Execution Plans: The Secret to Query Tuning Success

Jes Borland, Senior SQL Engineer

Tech on Tap

http://techontap.org

Love technology and beer? So do we!

Our next event - How Cloudy is Your Organization?
- is Saturday, May 16 at Stone Cellar Brewpub in Appleton

FoxPASS

The SQL Server community for NE WI!

http://fox.sqlpass.org

We meet the first Wednesday of each month at 5:30 pm at Omni Resources (on College Ave near the mall) in Appleton

What is an execution plan?

```
SELECT SOH.SalesOrderID,
SOH.OrderDate, PROD.Name,
SOD.OrderQty
FROM Sales.SalesOrderHeader SOH
    INNER JOIN Sales.SalesOrderDetail
SOD ON SOD.SalesOrderID =
SOH.SalesOrderID
    INNER JOIN Production.Product PROD
ON PROD.ProductID = SOD.ProductID
ORDER BY SOH.SalesOrderID
```

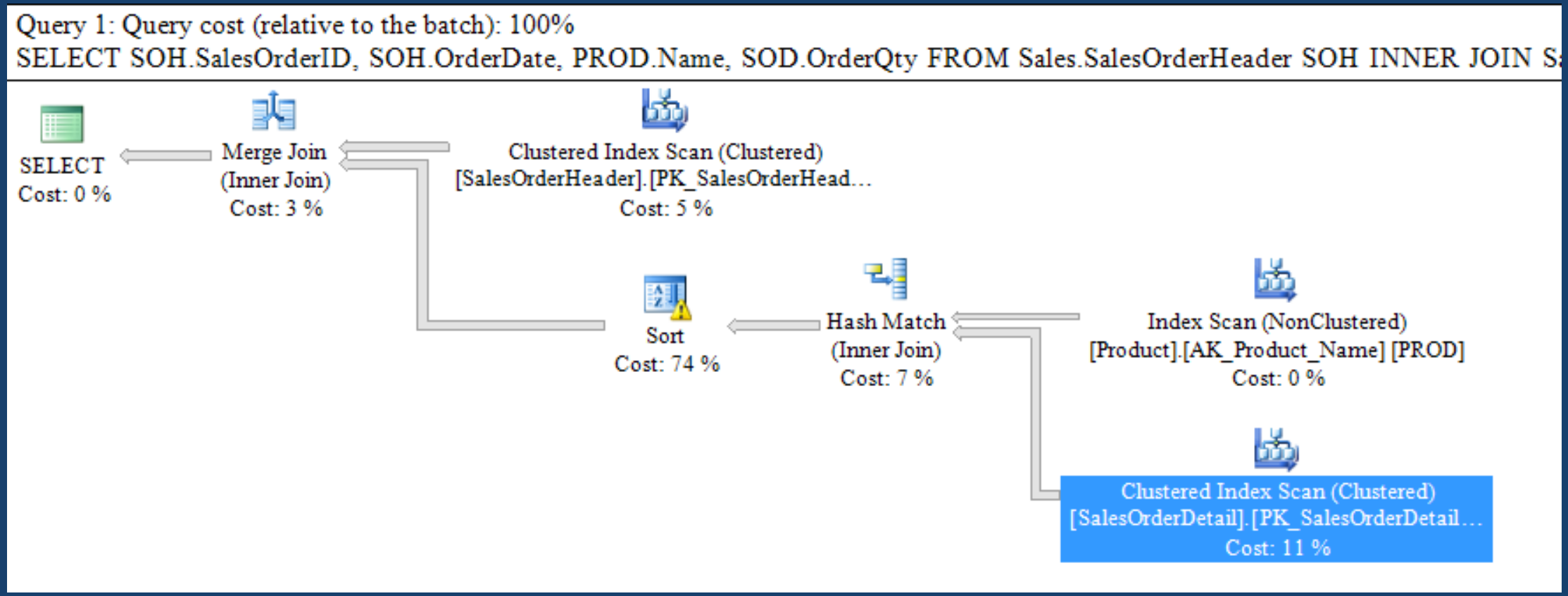
Ingredients

- 1 SalesOrderHeader Clustered Index Scan (PK_SalesOrderHeader_SalesOrderID)
- 1 SalesOrderDetail Clustered Index Scan (PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID)
- 1 Product Nonclustered Index Scan (AK_Product_Name)
- 1 Hash Match
- 1 Merge
- 1 Sort
- 1 SELECT

Directions

Scan the SalesOrderHeader clustered index. Scan the Product non-clustered index. Scan the SalesOrderDetail non-clustered index. Join the output of Product and SalesOrderDetail with a Hash Match. Sort the results. Join the output of SalesOrderheader with the Hash Match output using a Merge.

Return the correct results to the SELECT operator.

