - Can be a good thing, but with wildly varying plan shapes, can be bad
- **If suspected, validate:**
  - ParameterCompiledValue
  - ParameterRuntimeValue

Parameter List	@member_no
Column	@member_no
Parameter Compiled Value	(1)
Parameter Runtime Value	(2)

- **Several options to explore…**
  - Procedure branching and isolation based on skew parameters
  - OPTIMIZE FOR, local variable method
  - But beware since local variables are non-foldable and may introduce cardinality estimation issues of their own
  - Various recompile options at different scales

Implicit Data-Type Conversion Issues

- **Implicit data-type conversions**
  - Watch for data types of columns in search and join conditions



Warnings
Type conversion in expression (CONVERT_IMPLICIT(nvarchar(15),[Credit].[dbo].[member].[lastname],0)) may affect "CardinalityEstimate" in query plan choice, Type conversion in expression (CONVERT_IMPLICIT (nvarchar(15),[Credit].[dbo].[member].[lastname],0)=N'XAVIER') may affect "SeekPlan" in query plan choice

- **How to resolve?**
  - Use identical data types
  - Use consistent naming conventions so that you can use catalog views to monitor any inconsistencies across column references

Complex Predicates

- **Wrapping column references in functions and complex expressions can hamper accurate cardinality estimation**
  - Not all are problematic, e.g. LOWER, UPPER, GETDATE

```
WHERE    LEFT([member].[firstname], 15) + ' ' +
         LEFT([member].[lastname], 15) + '' = 'VKZEJQXGGPERFL XAVIER'
```

Operation	Object	Est Cost	Est Subtree Cost	Actual Rows	Est Rows
SELECT		100.0%	100.0%	16	184,611
Nested Loops (Inner Join)		18.6%	100.0%	16	184,611
Clustered Index Scan	[Credit].[dbo].[member].[member_iden...	2.8%	2.8%	1	1,000
Index Seek	[Credit].[dbo].[charge].[member_lastna...	78.6%	78.6%	16	184,611

- **How to resolve?**
  - Keep your expression column references “clean”
  - Move complex expressions off of the columns
  - Prepare the value before passing to the predicate

SQL intersection

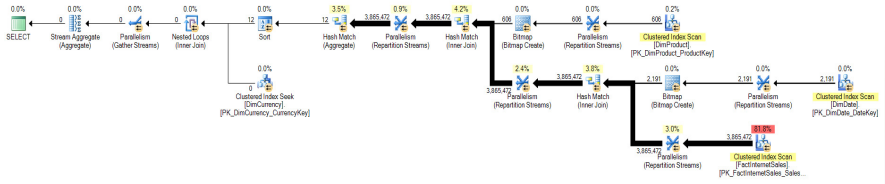
Query Complexity

- **Sometimes the complexity of a query can make it almost impossible to have accurate cardinality estimates**
  - Views referencing views referencing views referencing functions within other views which reference the same tables referenced within views…
- **Resolution?**
  - If such queries are not meeting performance SLAs, one valid technique is to break apart a monolithic query into smaller chunks and intermediate result sets
    - Gives the optimizer fewer factors to consider
    - Multiple optimal plans may outperform one large suboptimal one
      - Trade-off of intermediate result set overhead must be weighed against plan quality benefits

SQL intersection

Gatekeeper Rows

- **Star schema fact and dimension table pattern where specific dimension rows impact whether or not a significant number of fact rows are returned**



  - Denormalization to the fact table and direct predicates on fact table
  - Covering nonclustered index with FORCESEEK (I'm not a fan of doing this, but it works in this case)
  - Dynamic SQL
  - Columnstore indexing
  - See article "Fixing Gatekeeper Row Cardinality Estimate Issues" http://bit.ly/GatekeeperRow

Distributed Queries

- **Permissions associated with the linked-server definition and associated distributed queries can impact whether proper cardinality estimates can be generated**

- **Resolution?**
  - Ensure db_ddladmin fixed database role for the linked server account (minimum required permissions)
  - SQL Server 2012 Service Pack 1 has a fix that alleviates this requirement
    - See blog post "Distributed Query Plan Quality and SQL Server 2012 SP1" http://bit.ly/DQPQSQL
  - Localization via:
    - ETL?
    - Replication?
    - Consolidation?

