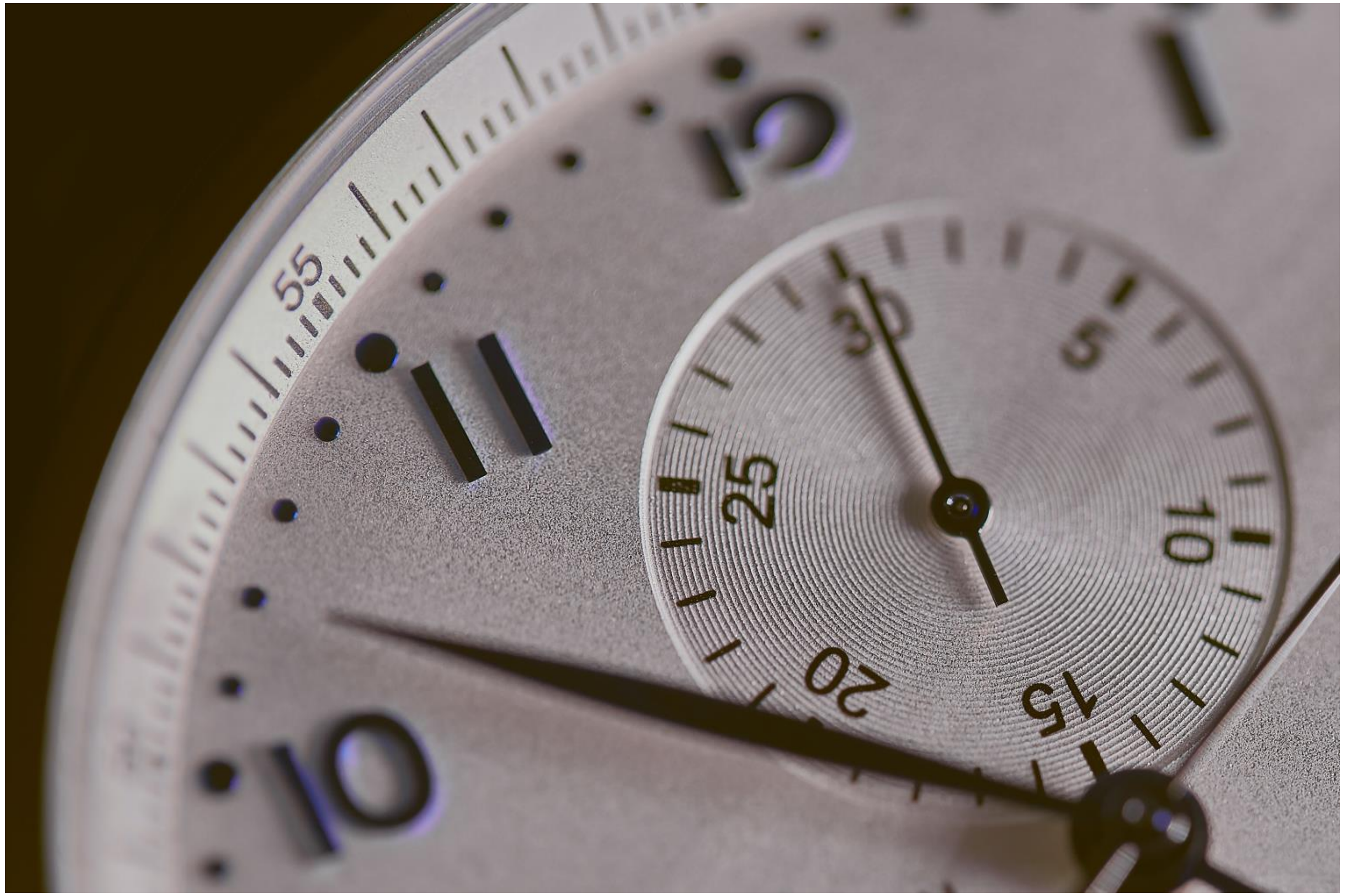
Optimizing Queries in SQL Server

by Maria Barnes

Barnes Business Solutions, Inc



Optimizing Queries in SQL Server

by Maria Barnes

Barnes Business Solutions, Inc

- How to decide which SQL queries need to be optimized?
- How to Use the Activity Monitor in SSMS
- An Introduction to the Performance Dashboard for SSMS 17.2 and above
- Query Simplification Basics
- Displaying the Estimated Execution Plan in SSMS
- A brief mention of the Query Store and what this is

Optimizing Queries in SQL Server

- How to decide which SQL queries need to be optimized?