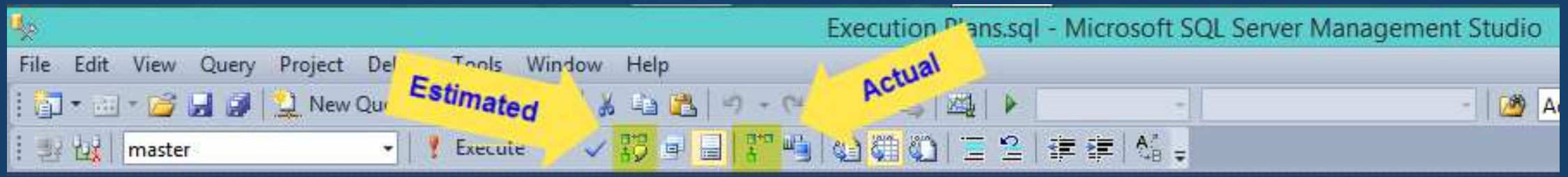


Execution plans are generated when a query is first executed

- Generally, they get placed in memory for re-use
- The cache stores information such as CPU, memory, and I/O as well

How you can view execution plans

- SQL Server Management Studio (SSMS)
  - For a currently executing query
    - SHOWPLAN XML – Estimated execution plan
    - Include Estimated execution plan
    - Include Actual Execution Plan
    - Use Extended Events to capture
  - For a previously-executed query
    - Plan cache



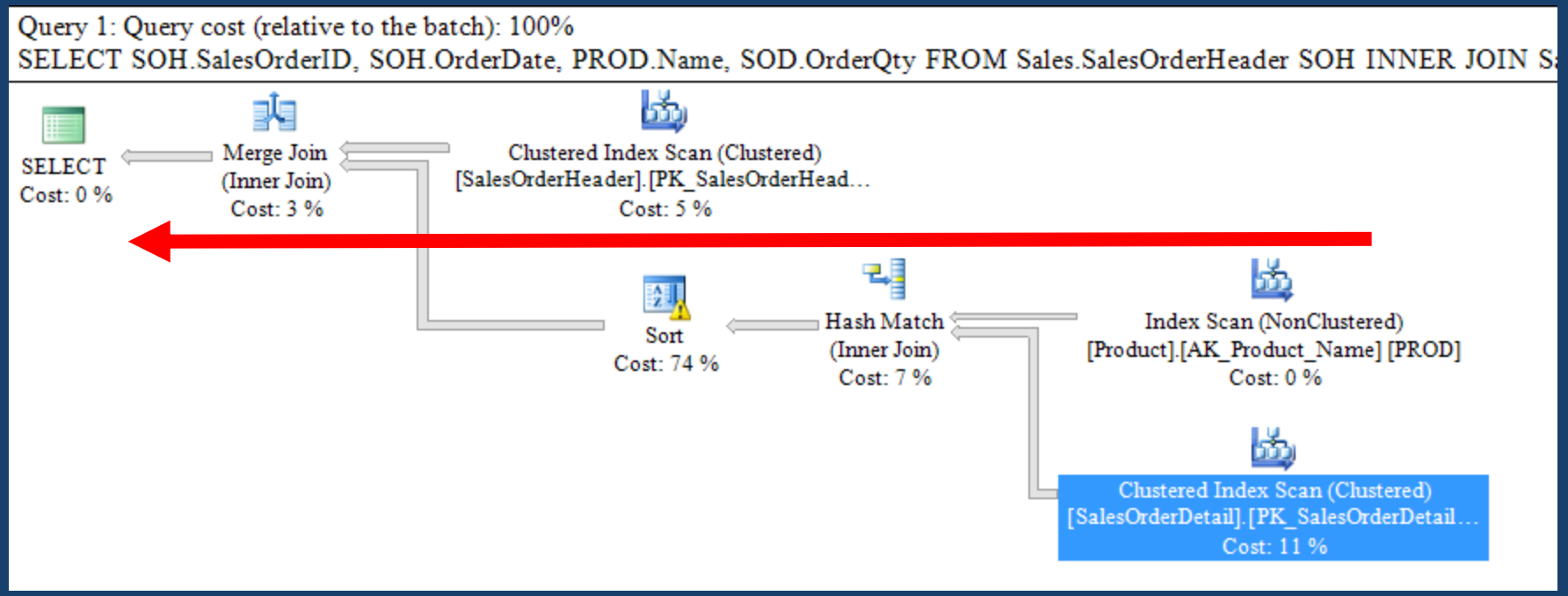
Proper permissions required

- Permission to execute the query
- SHOWPLAN permission granted in the database
  - **http://tinyurl.com/nn3j8mk**

