Microsoft

SQL Server Query Execution and Plans

Module 6

Learning Units covered in this Module

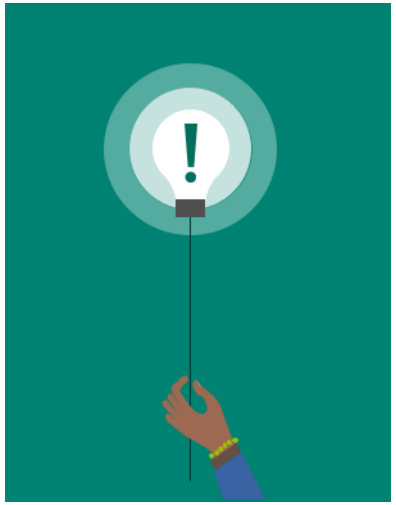
- Lesson 1: SQL Query Execution & Optimization
- Lesson 2: SQL Query Plan Analysis

Lesson 1: SQL Query Execution & Optimization

Objectives

After completing this learning, you will be able to:

- Explain Query Compilation and Optimization Process.
- Explain Query Execution Process.
- Explain Recompilation causes.



SQL Server Execution Plan

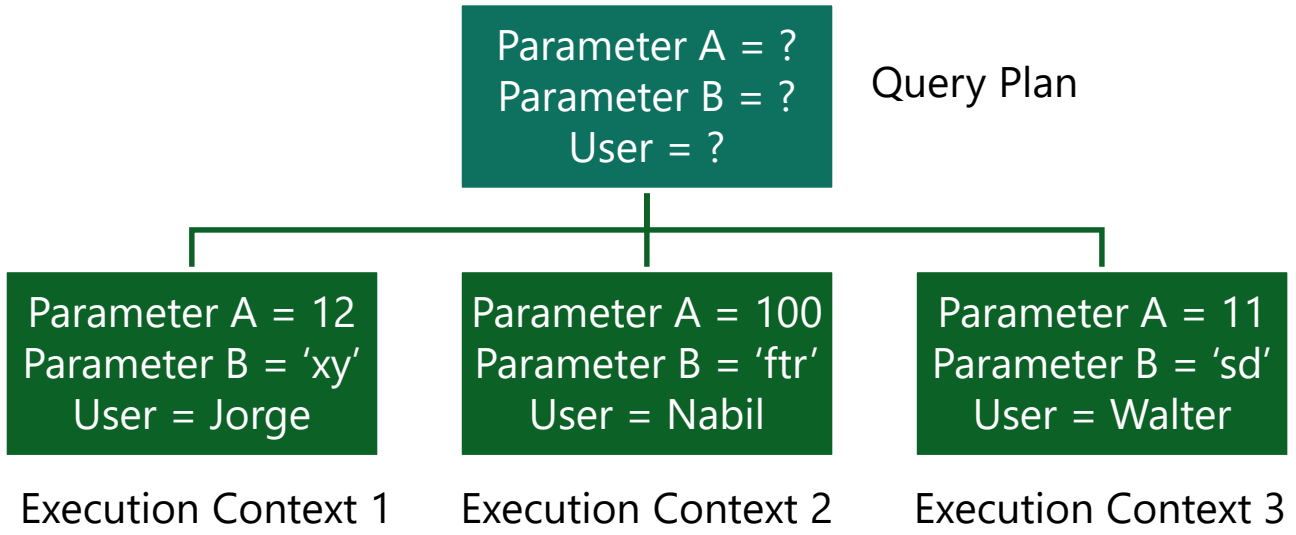
Main components

Compiled Plan (or Query Plan)

Compilation produces a query plan, which is a read-only data structure used by any number of users.

Execution Context

A data structure used to hold information specific to a query execution, such as parameter values.



SQL Server Execution Plan Caching

Overview

Part of the memory pool used to store execution plans – also known as plan cache.

The plan cache has two stores for all compiled plans:

The **Object Plans** cache store (OBJCP)

used for plans related to persisted objects (stored procedures, functions, and triggers).

The **SQL Plans** cache store (SQLCP)

used for plans related to autoparameterized, dynamic, or prepared queries.

