INTELIGENTNO PROCESIRANJE U SQL SERVERU 2019

MILOŠ RADIVOJEVIĆ

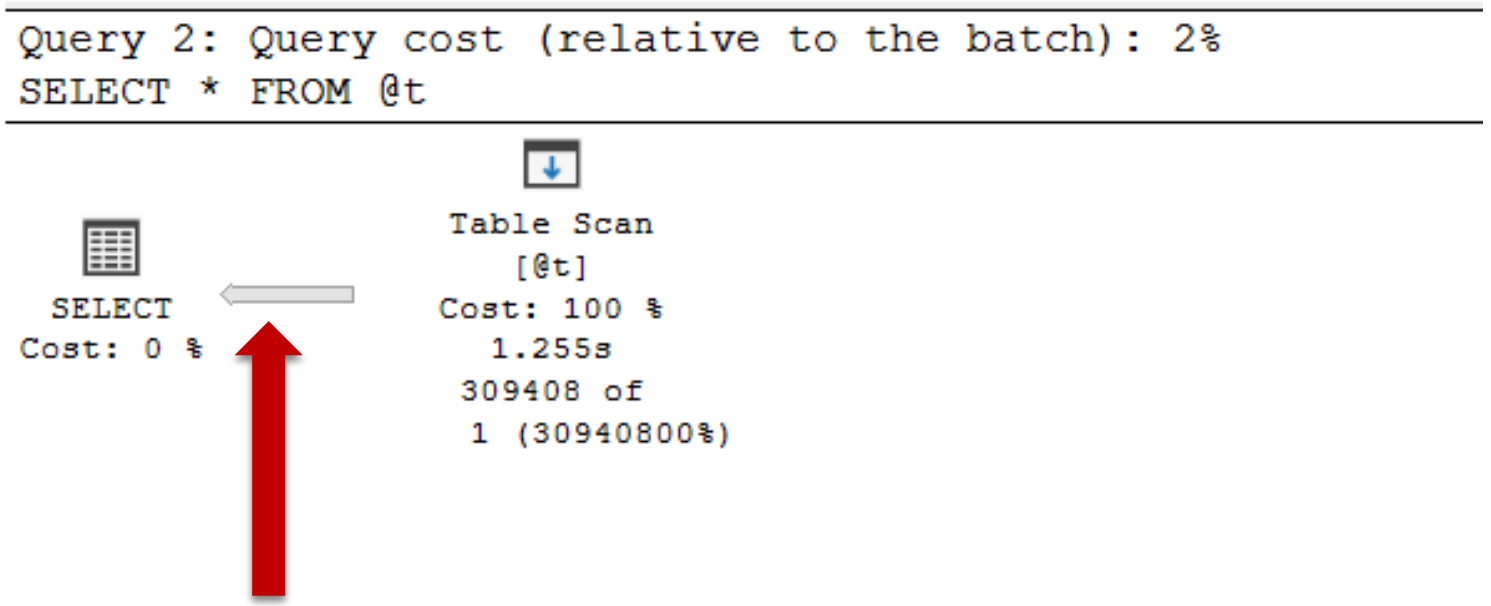
DATA PLATFORM MVP

26. DECEMBAR 2020

TABLE VARIABLE DEFERRED COMPILATION

PROBLEMS WITH TABLE VARIABLES

```
DECLARE @t TABLE(id INT)
INSERT INTO @t SELECT message_id FROM sys.messages;
SELECT * FROM @t;
```



Actual Number of Rows	309408
Number of Rows Read	309408
Estimated Number of Rows	1
Estimated Row Size	11 B
Estimated Data Size	11 B

- Unangemessene Operatoren im Ausführungsplan
- Unzureichende Speicherzuweisungen

PROBLEMS WITH TABLE VARIABLES

- Queries with table variables with many rows (> 10 KB) can have an inefficient plan
  - Inappropriate operators in the execution plan
  - Mostly nested loops join instead of hash join

- Insufficient memory allocations
- The number of rows is usually underestimated => less memory reserved for the query => tempdb spills