Consistently Slow Queries

Occasionally Slow Queries

At least monitor

Queries With Red Flags

Return warnings or errors

Queries That Majorly Contribute to Total Execution Time

More than 5 % of total execution time

Optimizing Queries in SQL Server

- How to decide which SQL queries need to be optimized?

Consistently Slow Queries – cause can be hardware constraints, suboptimal query structure, missing indexes, poor choice of query plan by the optimizer

Hardware constraints – see SOS_SCHEDULER_YIELD & CPACKET wait types, or PAGEIOLATCH_SH waits

Suboptimal query structure – row-based operation with cursors or WHILE loops. Table-valued functions (prior to 2017), scalar functions (improved in level 140 & 150).

Occasionally Slow Queries – most common cause is data skew, also blocking or hardware contention

SARGability – WHERE with column in an index, not SARG if LIKE starts with wildcard, or when use functions like CONVERT on column

Missing Indexes

Missing or out-of-date statistics

Poor optimizer choices – QUERY HINT

Parameter sniffing

Optimizing Queries in SQL Server

How to Use the Activity Monitor in SSMS

- Overview pane (most important)
- Processes pane – who is running what
- Resource Waits pane
- Data File I/O pane
- Recent Expensive Queries (from last 30 seconds)
- Active Expensive Queries

Need to have VIEW SERVER STATE permissions
To view Data File I/O pane also need
CREATE DATABASE,
ALTER ANY DATABASE, or
VIEW ANY DEFINITION permissions

To start it right click on the Server instance or select
The Icon on the toolbar

Panes can be expanded and collapsed

Right clicking on the Overview header gives you a
Submenu where you can change Refresh Interval from
1 second to 1 hour. Should be 10 seconds or higher.