SQL Server compilation and execution

Concepts

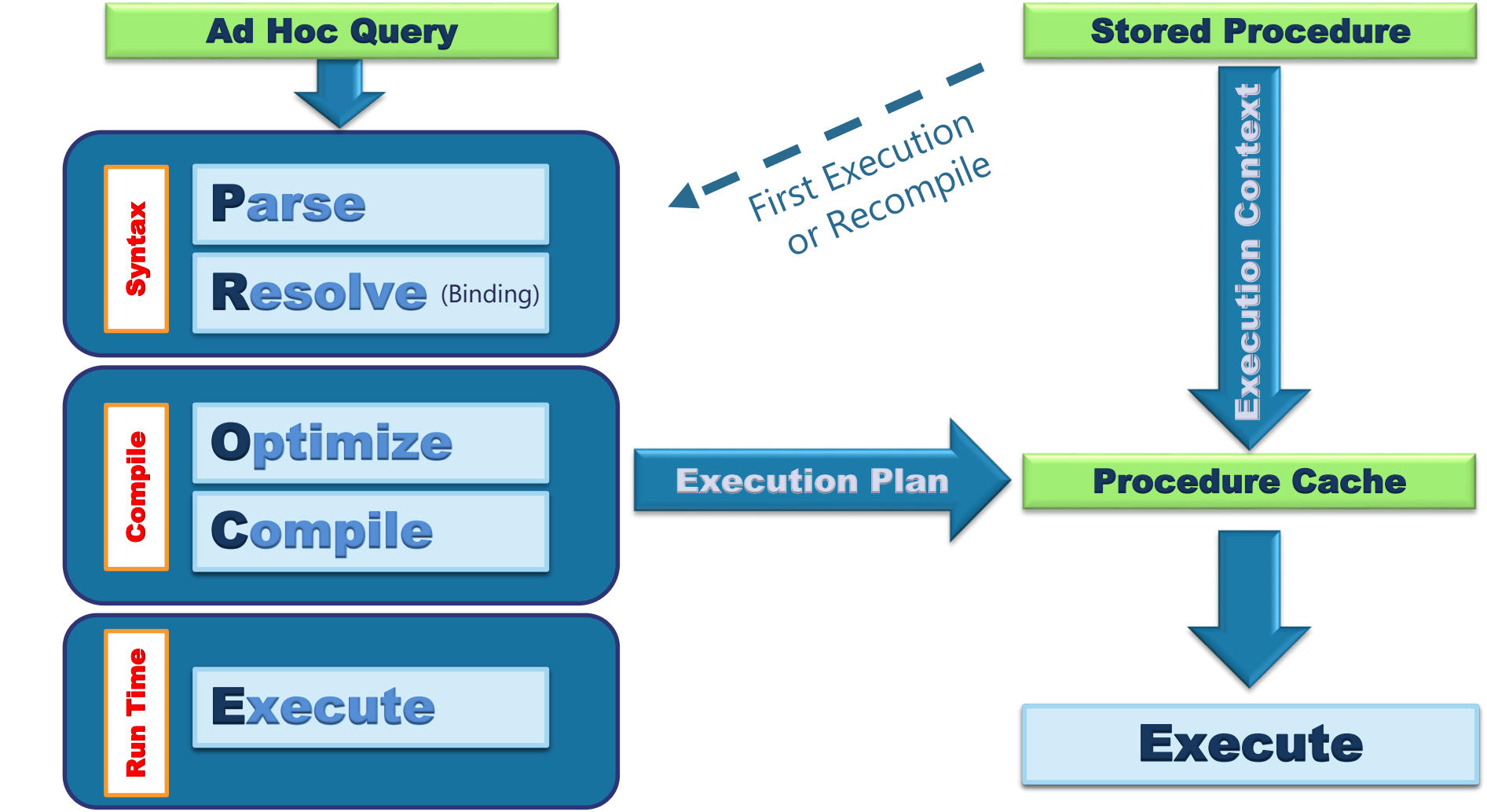
Compilation

Process of creating a good enough query execution plan, as quickly as possible for a query batch.

Refer to both the compilation of non-DML constructs in SQL statements (control flow, DDL, etc.) and the process of Query Optimization.

Query Execution

Process of executing the plan that is created during query compilation and optimization.



SQL

Sets

empid	lastname	firstna...	title	titleofcourt...	birthdate
1	Davis	Sara	CEO	Ms.	1958-12-08 00:00:00.000
2	Funk	Don	Vice President, Sales	Dr.	1962-02-19 00:00:00.000
3	Lew	Judy	Sales Manager	Ms.	1973-08-30 00:00:00.000
4	Peled	Yael	Sales Representative	Mrs.	1947-09-19 00:00:00.000
5	Buck	Sven	Sales Manager	Mr.	1965-03-04 00:00:00.000

What does the binding step resolve?

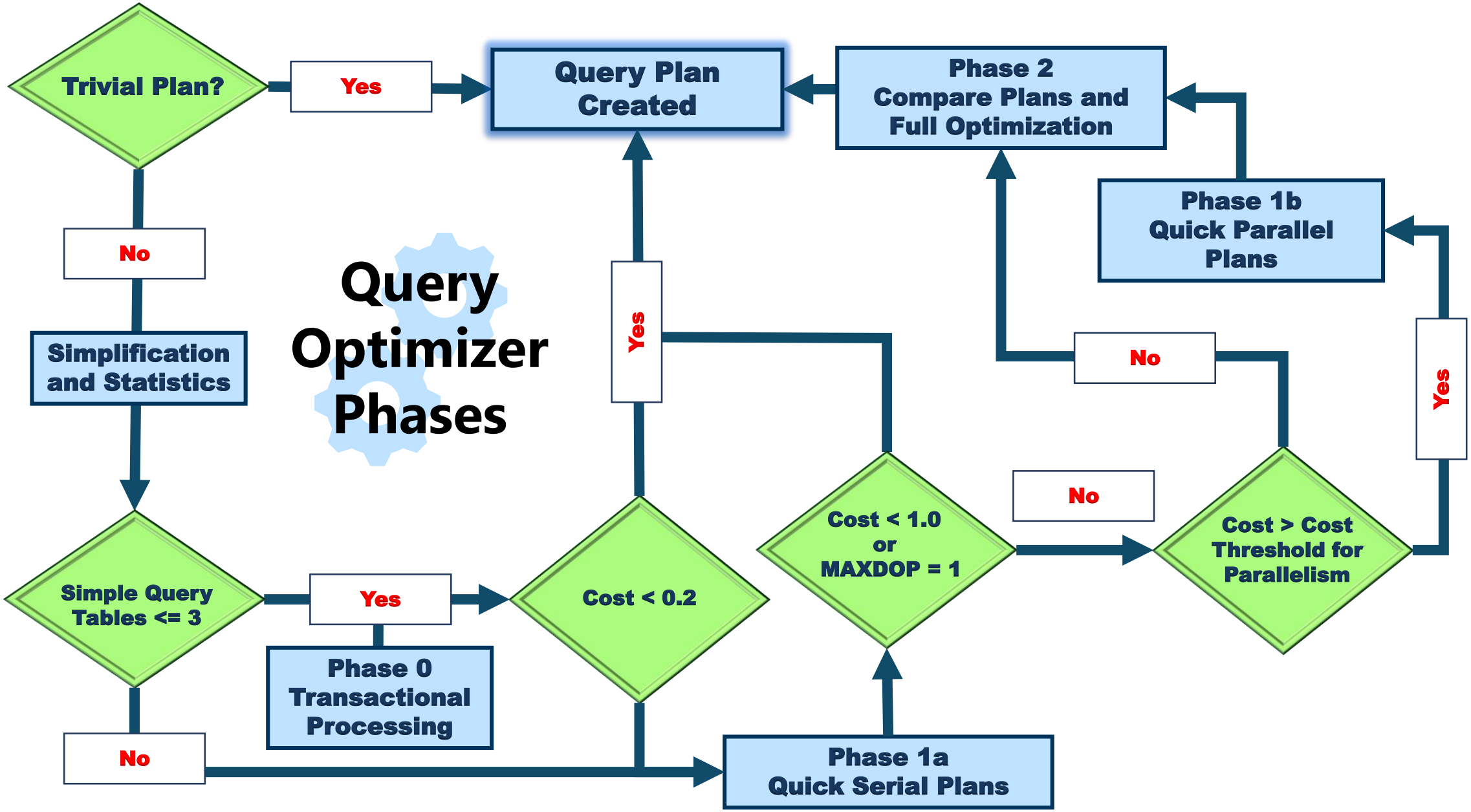
User permissions are checked.