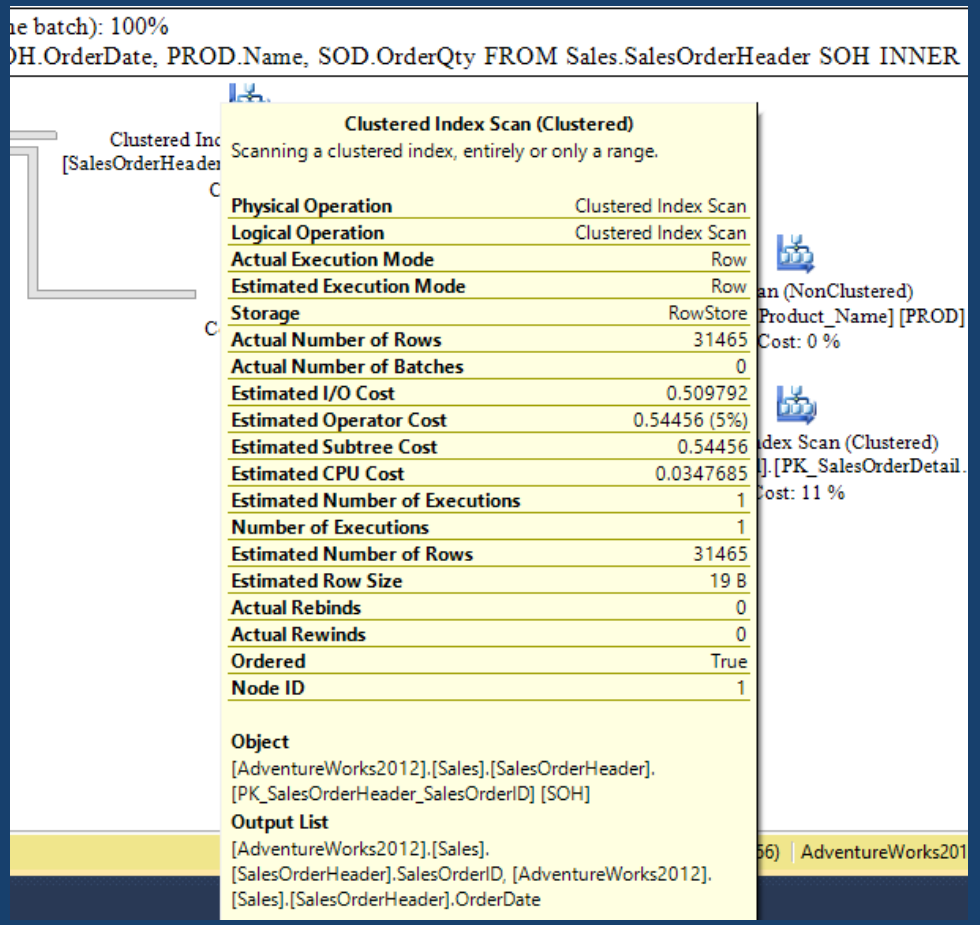
How we read execution plans

- Right to left
- Hover over operators for more information
- Use Properties for even more information
- Final operator gives us some extra goodies
- Lines show us how many rows were returned