Optimizing Queries in SQL Server

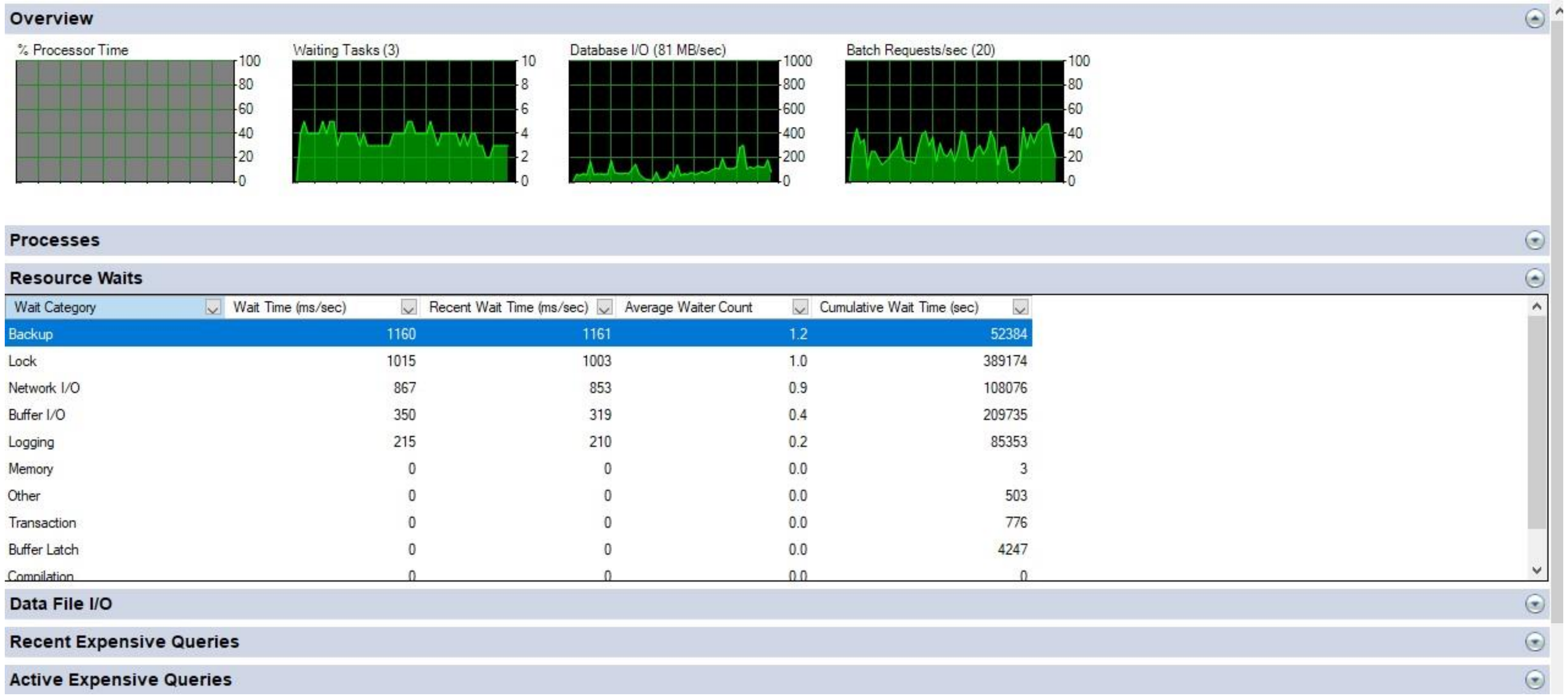
Activity Monitor Overview Pane

- *% Processor Time* – is the percentage of time the processors spend to execute threads that are not idle
- *Waiting Tasks* – is the number of tasks that are waiting for processor, I/O, or memory to be released so the tasks can be processed
- *Database I/O* – is the data transfer rate in MB/s from memory to disk, disk to memory, or disk to disk
- *Batch Requests/sec* – is the number of SQL Server batches received by the instance in a second

The Processes Pane

- By right clicking on the process you can get details about last command or kill it (if you are a member of sysadmin or processadmin server roles.