TABLE VARIABLE DEFERRED COMPILATION

```
DECLARE @T AS TABLE (ProductID INT);
INSERT INTO @T SELECT ProductID
FROM Production.Product
WHERE ProductLine IS NOT NULL;
```

```
SELECT * FROM @T t
INNER JOIN Sales.SalesOrderDetail od
ON t.ProductID = od.ProductID
INNER JOIN Sales.SalesOrderHeader h
ON h.SalesOrderID = od.SalesOrderID
ORDER BY od.UnitPrice DESC;
```

t_0

t_1

t_2

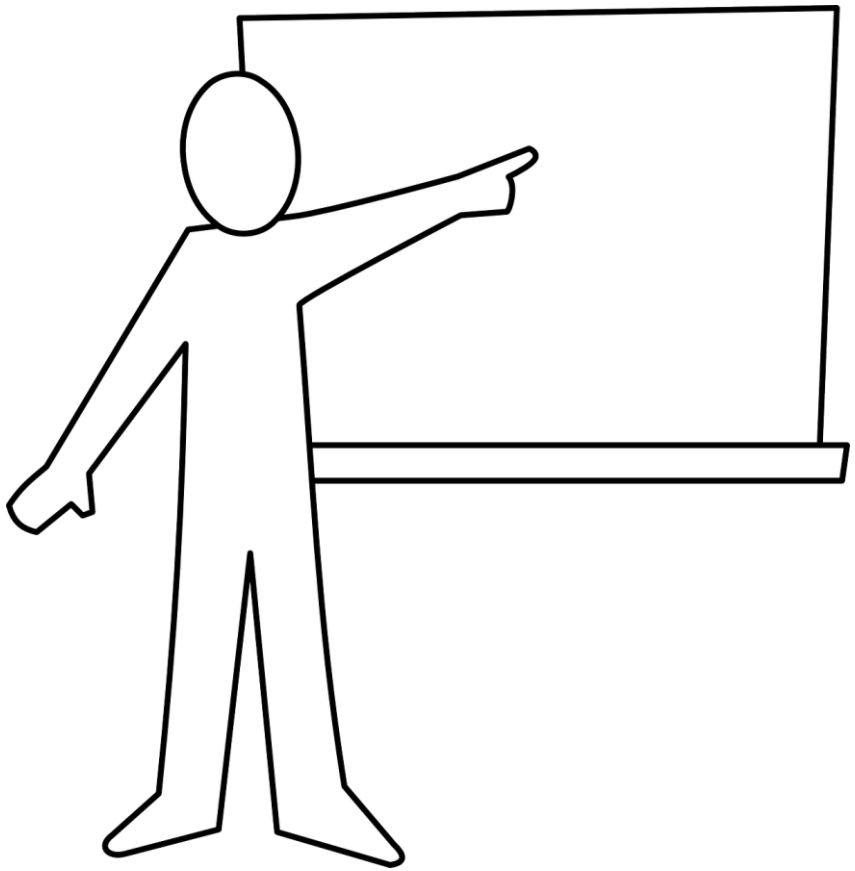
SQL Server <=2017

- Content of the table variable unknown at the compile time => cardinality 1

SQL Server 2019

- Content of the table variable known at the compile time => cardinality = actual number of rows