Does a cached plan exist?

Object names (Tables, Views, Columns, etc.) to see if they exist.

Resolve aliases of columns and tables

Data types and if implicit data type conversions are needed.


Query Optimizer Phases
Trivial Plan?
Yes
Query Plan Created
Phase 2 Compare Plans and Full Optimization
Phase 1b Quick Parallel Plans
No
Simplification and Statistics
Yes
No
Yes
Simple Query Tables <= 3
Yes
Phase 0 Transactional Processing
Cost < 0.2
Cost < 1.0 or MAXDOP = 1
No
Cost > Cost Threshold for Parallelism
No
Phase 1a Quick Serial Plans

Query Simplification Process

Constant Folding: Expressions with constant values are reduced

- **Quantity = 2 + 3** becomes **Quantity = 5**
- **10 < 20** becomes **True**

Contradiction Detection: Removes criteria that doesn't match table constraints

- **Constraint:** Age > 18
- **Contradiction:** WHERE Age < 18

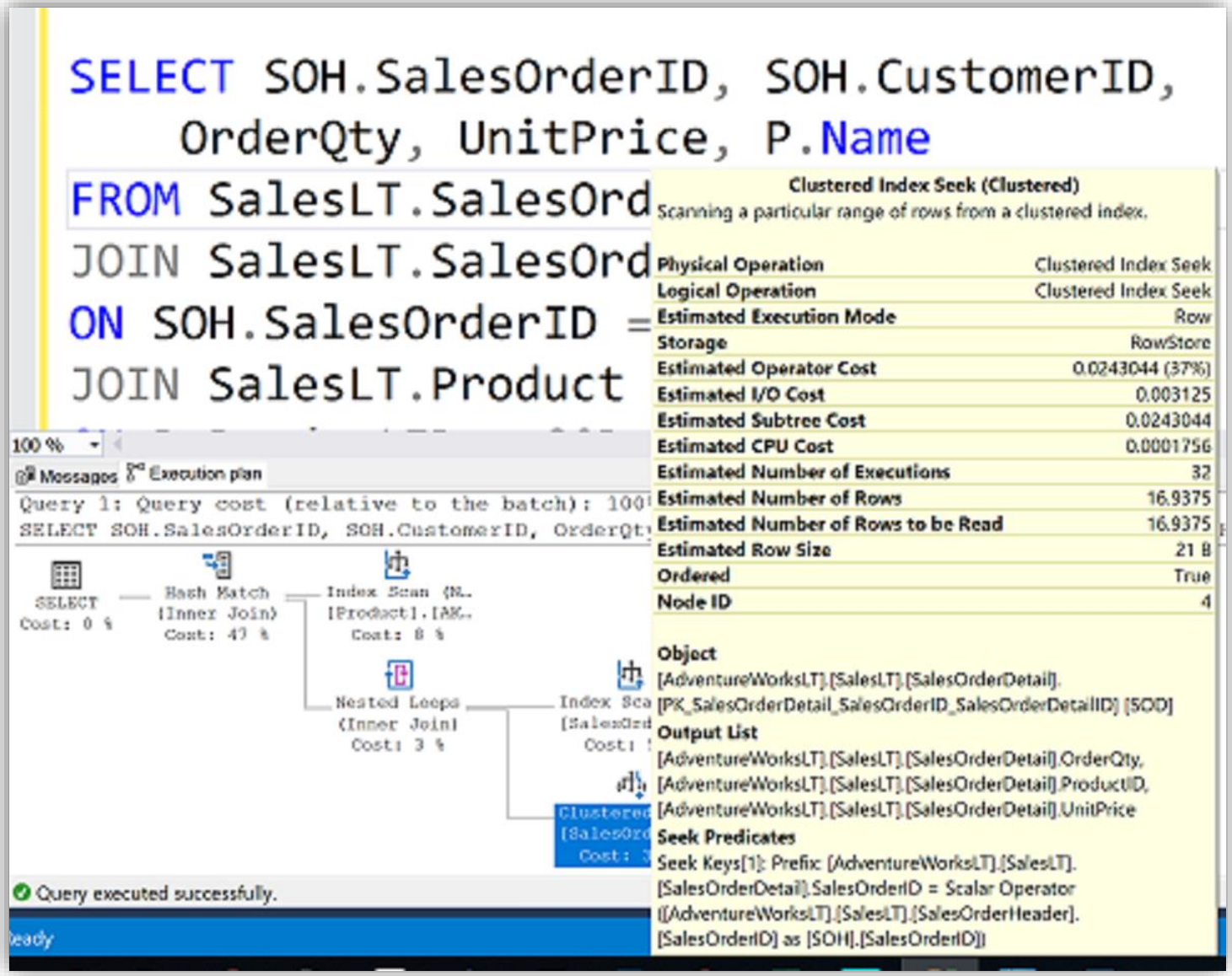
Domain Simplification: Reduces complex ranges to simple ranges