Optimizing Queries in SQL Server
Activity Monitor Overview Pane

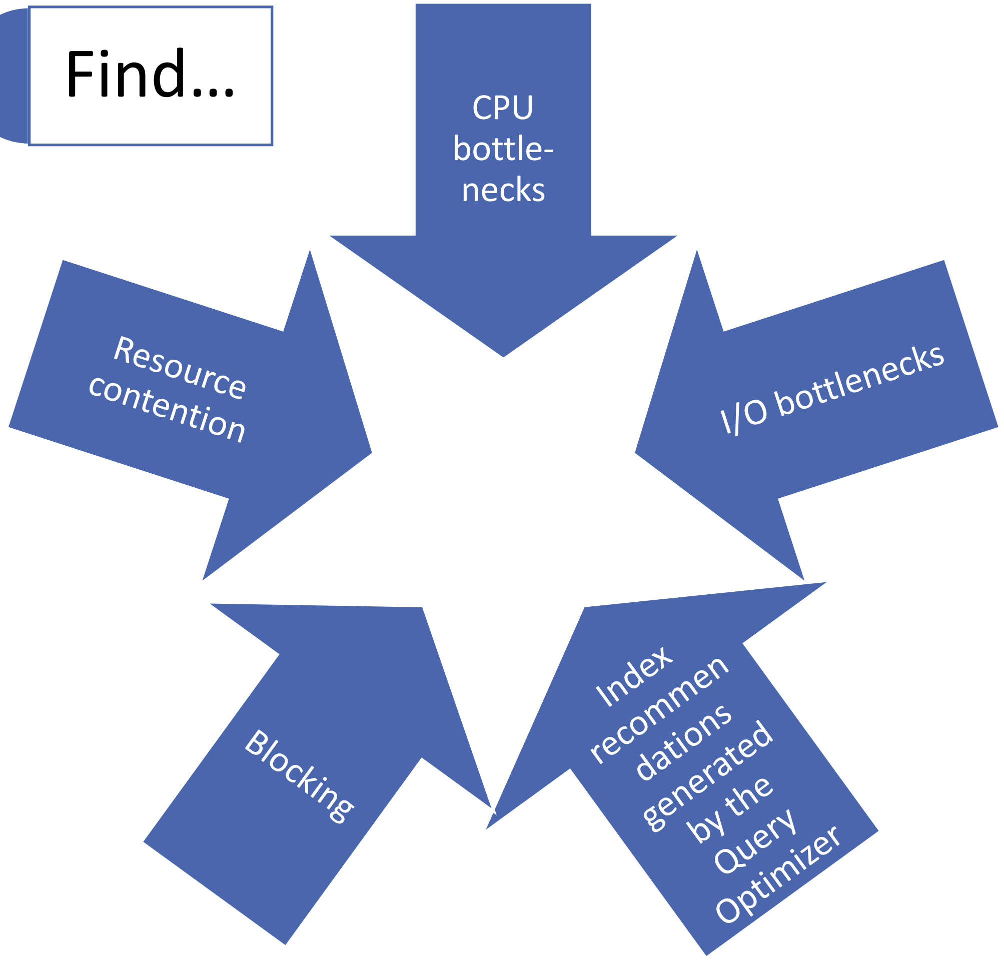


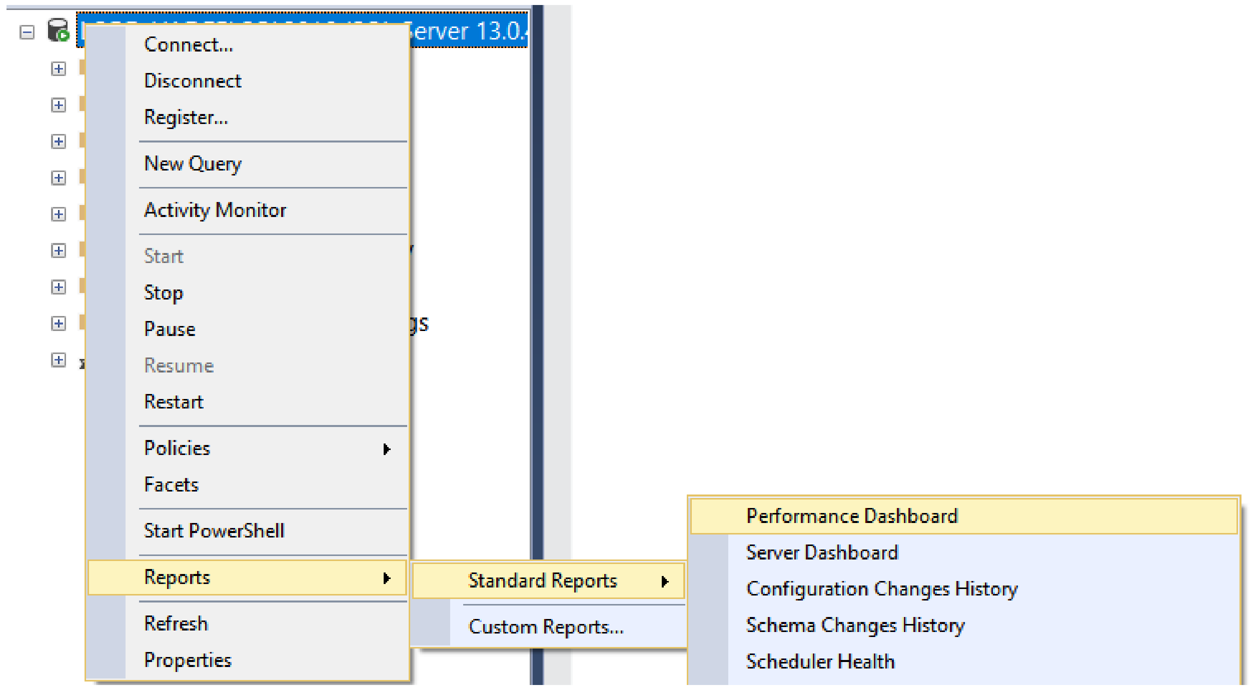
Wait Category	Wait Time (ms/sec)	Recent Wait Time (ms/sec)	Average Waiter Count	Cumulative Wait Time (sec)
Backup	1160	1161	1.2	52384
Lock	1015	1003	1.0	389174
Network I/O	867	853	0.9	108076
Buffer I/O	350	319	0.4	209735
Logging	215	210	0.2	85353
Memory	0	0	0.0	3
Other	0	0	0.0	503
Transaction	0	0	0.0	776
Buffer Latch	0	0	0.0	4247
Compilation	0	0	0.0	0

Optimizing Queries in SQL Server

An Introduction to the Performance Dashboard for SSMS 17.2 and above

Find…





Optimizing Queries in SQL Server

An Introduction to the Performance Dashboard for SSMS 17.2 and above

System CPU Utilization

Current Waiting Requests

Current Activity

Historical Information

- Waits
- Latches
- I/O Statistics
- Expensive Queries – By CPU, Duration, Logical or Physical Reads, Logical Writes, CLR Time

Miscellaneous

- Active Traces
- Active xEvent Sessions
- Databases
- Missing Indexes - Microsoft recommends that indexes with a score greater than 100,000 should be evaluated for creation, as those have the highest anticipated improvement for user queries.

