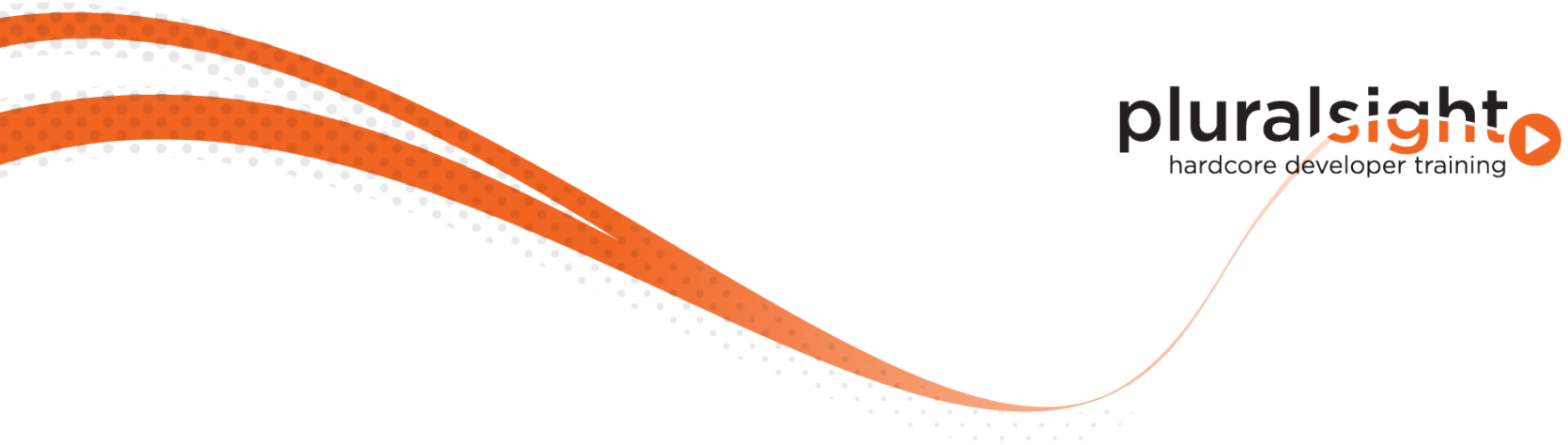
SQL Server: Troubleshooting Query Plan Quality Issues

Module 4: Query Plan Quality Patterns and Resolutions

Joe Sack

Joe@SQLskills.com



Module Introduction

- There are many reasons why query plan quality may be poor
- This module will review the more common patterns
- For each pattern we'll also cover the recommended next steps and associated resolutions

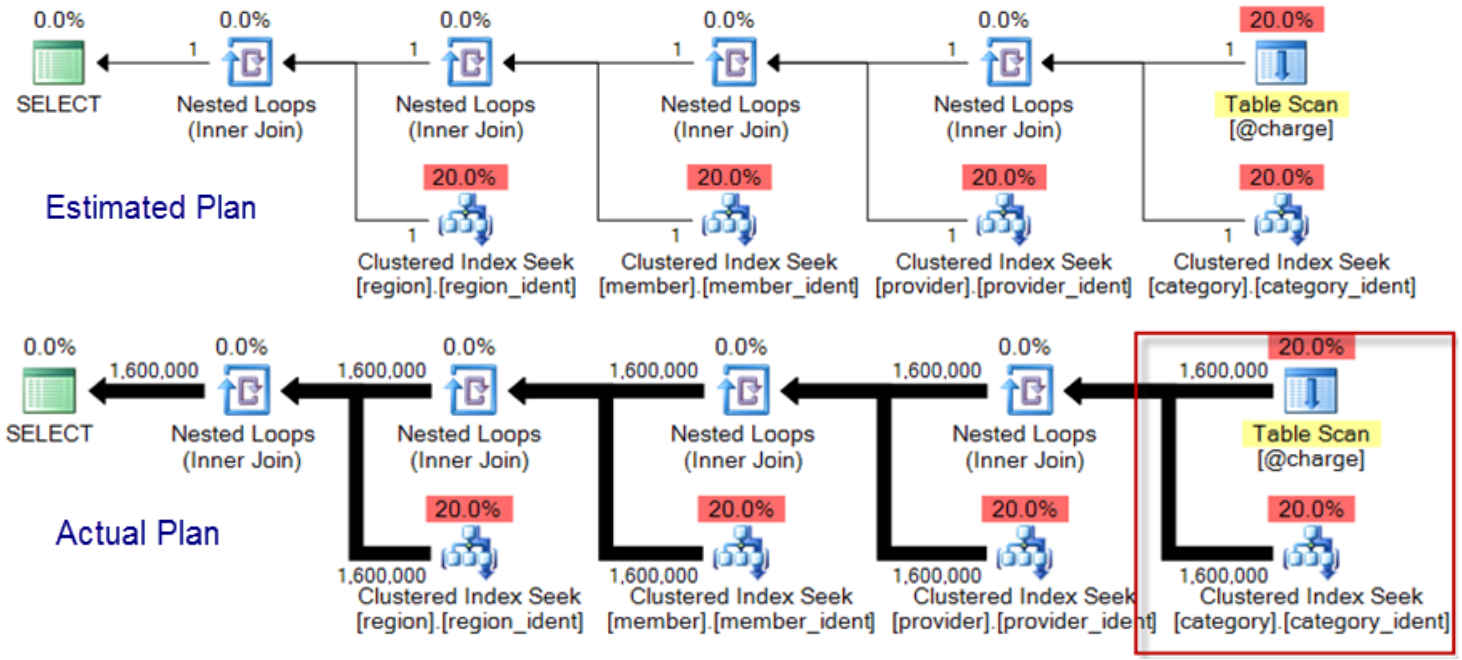
Before Jumping In…

- **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
    - No magic number for defining when a skew is problematic