- **Complex range:** ID > 10 and ID < 20 or ID > 30 and < 50
- **Simplified range:** ID > 10 and < 50

Join Simplification: Removes redundant joins that are not necessary

Predicate Pushdown: Perform calculations only on rows returned

What is an Execution Plan?



How to see the query plan

Graphical execution plan

Estimated Execution Plan (Before Execution)

- The compiled plan.

Actual Execution Plan (After Execution)

- The same as the compiled plan plus its execution context.
- This includes runtime information available after the execution completes, such as execution warnings, or in newer versions of the Database Engine, the elapsed and CPU time used during execution.

Live Query Statistics (During Execution)

- The same as the compiled plan plus its execution context.
- This includes runtime information during execution progress and is updated every second. Runtime information includes for example the actual number of rows flowing through the operators.
- Enables rapid identification of potential bottlenecks.

SQL Server Execution Plan Recompilations

Overview

Most recompilations are required either for statement correctness or to obtain potentially faster query execution plan.

The engine detects changes that invalidate execution plan(s) and marks those as not valid. New plan must be recompiled for the next query execution.

Starting with SQL Server 2005, whenever a statement within a batch causes recompilation, only the statement inside the batch that triggers recompilation is recompiled.

SQL Server Execution Plan Recompilations

Recompilation reasons

Table / Index Changes

- Changes made to objects referenced by the query (ALTER TABLE and ALTER VIEW).
- Changing or dropping any indexes used by the execution plan.

Stored Procedures

- Changes made to a single procedure, which would drop all plans for that procedure from the cache (ALTER PROCEDURE).
- Explicit call to sp_recompile.
- Executing a stored procedure using the WITH RECOMPILE option.

Data Volume

- Updates on statistics used by the execution plan
- For tables with triggers, if the number of rows in the inserted or deleted tables grows significantly.

Other

- Large numbers of changes to keys (generated by statements from other users that modify a table referenced by the query).
- Temporary table changes

Questions?

Knowledge Check

What is meant by SQL Server's query optimizer being **cost-based**?

When is a query considered for a parallel execution plan?

Will SQL Server evaluate **every** possible query plan in the process of optimization? Why?

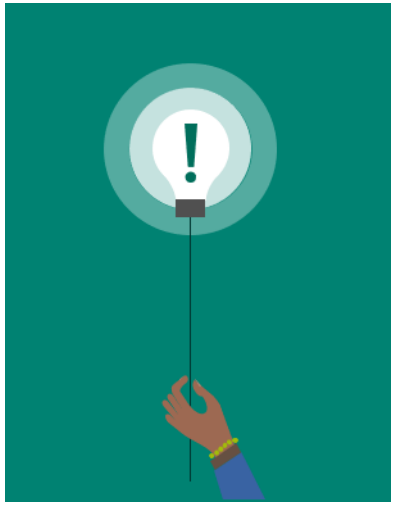
Name two recompilation causes.

Lesson 2: SQL Server Query Plan Analysis

Objectives

After completing this learning, you will be able to:

- Read execution plans.
- Understand logical and physical join operators.
- Describe data access.



How to see the query plan

Graphical execution plan

Estimated Execution Plan (Before Execution)

- The compiled plan.

Actual Execution Plan (After Execution)

- The same as the compiled plan plus its execution context.
- This includes runtime information available after the execution completes, such as execution warnings, or in newer versions of the Database Engine, the elapsed and CPU time used during execution.

Live Query Statistics (During Execution)

- The same as the compiled plan plus its execution context.
- This includes runtime information during execution progress and is updated every second. Runtime information includes for example the actual number of rows flowing through the operators.
- Enables rapid identification of potential bottlenecks.