Optimizing Queries in SQL Server

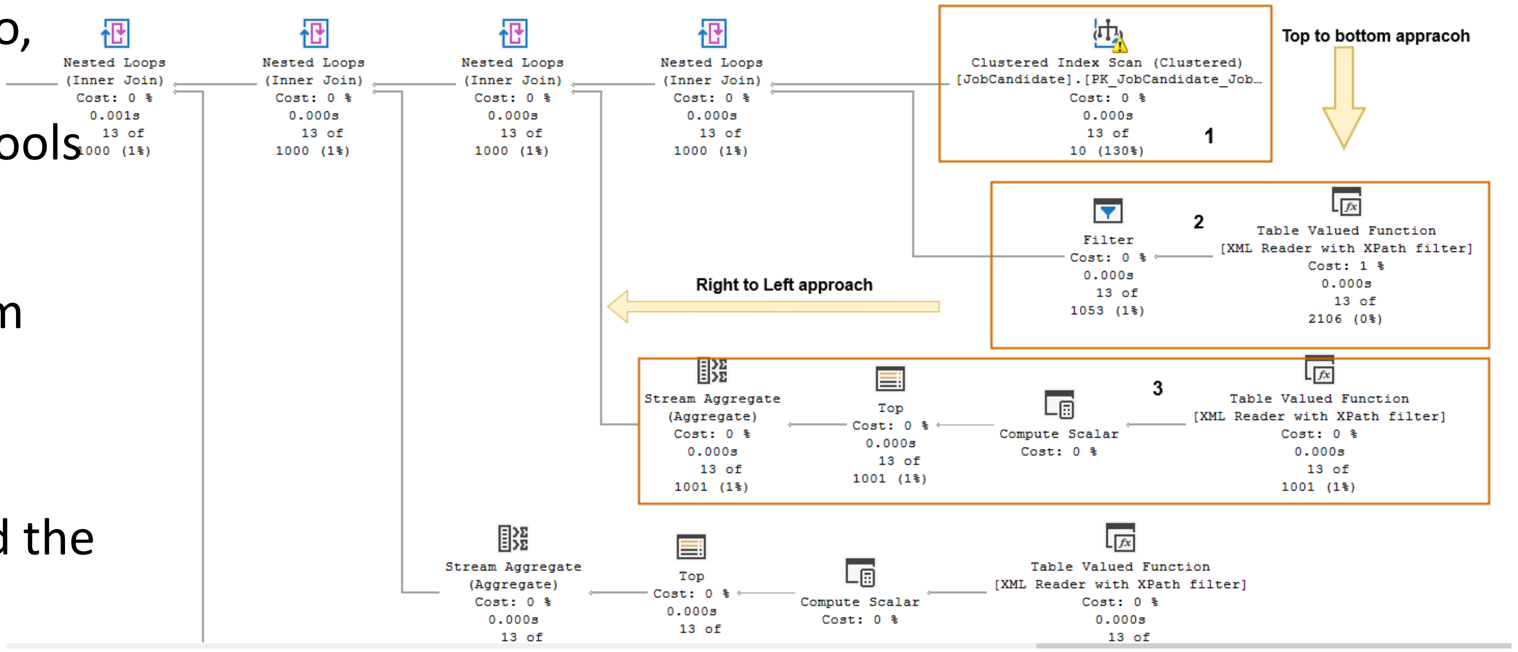
Query Simplification Basics

- Reduce Table Size
- Simplify Joins - Start with table that will return fewest results after filtering
- Use SELECT fields FROM instead of SELECT * FROM
- Use EXISTS() instead of COUNT()
- Use IN() instead of EXISTS() if subquery small
- Use WHERE instead of HAVING
- Create Indexes – clustered and non-clustered
- Avoid loops when you can use bulk insert or update

Optimizing Queries in SQL Server

Displaying the Estimated Execution Plan in SSMS

- From SSMS, estimated & actual
- Can also be done from Azure Data Studio, SQL Server profiler, Extended events, Dynamic management views, 3rd party tools
- Graphical, XML, Text
- Read from right to left and top to bottom
- Starts with more detailed steps. As you move to the left you see joins.
- Pay attention to the types of objects and the cost relative to the plan
- You can right click on plan and select "Analyze Actual Execution Plan"