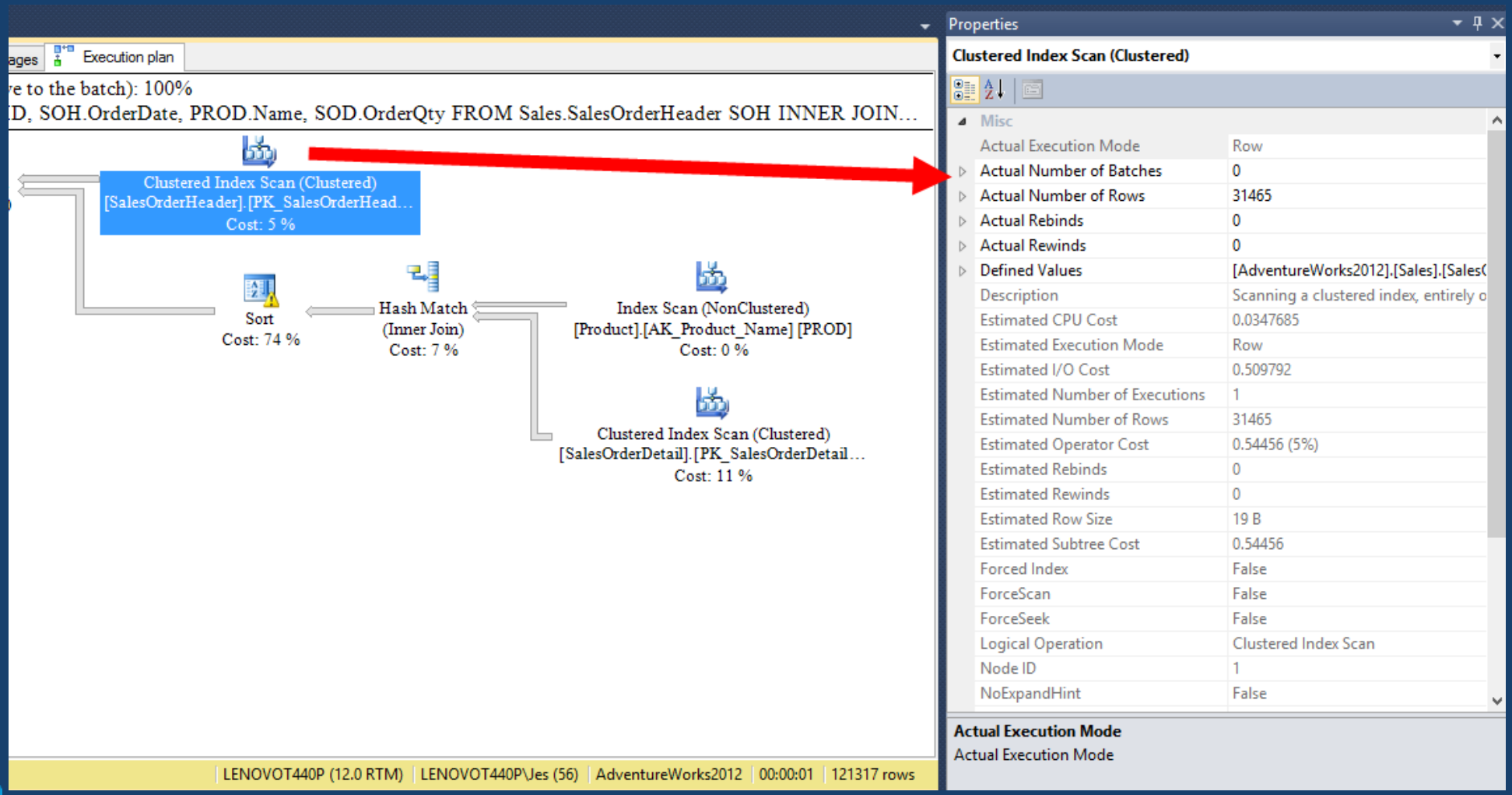
Right to left



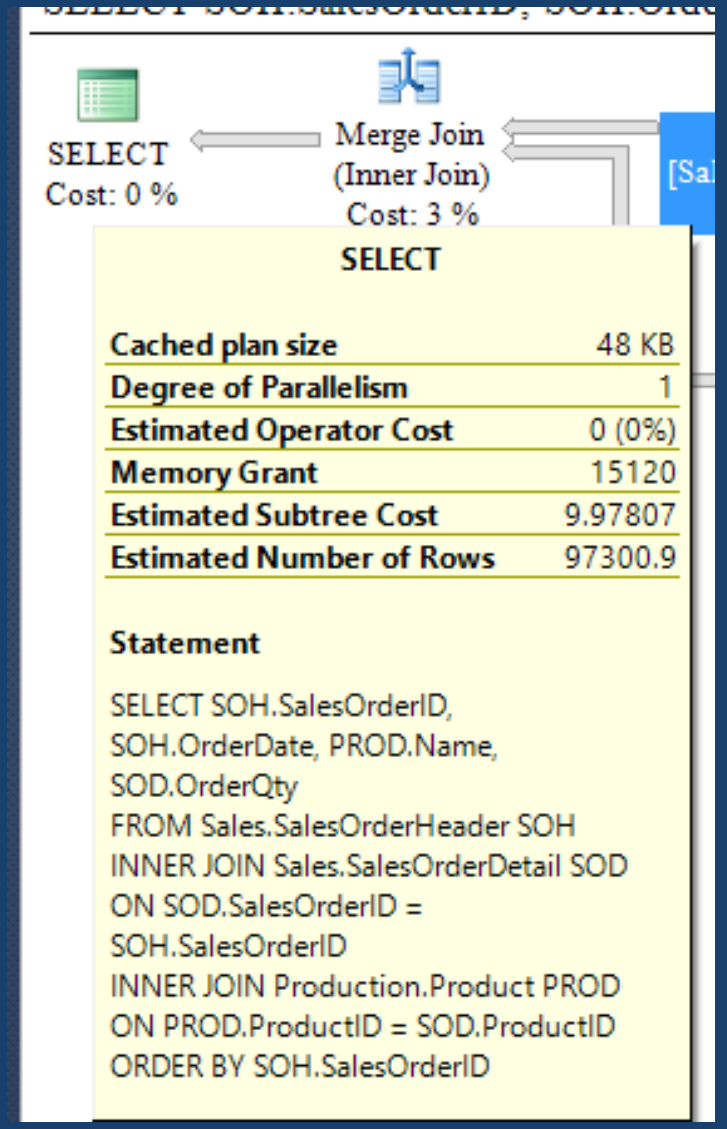
Hover over operators



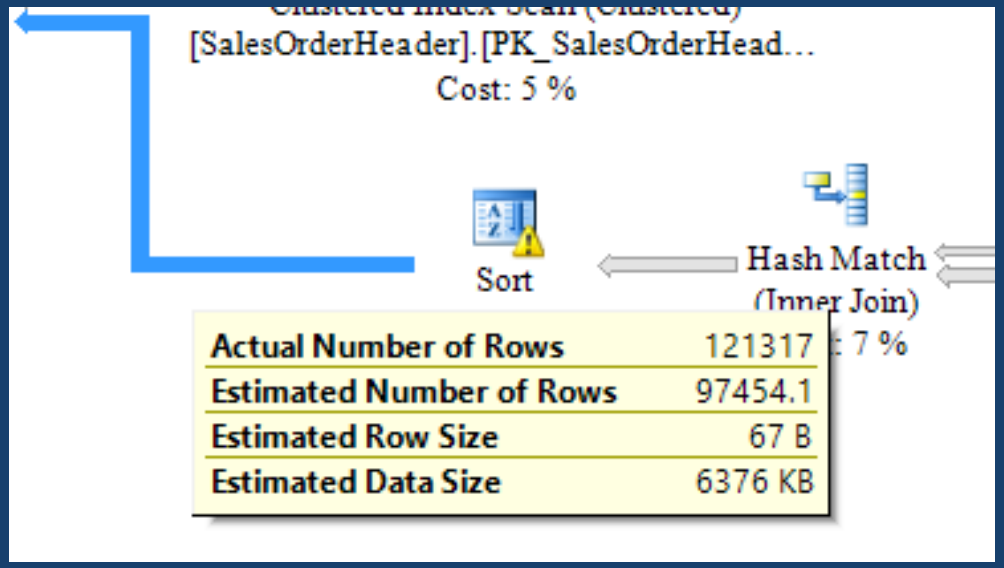
Use Properties



Final operator



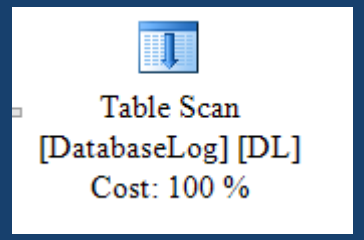
Read ~~between~~ the lines

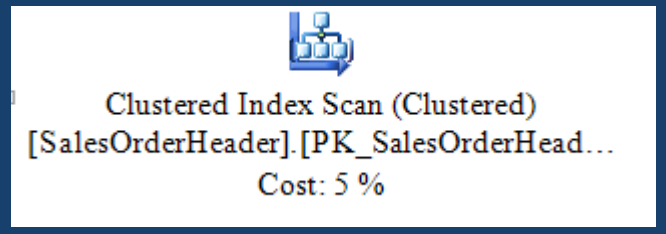


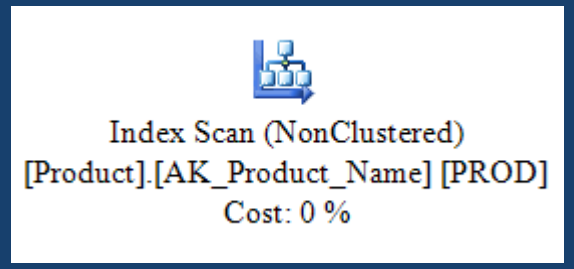