Contents of an Execution Plan

- Sequence in which the source tables are accessed.
- Methods used to extract data from each table.
- How data is joined
- Use of temporary worktables and sorts
- Estimated rowcount, iterations, and costs from each operator
- Actual rowcount and iterations

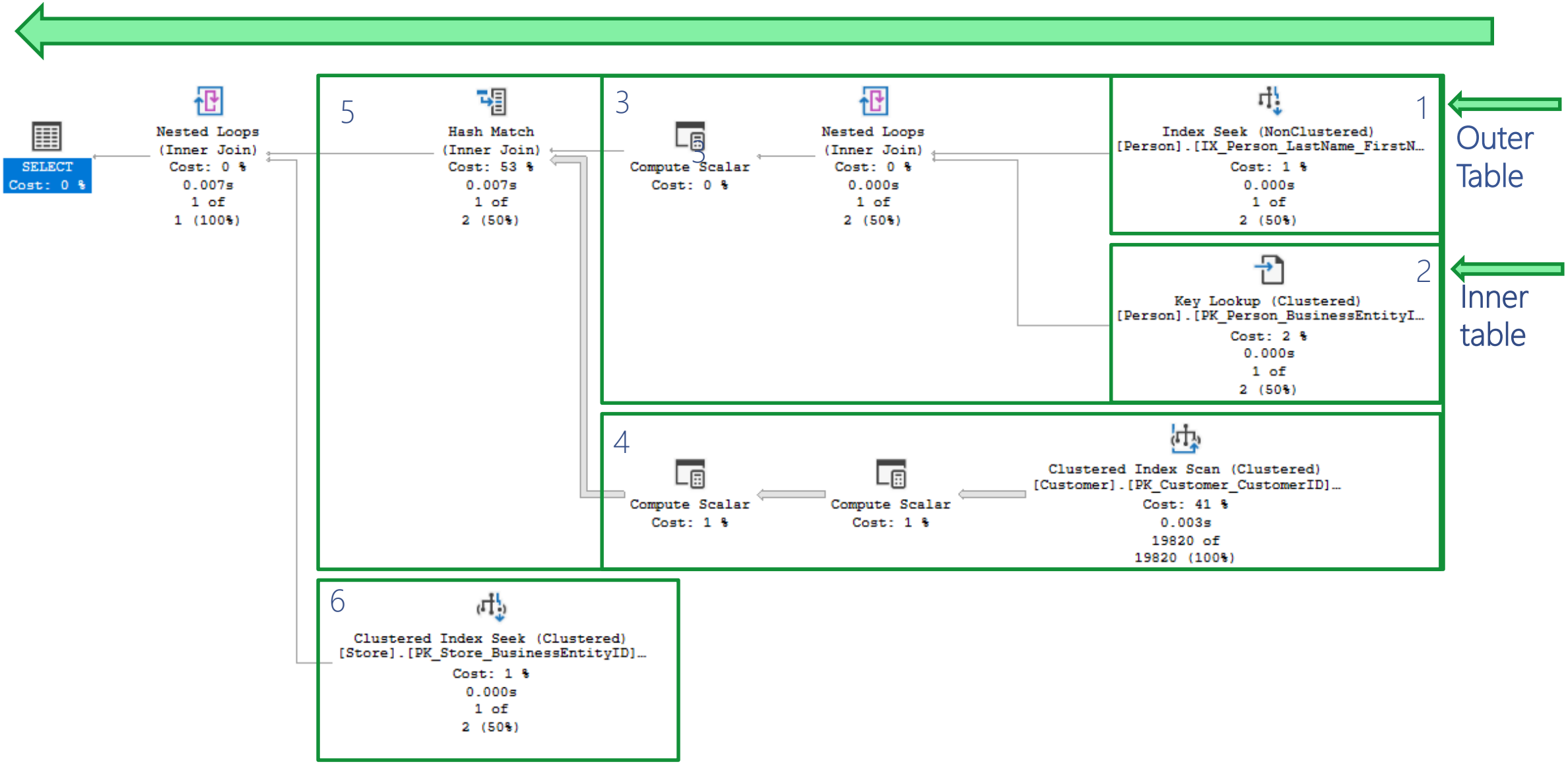
How to see the query plan

Text and XML

	Command	Execute query?	Include estimated row counts & stats (Estimated Query Plan)	Include actual row counts & stats (Actual Query Plan)
Text Plan	SET SHOWPLAN_TEXT ON	No	No	No
	SET SHOWPLAN_ALL ON	No	Yes	No
	SET STATISTICS PROFILE ON	Yes	Yes	Yes
XML Plan	SET SHOWPLAN_XML ON	No	Yes	No
	SET STATISTICS PROFILE XML	Yes	Yes	Yes