Nested Loops (Inner Join) Cost: 0 % 0.001s 13 of 1000 (1%)
Nested Loops (Inner Join) Cost: 0 % 0.000s 13 of 1000 (1%)
Nested Loops (Inner Join) Cost: 0 % 0.000s 13 of 1000 (1%)
Nested Loops (Inner Join) Cost: 0 % 0.000s 13 of 1000 (1%)
Clustered Index Scan (Clustered) [JobCandidate].[PK_JobCandidate_Job… Cost: 0 % 0.000s 13 of 10 (130%)
1
Top to bottom appracoh
Filter Cost: 0 % 0.000s 13 of 1053 (1%)
2
Table Valued Function [XML Reader with XPath filter] Cost: 1 % 0.000s 13 of 2106 (0%)
Right to Left approach
Stream Aggregate (Aggregate) Cost: 0 % 0.000s 13 of 1001 (1%)
Top Cost: 0 % 0.000s 13 of 1001 (1%)
Compute Scalar Cost: 0 %
3
Table Valued Function [XML Reader with XPath filter] Cost: 0 % 0.000s 13 of 1001 (1%)
Stream Aggregate (Aggregate) Cost: 0 % 0.000s 13 of
Top Cost: 0 % 0.000s 13 of
Compute Scalar Cost: 0 %
Table Valued Function [XML Reader with XPath filter] Cost: 0 % 0.000s 13 of

Optimizing Queries in SQL Server

Displaying the Estimated Execution Plan in SSMS - Terms	
Index Scan and Clustered Index Scan	This will read the entire index in order.
Index Seek and Clustered Index Seek	This will perform a B-tree traversal and find matching rows.
Key Lookup (Clustered)	This will find a single row from a clustered index.
RID Lookup	This will find a single row from a table.
Table Scan	This will read all rows and columns in the table. Also called a Full Table Scan. It’s an expensive operation.
Hash Match	Joins two tables by creating a hash table.
Nested Loops	Joins two tables by getting the result from one table and matching it to the other table.
Merge Join	Joins two tables that have been sorted by matching rows together.
Sort	Sorts the result of your query using the ORDER BY clause.

Optimizing Queries in SQL Server

Displaying the Estimated Execution Plan in SSMS – things to watch for

- Missing Indexes
- Table Scan should be avoided by adding an index or updating your query design
- Key Lookups
  - Fields currently being utilized in Seek Predicates.
  - Index in Object of Index above.
  - You can remove need for a Key Lookup by adding a covering index that includes all of the fields in the Output List.
  - You can improve further by adding any items in WHERE to top section, not INCLUDE section

Optimizing Queries in SQL Server

The Query Store

Collects plans for DML Statements (SELECT, INSERT, UPDATE, DELETE, MERGE, BULK INSERT)

Does not collect data for natively compiled stored procedures by default. Can turn on with **sys.sp_xtp_control_query_exec_stats**

Available with SQL Server 2016 and later

Enabled by default for new Azure SQL Databases and Azure SQL Managed Instance, and SQL Server 2022. Not enabled in SQL Server 2016, 2017, 2019.

```
    ALTER DATABASE AdventureWorks2017 SET QUERY_STORE = ON (
WAIT_STATS_CAPTURE_MODE = ON );
```

SQL Query Optimization

Query Store

- Regressed Queries
- Overall Resource Consumption
- Top Resource Consuming Queries
- Queries With Forced Plans
- Queries With High Variation
- Queries Wait Statistics
- Tracked Queries

Optimizing Queries in SQL Server

Maria Barnes

www.BarnesBusinessSolutions.com

mbarnes@ Barnes Business Solutions.com

@mbarnesatbbs

https://www.linkedin.com/in/mariabarnes

Microsoft Certified Technology Specialist, SQL Server 2008, Database Development,

Microsoft Certified in Azure Fundamentals and Azure Database Administrator

Resources

- Microsoft Learn Monitor https://learn.microsoft.com/en-us/sql/relational-databases/performance/monitor-and-tune-for-performance?view=sql-server-ver16
- Microsoft Learn – Optimize query performance in Azure SQL https://learn.microsoft.com/en-us/training/paths/optimize-query-performance-sql-server/
- SQL AdventureWorks2017 database https://github.com/MicrosoftLearning/dp-300-database-administrator/blob/master/Instructions/Templates/AdventureWorks2017.bak
- Execution plans https://blog.quest.com/sql-server-execution-plan-what-is-it-and-how-does-it-help-with-performance-problems/
- Isolate Problem Areas Poor Performing Queries https://github.com/MicrosoftLearning/dp-300-database-administrator/blob/master/Instructions/Labs/10-isolate-problem-areas-poor-performing-queries.md