Let’s take a look

Common operators

The Scan

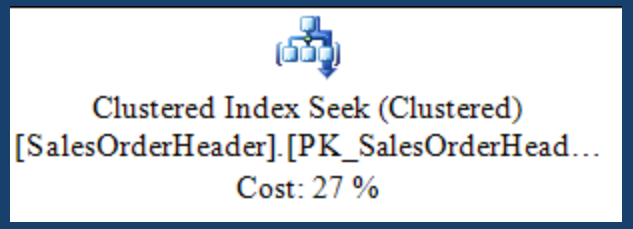


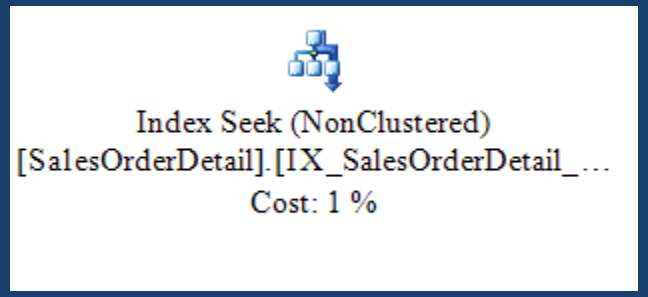




- Reading all of the rows in a table or index
- If the query has a WHERE clause against that table or index, only the rows matching the predicate are *returned*
- *Can be* expensive

