SQL Server: Troubleshooting Query Plan Quality Issues

Module 2: Why Query Plan Quality Matters

Joe Sack

Joe@SQLskills.com

pluralsight
hardcore developer training

Module Introduction

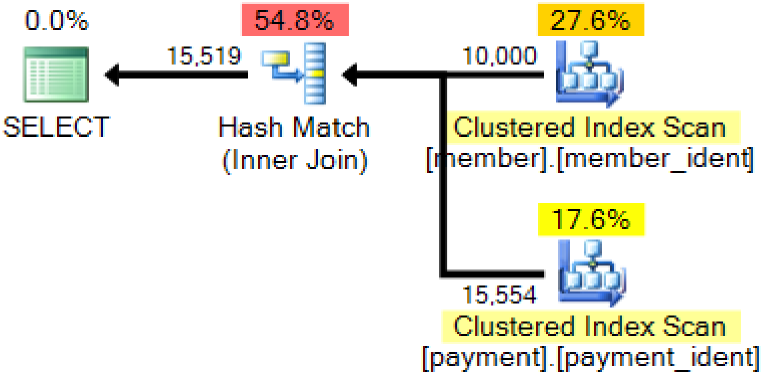
- When a query execution plan is generated based on skewed data, the consequences can be significant

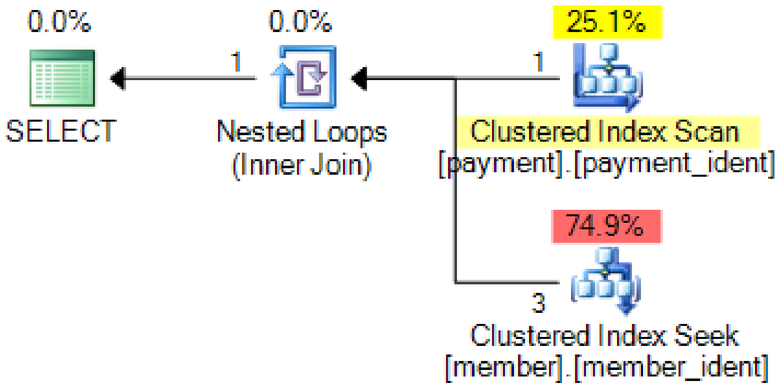
- It’s common to troubleshoot the side-effects of query plan quality issues, but troubleshooting the root cause can be far more effective

- This module will cover the fundamentals of why you should care about query plan quality

Which is the "Good" Plan?

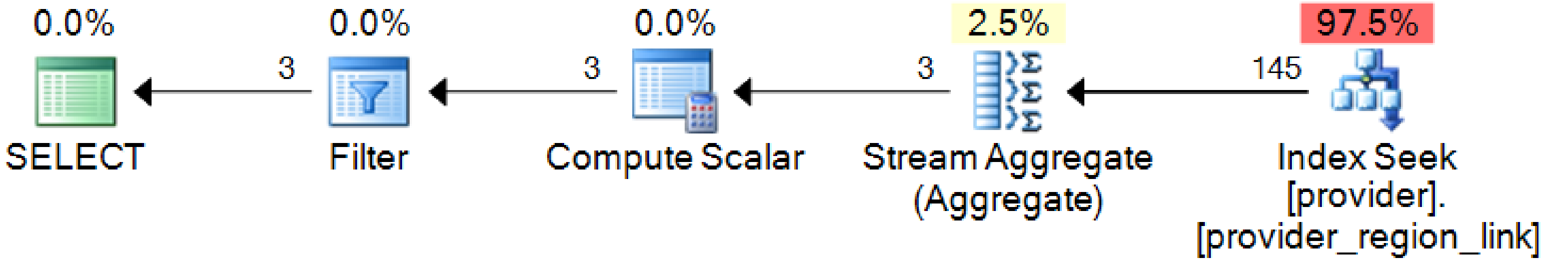
```sql
SELECT
    [member].[member_no],
    [member].[lastname],
    [payment].[payment_no],
    [payment].[payment_dt],
    [payment].[payment_amt]
FROM [dbo].[member]
INNER JOIN [dbo].[payment] ON
    [member].[member_no] = [payment].[member_no];
```





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**

Costing and Plan Quality

- **Row estimates are fed into individual operator cost models**
  - Costing is a major factor in deciding which plan is chosen

- **What if row count assumptions are incorrect?**
  - Bad assumptions can lead to inefficient plan choices
  - A suboptimal plan can result in a slow-running query, excessive resource consumption, and reduced concurrency

Operator Cost (1)

- **Cost originally equated to elapsed time in seconds required to run on a specific Microsoft employee's machine (during SQL 7.0 time-frame)**
- **So really, "cost" today in the context of query plans is a unit-less measure**
    - Cost does not equate to time
- **Cost is used for relative comparison across plan operators and between plans**

Operator Cost (2)

- **Operator cost = I/O cost + CPU cost**
- **Cost calculation varies by operator**
  - Some have I/O and CPU costs
  - Some have just CPU cost

- **Sub-tree cost = cost of specific operator + descendants**
  - Total cost found in root operator

Operator Cost (3)

- **I/O cost assumptions (example):**
  - Data pages NOT in cache
  - Random I/O = 0.003125
  - Sequential I/O = 0.000740741
- **These aren’t formally documented**