- **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

Issue Prioritization

- Poor assumptions propagate “up” the tree (right-to-left)



- Troubleshoot the root cause, not the side effects

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- **Missing or out-of-date statistics**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

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, less frequent

- **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/qOAlqs

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”
    - Can also use the new sys.dm_db_stats_properties
      - See Erin Stellato’s blog post for more details -> http://bit.ly/NyCzWf
  - 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

Hidden Column Correlation

- **Are columns correlated?**
  - Query Optimizer assumption is that columns are independent
- **How to resolve?**
  - Create multi-column statistics
  - If appropriate, create multi-column indexes

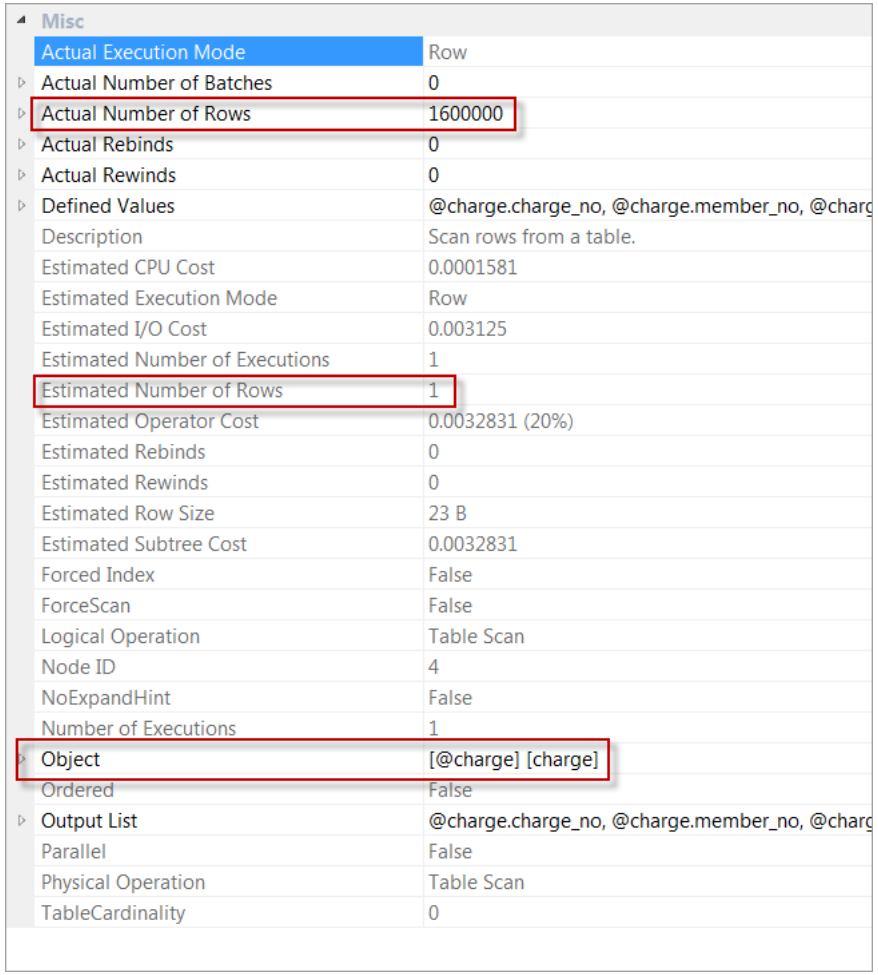
Comparison of Intra-Table Columns

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

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

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



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?

Parameter Sniffing

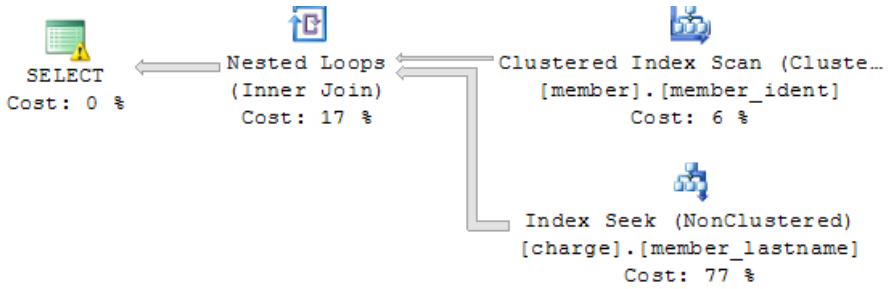
- **Initial compilation of a plan based on the first value passed**
  - Can be a good thing, but with wildly varying plan shapes, can be bad
- **If suspected, validate:**
  - ParameterCompiledValue
  - ParameterRuntimeValue

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 may introduce cardinality estimation issues of their own
  - Various recompile options at different scopes

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

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

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

Hints

- **Hint usage can cause cardinality estimate issues**
  - Various hints, including join hints, FAST number_rows
- **How to resolve?**
  - Evaluate whether hints are still needed and remove accordingly

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/Tapozh
  - Localization via:
    - ETL?
    - Replication?
    - Consolidation?

Query Optimizer Bugs

- **Some query plan quality issues may be "bugs" or regressions**

- **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

Course Summary

- Cardinality estimates are critical for proper operator costing and overall query execution plan quality
- Poorly performing query? Pay attention to skews
- Problem manifests in a variety of way, so your solutions will vary
- Resolve operator leaf-level skews first before addressing intermediate-level operator cardinality estimation skews
- Improving cardinality estimates, improves query plan quality
- Thanks for watching!