The Seek

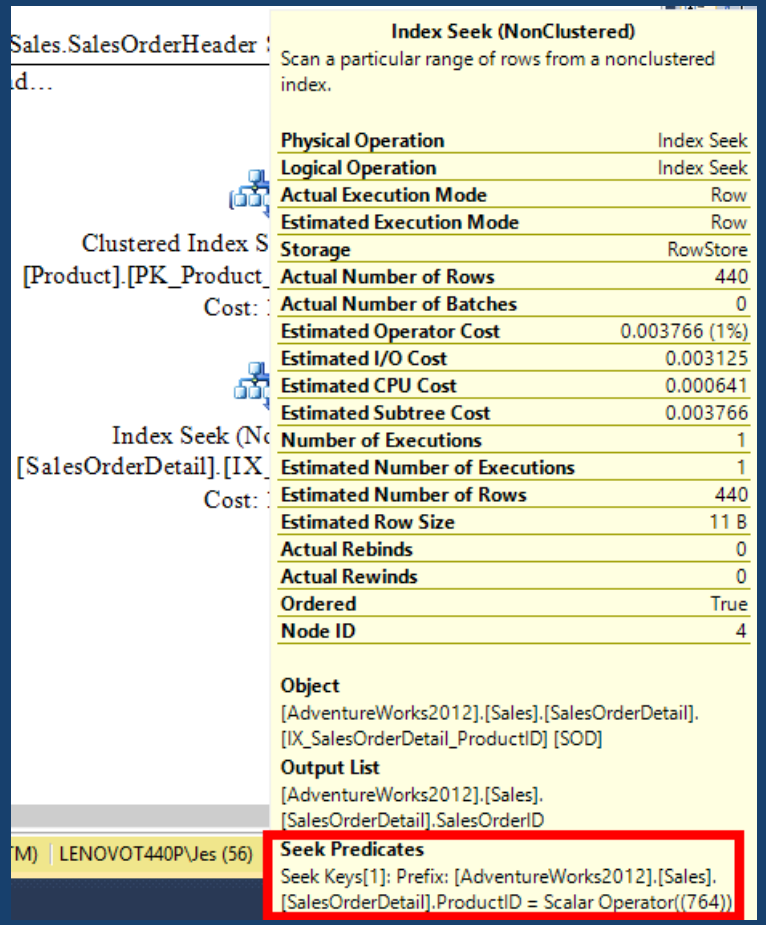




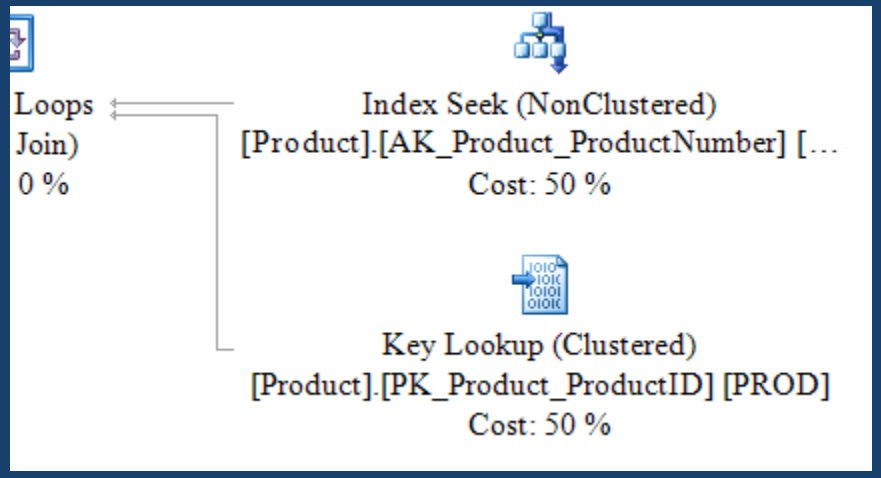
- Reads only a portion of the clustered or non-clustered index
- This is determined by the "seek predicate"

Seek predicates

```
SELECT SOH.SalesOrderID,
SOH.OrderDate, PROD.Name
FROM Sales.SalesOrderHeader SOH
INNER JOIN Sales.SalesOrderDetail
SOD ON SOD.SalesOrderID =
SOH.SalesOrderID
INNER JOIN Production.Product PROD
ON PROD.ProductID = SOD.ProductID
WHERE SOD.ProductID = 764
```