SSMS Graphical Plan

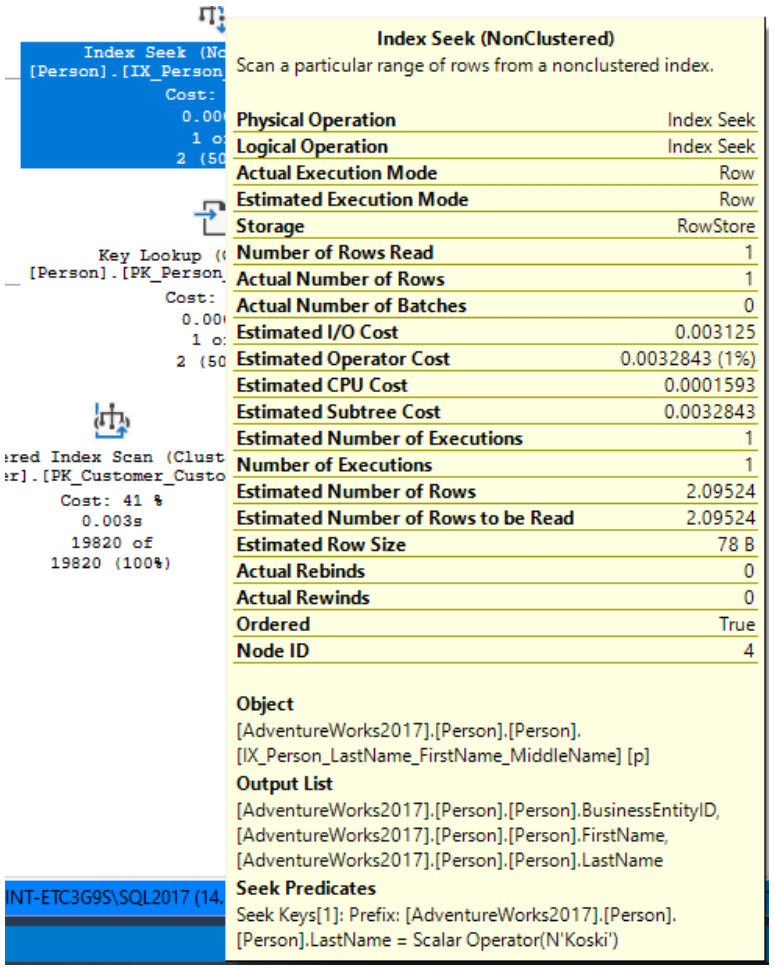
Execution Flow



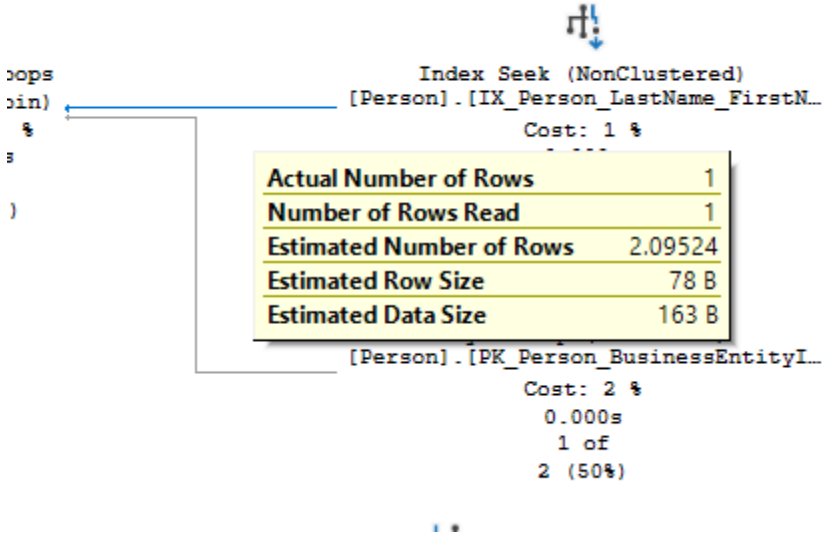
SSMS Graphical Plan

Data flow between the operators and statistical data of each operator

Statistical data for the operator

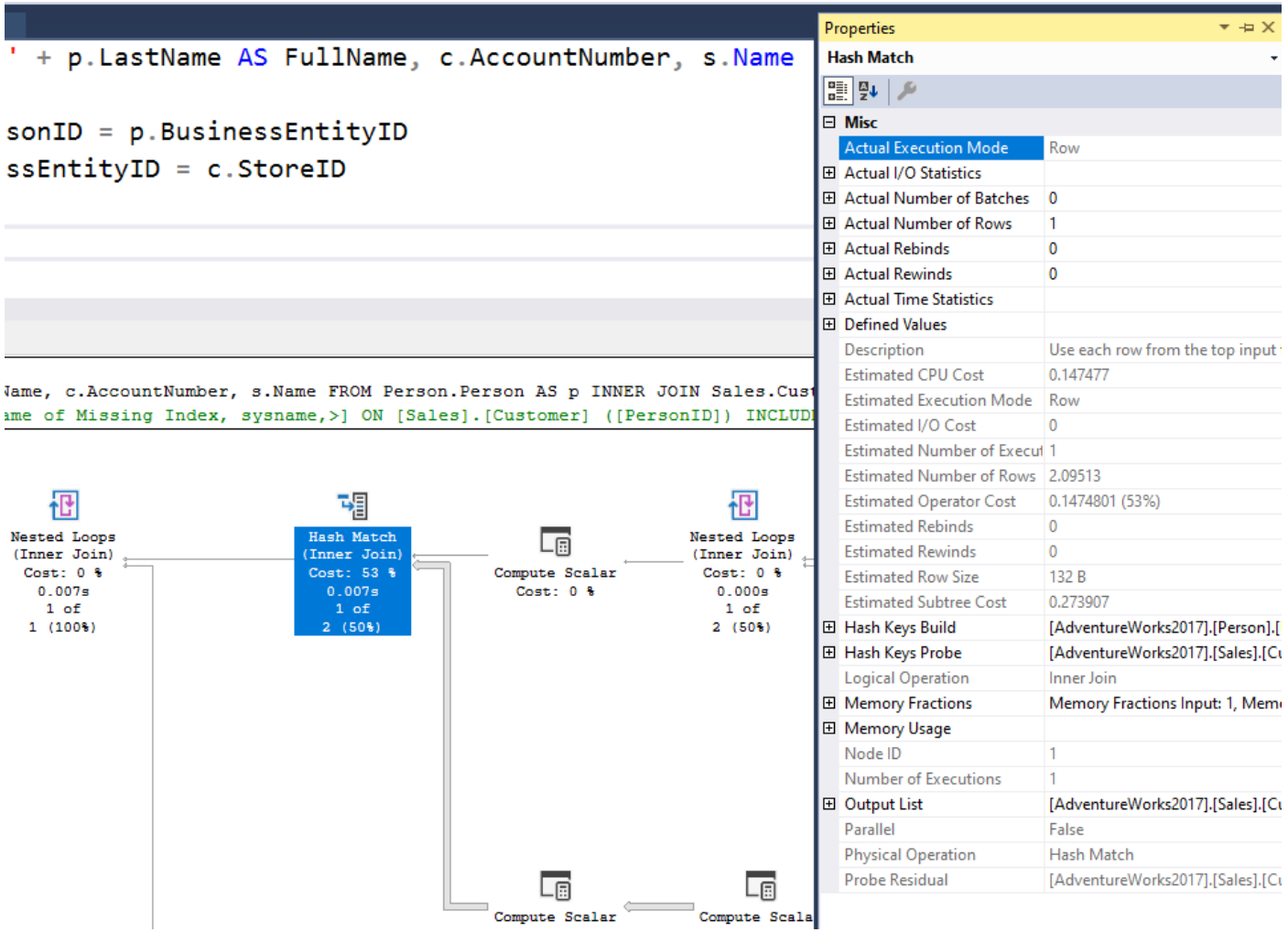


Data flow statistics



SSMS Graphical Plan

Properties sheet



Management Studio Properties sheet includes even more detailed information about each operator and about the overall query plan.