Operator Memory

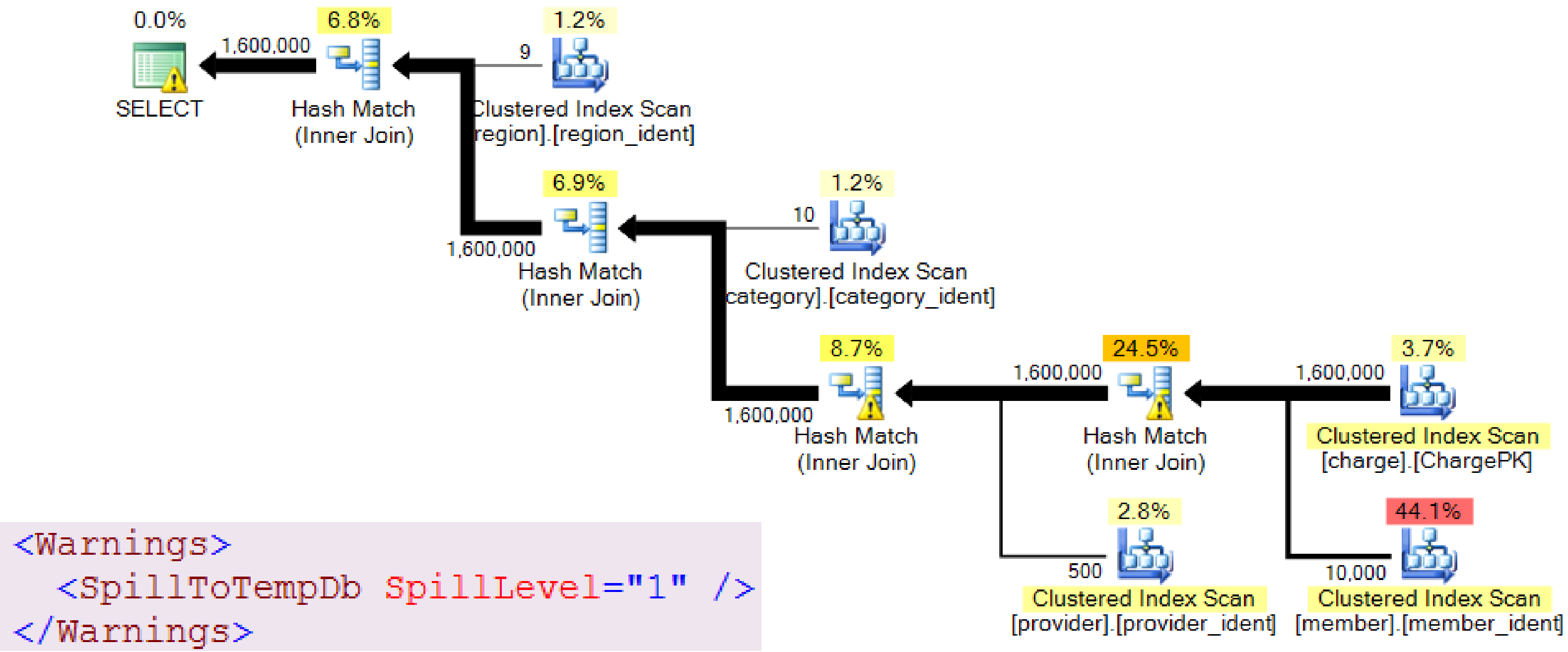
- **Each type of operator requires varying amounts of memory in order to perform the associated operation**
- **Some operators require more memory because they cache rows**
  - More rows = more memory required
    - SQL Server performs estimates of the required memory and tries to reserve the memory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

Memory Operators

- **Memory consuming operators include:**
  - Hash Match
  - Sort
- **When available memory is insufficient, queries that require a lot of memory may wait to execute**
- **Under-estimating memory (due to cardinality estimate issues) can cause memory spills to tempdb (I/O)**
- **Over-estimating memory can reduce concurrency!**

Under-estimates and Spills

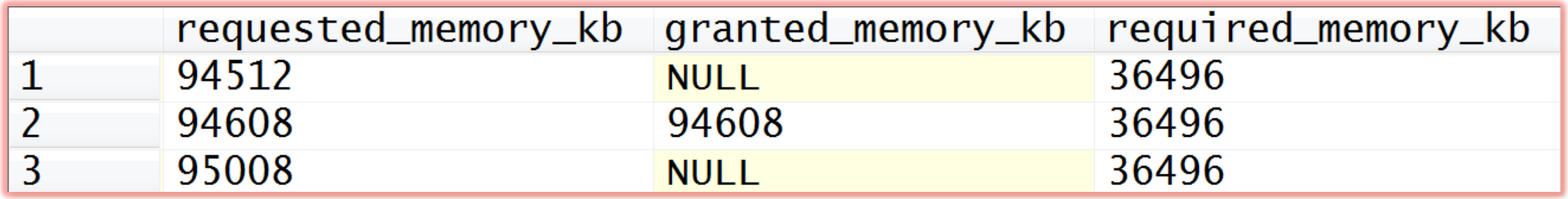
- **Significant under-estimates?**
  - Hash and Sort operations example
  - Memory grants may be insufficient
  - Risk of spills to disk, excessive I/O



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

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

Impacted Query Optimizer Decisions

- **Decisions impacted by cardinality estimates?**
  - Index selection
  - Seek or scan
  - Parallel or serial plan execution
  - Join algorithms
  - Inner or outer table selection
  - Spool generation
  - Bookmark lookups
  - Stream or hash aggregates
  - Wide or narrow updates

Excessive Resource Consumption

- **Bad plans can be responsible for**
  - Excessive I/O
  - Inflated CPU
  - Memory pressure
  - Decreased throughput
  - Reduced concurrency

- **The takeaway?**
  - Don’t ignore cardinality estimate issues