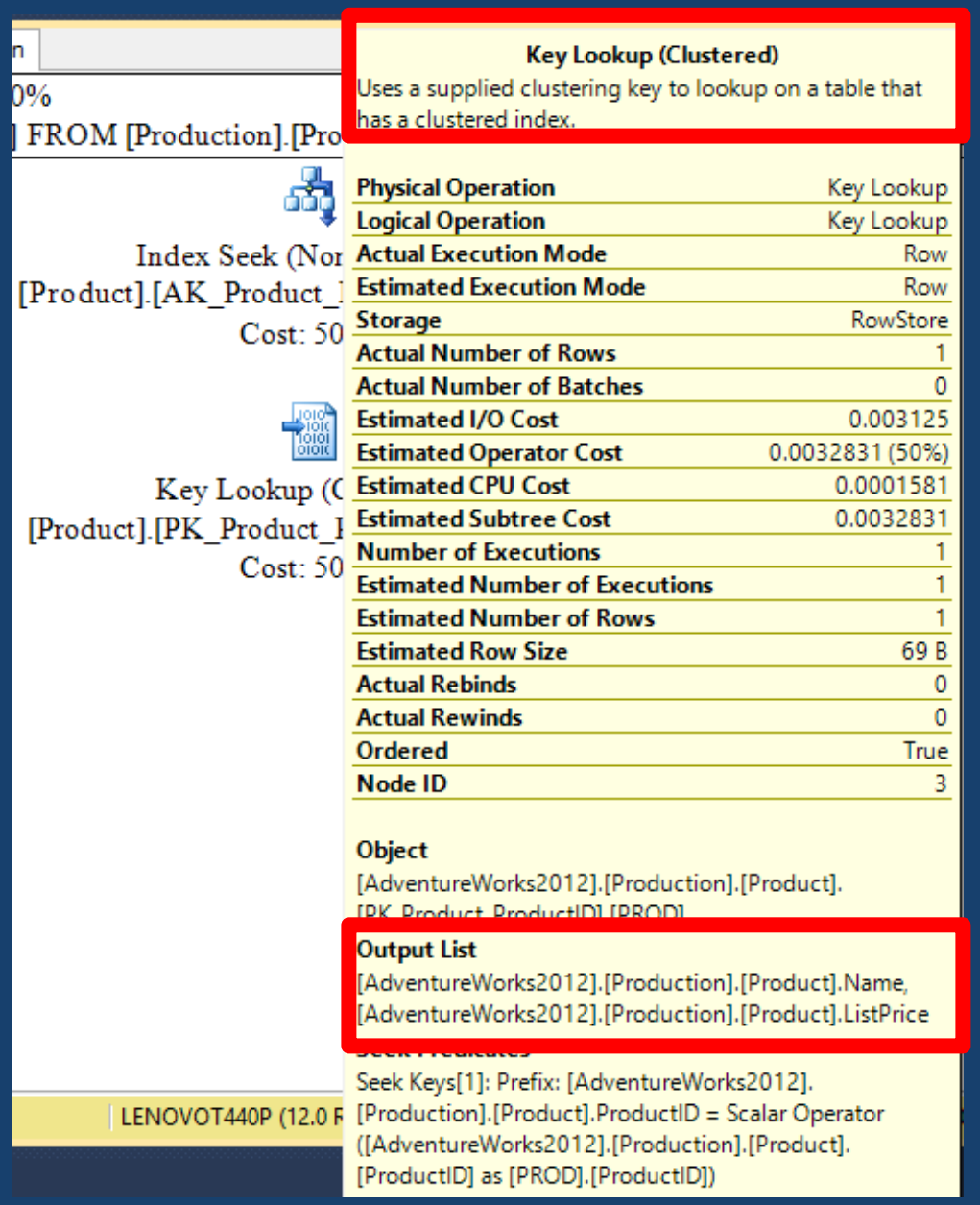
The Lookup



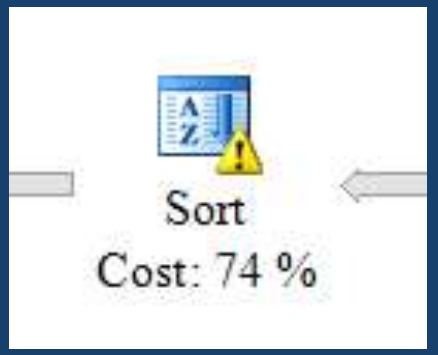
- The WHERE clause can be evaluated using a non-clustered index
- Other columns returned in other statements (SELECT) are not in the non-clustered index structure
- SQL Server uses the clustered index key stored on the non-clustered index page to get the other columns from the clustered index

Lookup

```
SELECT PROD.Name, PROD.ListPrice
FROM Production.Product PROD
WHERE PROD.ProductNumber='FR-R92R-44'
```



The Sort



- Data needs to be sorted due to an ORDER BY or GROUP BY
- Typically very expensive!

Joins

The Logical Join

- INNER JOIN
- OUTER JOIN
- LEFT JOIN
- RIGHT JOIN

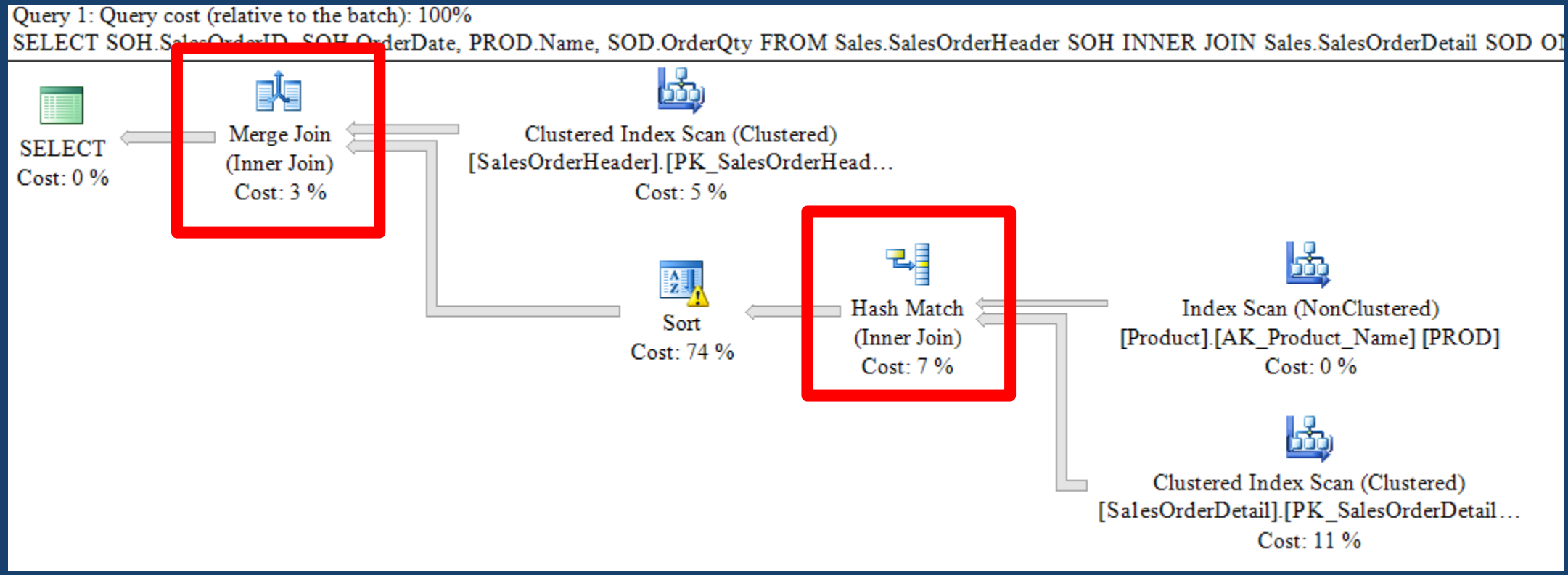
INNER JOIN

```
SELECT SOH.SalesOrderID, SOH.OrderDate,
PROD.Name, SOD.OrderQty
FROM Sales.SalesOrderHeader SOH
INNER JOIN Sales.SalesOrderDetail SOD ON
    SOD.SalesOrderID = SOH.SalesOrderID
INNER JOIN Production.Product PROD ON
PROD.ProductID   = SOD.ProductID
ORDER BY SOH.SalesOrderID
```