TABLE VARIABLE DEFERRED COMPILATION

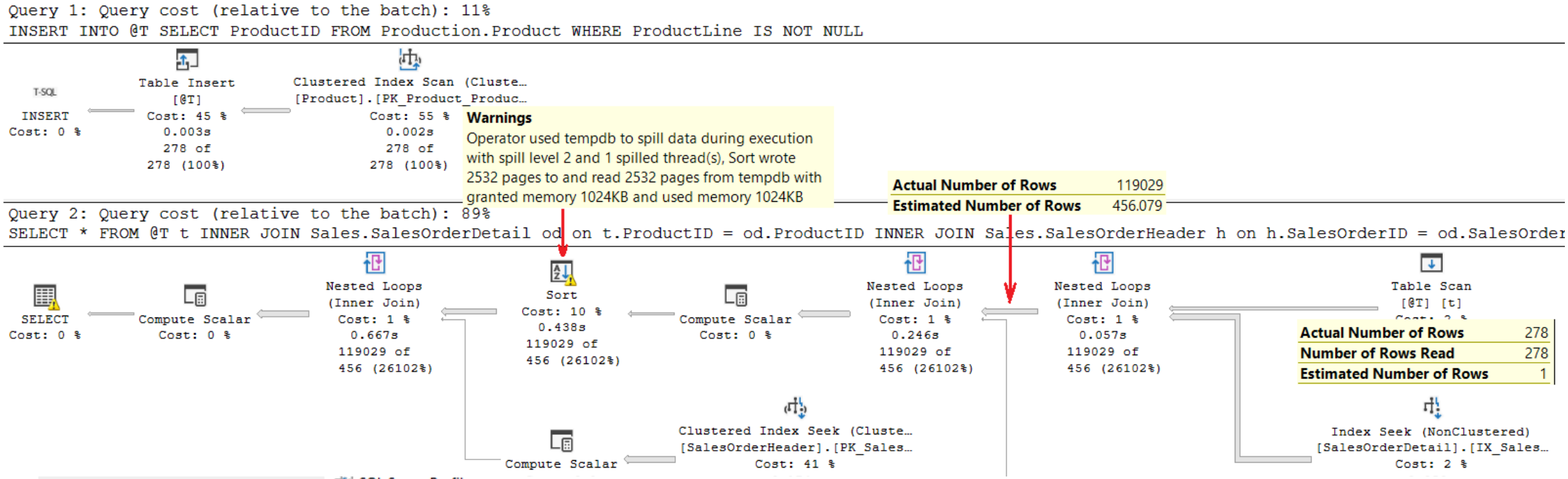


DEMO

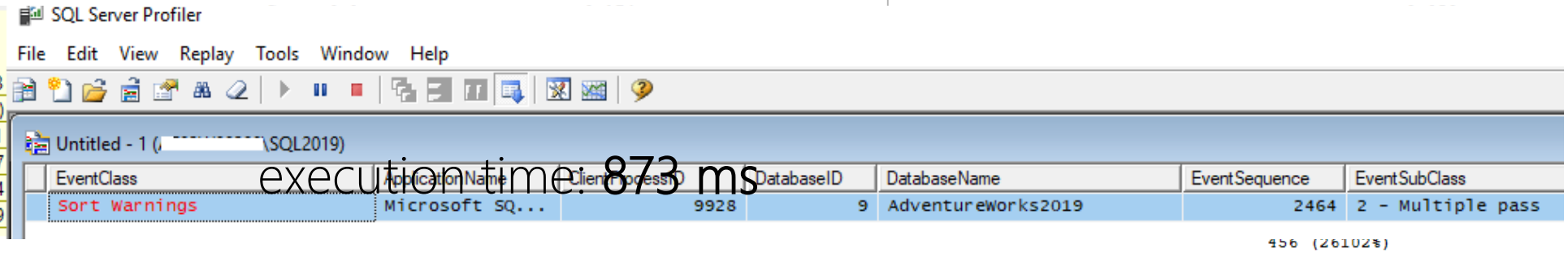
```sql
DECLARE @T AS TABLE (ProductID INT);
INSERT INTO @T SELECT ProductID FROM Production.Product WHERE ProductLine IS NOT NULL;

SELECT * FROM @T t
INNER JOIN Sales.SalesOrderDetail od on t.ProductID = od.ProductID
INNER JOIN Sales.SalesOrderHeader h on h.SalesOrderID = od.SalesOrderID
ORDER BY od.UnitPrice DESC;
```

SQL Server 2017



SELECT	
Cached plan size	96 KB
Estimated Operator Cost	0 (0%)
Degree of Parallelism	1
Estimated Subtree Cost	0,181817
Memory Grant	1024
Estimated Number of Rows	456,079



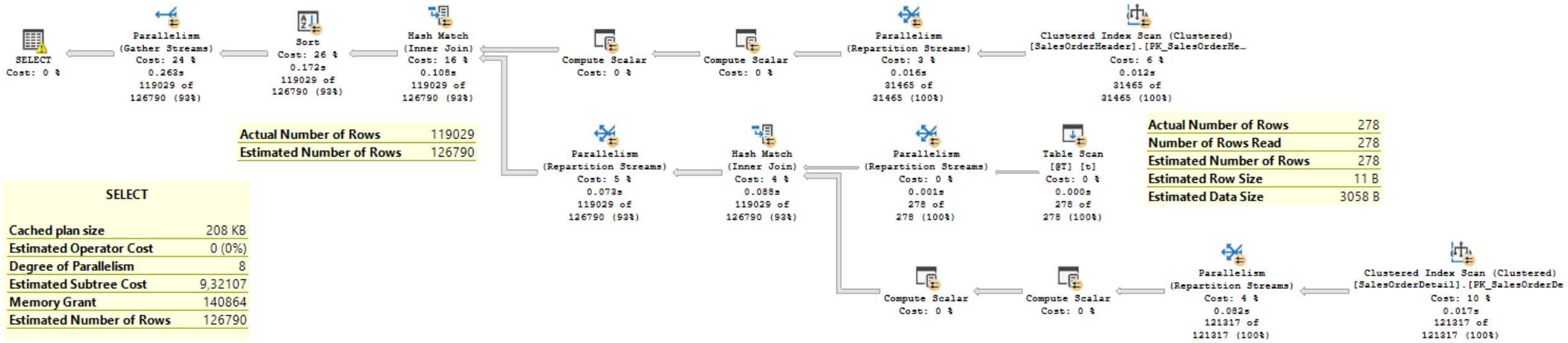
execution time: 873 ms

```sql
DECLARE @T AS TABLE (ProductID INT);
INSERT INTO @T SELECT ProductID FROM Production.Product WHERE ProductLine IS NOT NULL;

SELECT * FROM @T t
INNER JOIN Sales.SalesOrderDetail od on t.ProductID = od.ProductID
INNER JOIN Sales.SalesOrderHeader h on h.SalesOrderID = od.SalesOrderID
ORDER BY od.UnitPrice DESC;
```