Query Optimizer Bugs

- **Some query plan quality issues are in fact "bugs"**
- **You may see this particularly in conjunction with newer functionality**
- **Do you think it's a bug? If so:**
  - Check http://connect.microsoft.com
    - For existing entries, add your comments
    - When there is no match, take the time to enter in the reproduction of the bug with as much detail as possible

Troubleshooting Questions (1)

- **Are my statistics missing?**
- **Are my statistics stale?**
- **Is the sampling adequate? (jagged distributions, "lossy" histograms)**
- **Would multi-column statistics help?**
- **Is this a parameter sensitivity issue?**
- **Are table variables and/or MSTVFs causing problems?**

Troubleshooting Questions (2)

- Am I usually only querying the most recent rows?
- Are data type conversions causing issues?
- Am I comparing columns from the same table?
- Am I accessing remote data sources?
- Are my predicates being buried in complexity?
- Am I trying to do too much within a single query? (deep tree)

SQL intersection

Demo

The New Cardinality Estimator

SQL intersection

Cardinality Estimator Changes

- Past changes to the CE have been in the form of trace flags and have been disabled by-default
- SQL Server 2014 marks the first major set of changes to the CE since 7.0
- On average, workloads may benefit, but you can expect to see regressions!

Enabling the new CE

- Database compatibility level 120

```
-- SQL Server 2014 compatibility level
ALTER DATABASE [AdventureWorks2012]
SET COMPATIBILITY_LEVEL = 120;
GO
```

- Database session *context* matters!

Enabling the new CE

- **Fully supported trace flags via QUERYTRACEON or server-level DBCC TRACEON, startup –T:**
- **2312 - use the new CE**
- **9481 - use the legacy CE**

- **QUERYTRACEON requires sysadmin – but plan guides can be one way to separate the duties here**

- **KB link: bit.ly/querytraceon**

SQL intersection

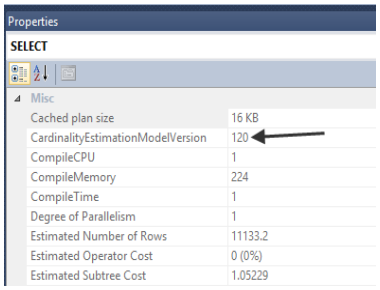
119

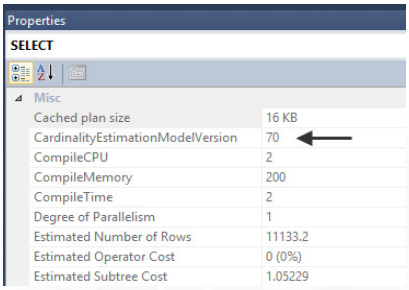
Demo

Enabling the new CE

SQL intersection

Which CE model are we using?






```
<StmtSimple StatementCompId="1" StatementEstRows="45.6074" StatementId="1"
StatementOptmLevel="FULL" StatementOptmEarlyAbortReason="GoodEnoughPlanFound"
CardinalityEstimationModelVersion="70" StatementSubTreeCost="0.180413"
StatementText="SELECT [AddressID],[AddressLine1],[AddressLine2] FROM [Person].[Address]
WHERE [StateProvinceID]=@1 AND [City]=@2" StatementType="SELECT"
QueryHash="0xC8ACCD7DC44F7942" QueryPlanHash="0xD5F23E0FCD5A723"
RetrievedFromCache="false">
```

SQL intersection

121

What changed?