Use the most recent version of Management Studio as every new version display more detailed information about the Query Plan when examining the plan in graphical mode.

SSMS Graphical Plan

Live Query Statistics

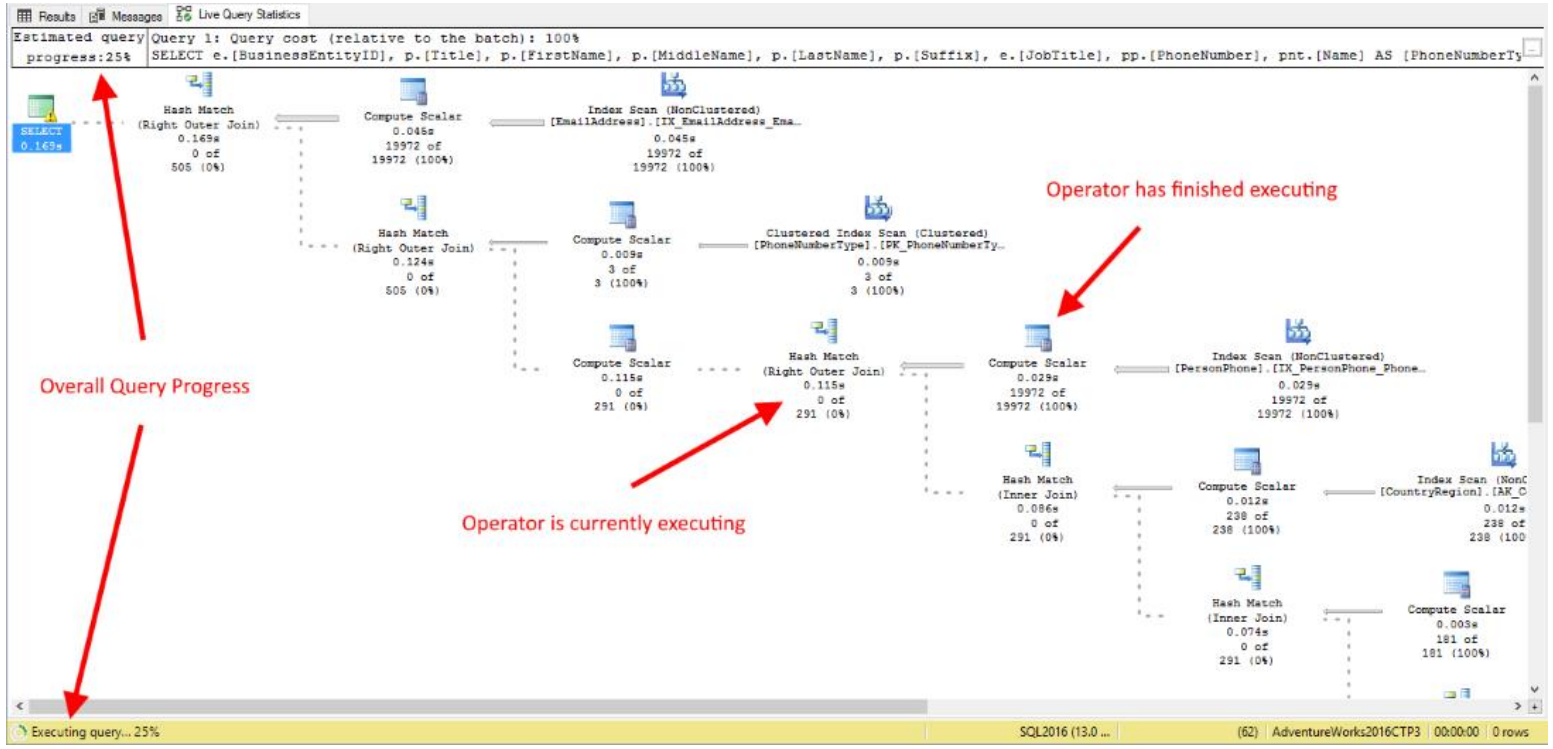
View CPU/memory usage, execution time, and query progress

Allows drill down to live operator-level statistics, such as:

- Number of generated rows
- Elapsed time
- Operator progress
- Live warnings

This feature is primarily intended for troubleshooting purposes.