SQL Server 2019

```
Query 2: Query cost (relative to the batch): 100%
SELECT * FROM @T t INNER JOIN Sales.SalesOrderDetail od on t.ProductID = od.ProductID INNER JOIN Sales.SalesOrderHeader h on h.SalesOrderID = od.SalesOrderID ORDER BY od.UnitPr…
Missing Index (Impact 18.7112): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [Sales].[SalesOrderDetail] ([ProductID]) INCLUDE ([CarrierTrackingNumber],[Orde…
```



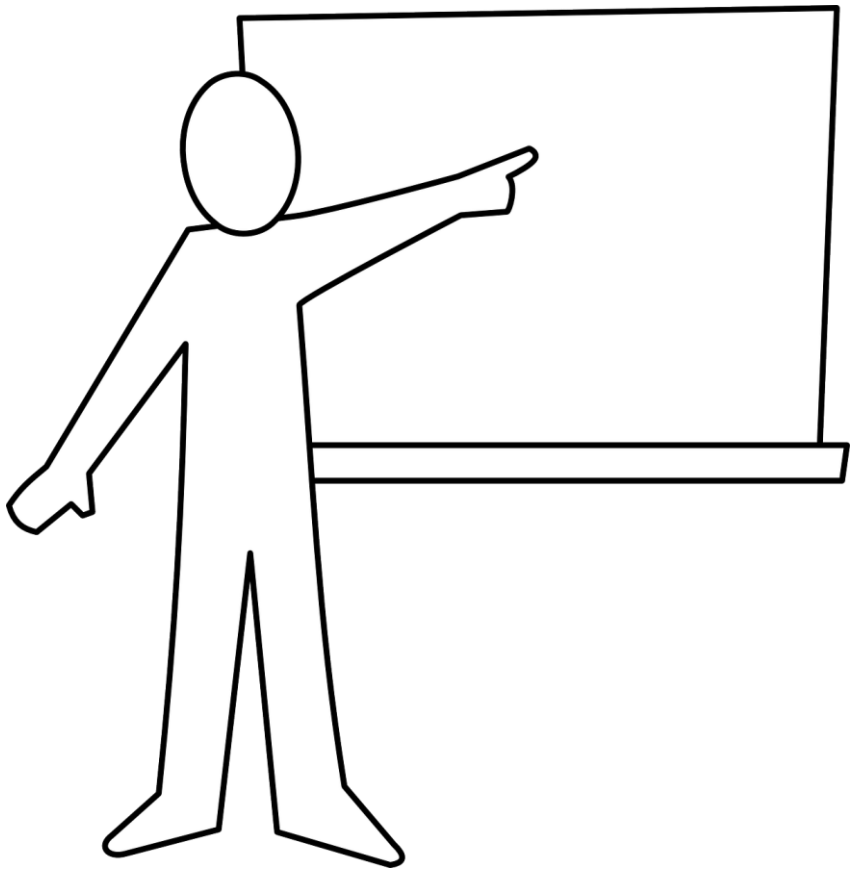
execution time: **484 ms**

2x

TABLE VARIABLE DEFERRED COMPILATION

- Improves plan quality and overall performance for queries referencing table variables
- Cardinality estimates are based on actual table variable row counts
- This accurate row count information will be used for optimizing downstream plan operations

TABLE VARIABLE DEFERRED COMPILATION



DEMO

CONCLUSION

- Designed to address cardinality issues caused by fixed estimation:
  - Nested Loop Joins where Hash Joins are more appropriate
  - Memory grant underestimation issues

- Better estimate for new queries, one can choose the behavior of table variables 

- Prone to Parameter Sniffing 

- Can break existing workarounds