- Correlation for multi-predicate statements
- Modified out-of-range value estimation
- Simplified join estimate algorithm changes
- Modified join-containment assumptions
- Distinct value count estimation changes
- New diagnostic output

SQL intersection

122

CE Model Assumptions

- **Independence**
  - Filters are uncorrelated in absence of statistics indicating otherwise
- **Uniformity**
  - Values in a histogram step are evenly distributed (spread) and have the same frequency
- **Inclusion**
  - When using a column-equal-constant predicate, it is assumed the value actually exists
- **Containment**
  - When estimating an equality join, it is assumed that there is a maximum overlap of distinct values (think "PK-to-FK" relationship")

Correlation for multi-predicate statements (Legacy Behavior)

- **"Independence" assumption at work…**
- **Selectivity of conjunctive predicates are computed as the multiplication of individual selectivities**
- $p_0 \cdot p_1 \quad \cdot p_2 \quad \cdot p_3$

Correlation for multi-predicate statements (New CE Behavior)

- **Predicates are sorted by selectivity, keeping the four most selective predicates for use in the calculation**
- **Successive predicate is then "softened" by taking larger square roots**
- $p_0 \cdot {p_1}^{1/2} \cdot {p_2}^{1/4} \cdot {p_3}^{1/8}$

SQL intersection

Demo

Correlated Predicates

SQL intersection

Correlation for multi-predicate statements (Legacy Behavior)

```
SELECT
    [AddressID],
    [AddressLine1],
    [AddressLine2]
FROM [Person].[Address] AS [a]
WHERE [StateProvinceID] = 9 AND
      [City] = N'Burbank' AND
      [PostalCode] = N'91502';

-- Old CE
-- Estimate falls beneath 1 row, so minimum "1" used
SELECT
    0.009891 * -- PostalCode predicate selectivity
    0.009993 * -- City predicate selectivity
    0.232691 * -- StateProvinceID predicate selectivity
    19614; -- Table cardinality
```

SQL intersection

127

Correlation for multi-predicate conjunctions

```
SELECT
    [AddressID],
    [AddressLine1],
    [AddressLine2]
FROM [Person].[Address] AS [a]
WHERE [StateProvinceID] = 9 AND
      [City] = N'Burbank' AND
      [PostalCode] = N'91502';


-- New CE
-- Estimate increase to 13.4692 rows
SELECT
    0.009891 * -- PostalCode predicate selectivity
    SQRT(0.009993) * -- City predicate selectivity
    SQRT(SQRT(0.232691)) * -- StateProvinceID predicate selectivity
    19614; -- Table cardinality
```

SQL intersection

128

Modified out-of-range value estimation

- (Legacy) "Ascending key problem" - query predicates reference newly inserted data that falls out of the range of a statistic object histogram
- Can result in under-estimates
- Trace flags 2389 and 2390 to enable automatic generation of statistics for ascending keys

- (New CE) Assumption that histogram out-of-range rows DO exist
- Uses an average frequency, calculated by multiplying the table cardinality by the density

SQL intersection

Demo

Ascending Key Estimates

SQL intersection

Modified out-of-range value estimation (Legacy Behavior)

```
DBCC SHOW_STATISTICS('Sales.SalesOrderHeader',
                     _WA_Sys_00000003_4B7734FF);
```

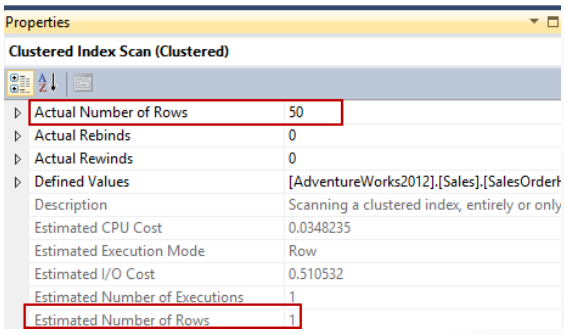
	RANGE_HI_KEY	RANGE_ROWS	EQ_ROWS	DISTINCT_RANGE_ROWS	AVG_RANGE_ROWS
182	2008-06-07 00:00:00.000	87	82	1	87
183	2008-06-09 00:00:00.000	74	68	1	74
184	2008-06-11 00:00:00.000	59	86	1	59
185	2008-06-14 00:00:00.000	123	102	2	61.5
186	2008-06-16 00:00:00.000	81	65	1	81
187	2008-06-18 00:00:00.000	86	72	1	86
188	2008-06-20 00:00:00.000	82	75	1	82
189	2008-06-22 00:00:00.000	91	76	1	91
190	2008-06-24 00:00:00.000	62	74	1	62
191	2008-06-26 00:00:00.000	72	79	1	72
192	2008-06-28 00:00:00.000	73	89	1	73
193	2008-07-01 00:00:00.000	119	37	2	59.5
194	2008-07-08 00:00:00.000	180	51	6	30
195	2008-07-13 00:00:00.000	117	19	4	29.25
196	2008-07-17 00:00:00.000	99	39	3	33
197	2008-07-24 00:00:00.000	183	40	6	30.5
198	2008-07-30 00:00:00.000	148	23	5	29.6
199	2008-07-31 00:00:00.000	0	40	0	1