Using this feature can moderately slow the overall query performance.



Execution Plan

Notable operators

Operators describe how SQL Server executes a query. The query optimizer uses operators to build a query plan to create the result specified in the query.


Table scan


Clustered index scan


Clustered index seek


RID lookup


Key lookup


ColumnStore Index Scan

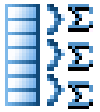

Nonclustered index scan


Nonclustered index seek


Sort


Table spool


Index spool


Stream Aggregation

Execution Plan Join Operators (Code)

```
SELECT SOH.SalesOrderID, SOH.CustomerID,
    OrderQty, UnitPrice, P.Name
FROM SalesLT.SalesOrderHeader as SOH
JOIN SalesLT.SalesOrderDetail AS SOD
ON SOH.SalesOrderID = SOD.SalesOrderID
JOIN SalesLT.Product AS P
ON P.ProductID = SOD.ProductID
```

Execution Plan Join Operators

A Merge Join is useful if both table inputs are in the same sorted order on the same value.

Merge Join
(Inner Join)
Cost: 39 %

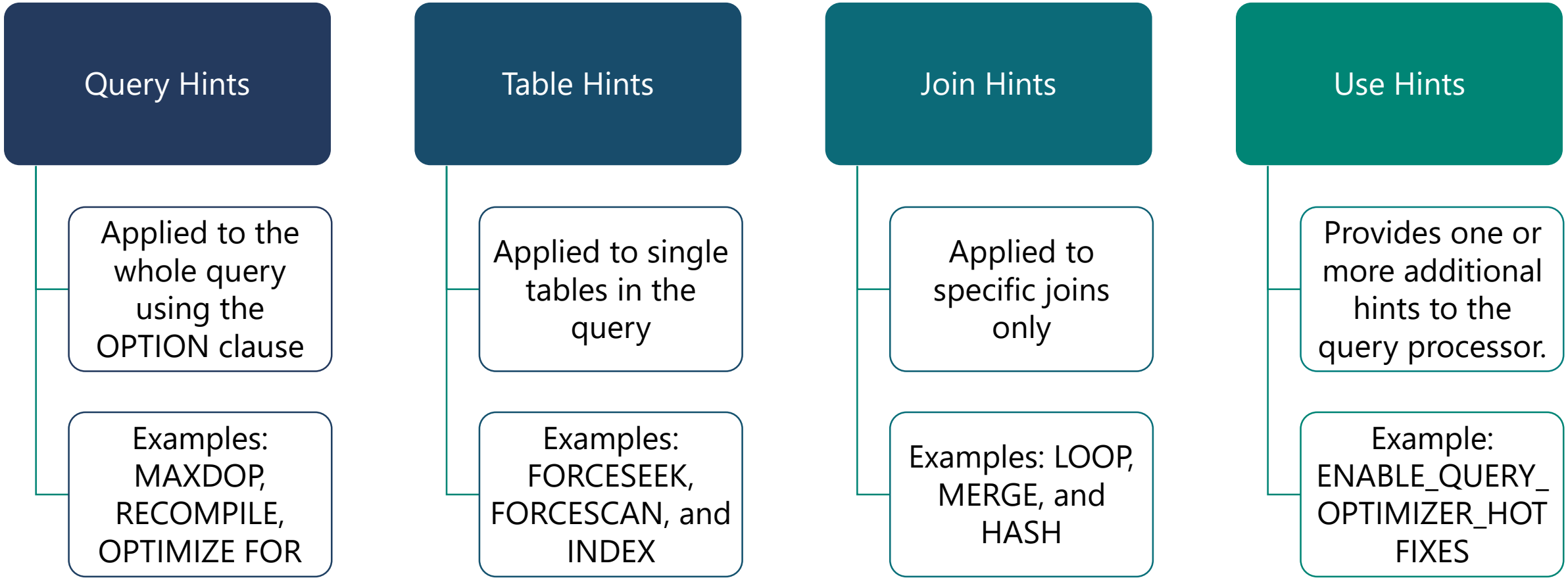
A Hash Match is used when the tables being joined are not in the same sorted order.

Hash Match
(Inner Join)
Cost: 47 %

A Nested Loop is use when a small (outer) table is used to lookup a value in a larger (inner) table.

Nested Loops
(Inner Join)
Cost: 3 %

Query Hint Types



What to look for in the query plan

Warnings	• Information about possible issues with the plan
Left Most Operator	• Overall properties of the plan to establish baseline
Right to Left	• Solve issues early in the plan
Expensive Operators	• Look from most expensive to least expensive
Sort Operators	• Locate why there is a sort operation and is it needed
Data Flow Statistics	• Thicker arrows mean more data is being passed
Nested Loop Operator	• Possible to create index that covers query
Scans vs Seeks	• Not necessarily bad, but could indicate I/O issues
Skewed Estimates	• Statistics could be stale or invalid

Demonstration

Query Plan Analysis

- Use the graphical execution plan and IO statistics to tune a query.
- Explore Live Query Statistics.

Query Plan Analysis

- Identifying and tuning high cost operators in the query plan

Questions?

Knowledge Check

What are the physical join operators?

What is a method to eliminate a lookup?

Is it recommended to eliminate all lookups operators?

Under what circumstances would a table scan be more efficient than an index seek on a non-clustered, non-covering index?

Is a Clustered Index Scan more efficient than a Table Scan?