SQL intersection

131

Modified out-of-range value estimation (Legacy Behavior)

```
SELECT [SalesOrderID], [OrderDate]
FROM [Sales].[SalesOrderHeader]
WHERE [OrderDate] = '2014-02-02 00:00:00.000'
OPTION (QUERYTRACEON 9481);  -- CardinalityEstimationModelVersion 70
```



SQL intersection

132

Modified out-of-range value estimation (New CE Behavior)

All density	Average Length	Columns
0.0008896797		OrderDate

Name	Updated	Rows	Rows Sampled	Steps
_WA_Sys_00000003_4B7734FF	Mar 15 2014 10:53AM	31465	31465	199

Properties

Clustered Index Scan (Clustered)

Misc	
Actual Execution Mode	Row
Actual Number of Batches	0
Actual Number of Rows	50
Actual Rebinds	0
Actual Rewinds	0
Defined Values	[AdventureWorks2012].[Sales].[SalesOrder
Description	Scanning a clustered index, entirely or onl
Estimated CPU Cost	0.0348235
Estimated Execution Mode	Row
Estimated I/O Cost	0.510532
Estimated Number of Executions	1
Estimated Number of Rows	27.9938

SQL intersection

133

Join estimates

- **Simplified join estimation algorithms**
  - Various changes to how joins are estimated – for example –aligning histograms on minimum and maximum boundaries instead of step-by-step alignment
- **Legacy: "Simple containment" assumption**
  - In the presence of a non-join filter predicate against a join table, some correlation is assumed
- **New CE: "Base containment"**
  - Filter predicates on separate tables are NOT assumed to be correlated with each other

SQL intersection

134

Modified join-containment assumptions

```
SELECT  [od].[SalesOrderID], [od].[SalesOrderDetailID]
FROM    [Sales].[SalesOrderDetail] AS [od]
INNER JOIN [Production].[Product] AS [p]
    ON [od].[ProductID] = [p].[ProductID]
WHERE   [p].[Color] = 'Red'
        AND [od].[ModifiedDate] = '2008-06-29 00:00:00.000';
```

Are red products correlated with orders placed on 6/29/2008?

Modified join-containment assumptions (Old CE)

- Histogram of the Product table's [ProductID] column is loaded
- Histogram is scaled down by applying the selectivity of the [p].[Color] = 'Red' filter predicate
- Histogram of the SalesOrderDetail's [ProductID] column is loaded
- Histogram is scaled down by applying the selectivity of the [od].[ModifiedDate] = '2008-06-29 00:00:00.000' filter predicate
- The two histograms are merged together, assuming containment

Modified join-containment assumptions (New CE)

- Histogram of the Product table's [ProductID] column is loaded without scaling down via filter predicates
- Histogram of the SalesOrderDetail's [ProductID] column is loaded without scaling down via filter predicates
- Join selectivity is computed with the two histograms, assuming containment
- CE computes selectivity of the filter predicates on [p].[Color] and [od].[ModifiedDate] respectively
- Join selectivity, [p].[Color] selectivity and [od].[ModifiedDate] selectivities are multiplied in order to calculate the final result

SQL intersection

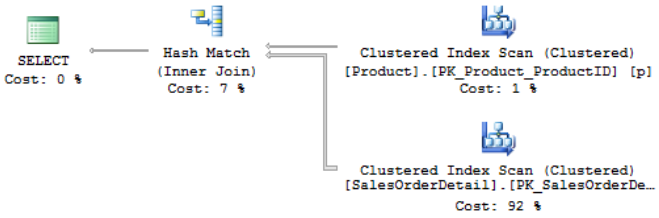
137

Demo

Join Containment

SQL intersection

Modified join-containment assumptions (New CE)




```
SELECT [od].[SalesOrderID], [od].[SalesOrderDetailID]
FROM    [Sales].[SalesOrderDetail] AS [od]
INNER JOIN [Production].[Product] AS [p]
ON [od].[ProductID] = [p].[ProductID]
WHERE   [p].[Color] = 'Red' AND
[od].[ModifiedDate] = '2008-06-29 00:00:00.000';
```

Legacy CE	New CE
51.5437 rows	24.5913

SQL intersection

Distinct value count estimation changes

```
SELECT [f].[ProductKey], [d].[DayNumberOfYear]
FROM    [dbo].[FactInternetSales] AS [f]
INNER JOIN [dbo].[DimDate] AS [d]
    ON [f].[OrderDateKey] = [d].[DateKey]
INNER JOIN [dbo].[FactProductInventory] AS fi
       ON  [fi].[DateKey] = [d].[DateKey]
WHERE [f].[SalesTerritoryKey] = 8
GROUP BY [f].[ProductKey], [d].[DayNumberOfYear];
```

New CE Estimated Rows – Distinct Sort	Legacy CE Estimated Rows – Distinct Sort	Actual Rows – Distinct Sort
5,350 rows	36,969 rows	4,718 rows

New CE uses "ambient cardinality", taking the smallest join that contains the GROUP BY or DISTINCT

SQL intersection

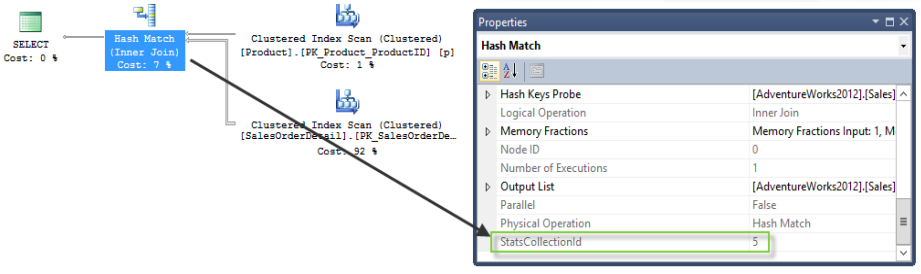
Demo

Distinct Value Count Estimates

SQL intersection

New diagnostic output

- **query_optimizer_estimate_cardinality XEvent**
  - Calculation strategy, statistics used
  - Output for CSS cases
  - When enabled, surfaces StatsCollectionID in QPs
- **Undocumented trace flags? ***



SQL intersection

142

What actions can you take for regressions?

Option:
- **Don't change to compatibility level 120**
- **Use TF 2312 for queries that benefit from the CE**

Option:
- **Change to compatibility level 120**
- **Enable a server-trace flag 9481 to use legacy CE**

Option:
- **Change to compatibility level 120**
- **Use TF 9481 for queries that regressed**

Testing workloads prior to migration

- **Average workload performance may improve or remain the same, but you should expect some regressions**
- **Thoroughly test critical workloads with the new CE before changing in production**
- **If you don't have time to test, don't enable the new CE in production until you have**

References

- **"Troubleshooting Query Plan Quality" (Pluralsight), bit.ly/CEtshoot**
- **Cardinality estimation blog posts on this subject, bit.ly/CEblogposts**
- **Optimizing Your Query Plans with SQL Server 2014 with the New Cardinality Estimator (Microsoft), bit.ly/SackCEPaper**

Workshop Summary

- **Statistics and data distribution**
  - Statistics
  - Data distribution
- **Cardinality Estimation**
  - Why CE matters
  - Predicates and selectivity
  - Problem identification
  - Troubleshooting bad estimates
  - The new cardinality estimator


Questions?
SQLskills
immerse yourself in sql server
Don't forget to complete an online evaluation on EventBoard!
POSTCON06
Queries Gone Wild 2: Statistics and Cardinality
in Versions 2008, 2008R2, 2012, and 2014
Your evaluation helps organizers build better conferences
and helps speakers improve their sessions.
SQL
intersection
Thank you!