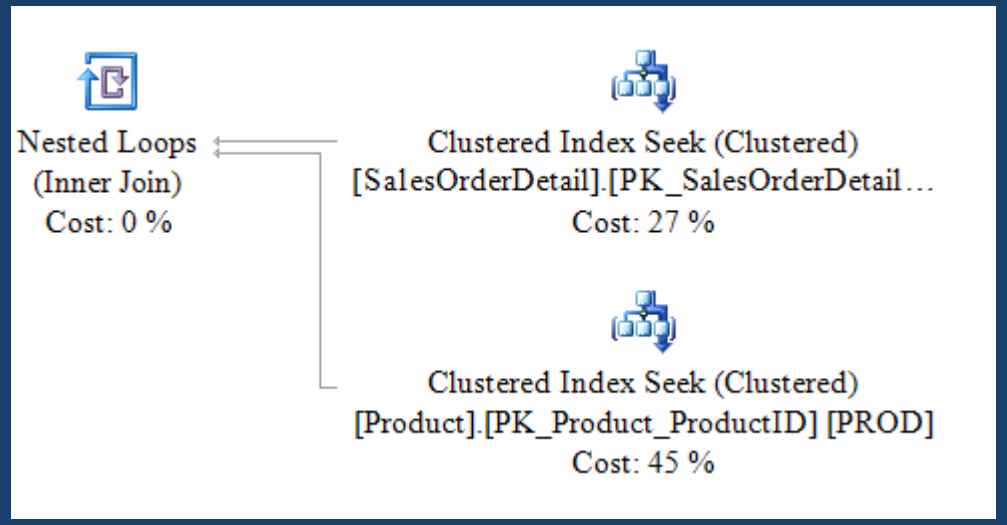
The Physical Join

- Nested loops
- Merge
- Hash

This is the execution plan of the INNER JOIN query



Nested Loops

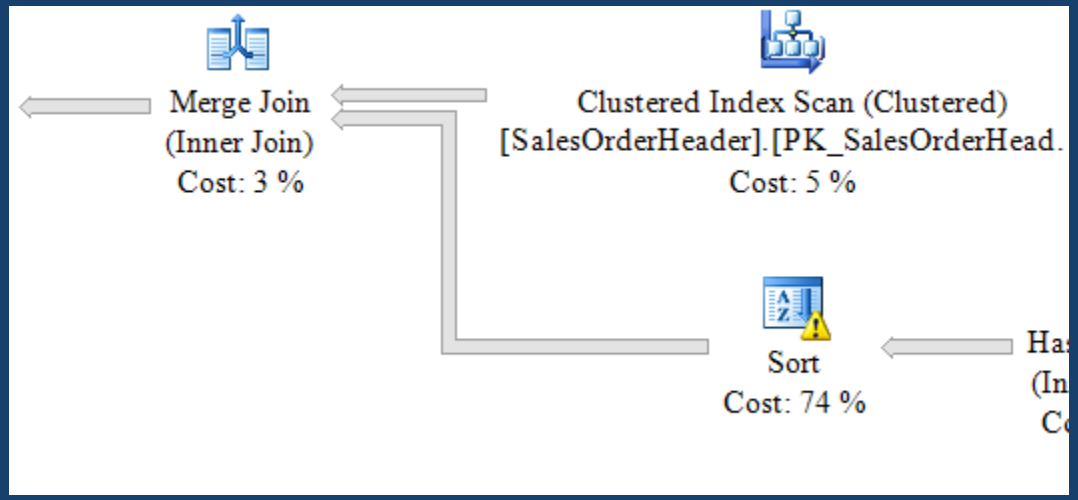


- Best for small data sets
- Outer (top) and inner (bottom) inputs
- Compare every row in inner loop to outer loop

Photo: Aka Hige https://www.flickr.com/photos/akahige/1044518492/in/photolist-2AiqXw-8peEYs-gsUJq-53AgoC-8t1YnK-28zraz-bWzyjD-bWzuYH-9MkCBU-9MhP5a-9MkCd7-oDUaRR-oWjgSF-oDQ246-oEifhQ-oWL6a3-oWL5xb-c7UTKL-2tGgh-oDPa24-oUuvGC-oWotJi-oW7NLe-oW7N2D-pNzVr2-p9gsCk-q5QCex-q69h1G-p9dL6o-pNBVMs-q698ud-pNzGsg-q5Qobx-q5ZYx4
-pNzBoP-oWmA3m-oDQ3sZ-oE2itv-oDPWbS-oDPRQC-oUhfCd-oW3twR-oEjmrt-oEj98b-oULtAm-oWwPHB-oEiFSY-oWLxA9-oEiE35-oEiiMK/

sunshinecity https://www.flickr.com/photos/sunshinecity/455880294/in/photolist-Ghvk5-8hWCYQ-siZBM-5LexoC-3bQ9o-qzXsi-w6vki-5LevK5-5Lagb2-75F9J-5Lagwv-5Lestu-5LevpN-8xYKsp-5LabCT-bzC3Dx-6j3DXv-6fJnht-8xYKL6-8y2NoW-5WVFpH-51CWGy-6vm3Mg-5Lerd5-55py8R-6gV2i5-5LerZU-6PmiNj-vr6uX-czhtWQ-7r91yj-5V59yU-siZBN-4RzTQj-bijRqT-dJ85Xp-6iSfjp-zBuHk-aEvew1-akQJb2-6iWpy7-5LewWj-5Lev3j-5LaeKn-5Leuju-5LetQS-5LetuU-5Lad9x-5LesLC-9rbGRt

Merge

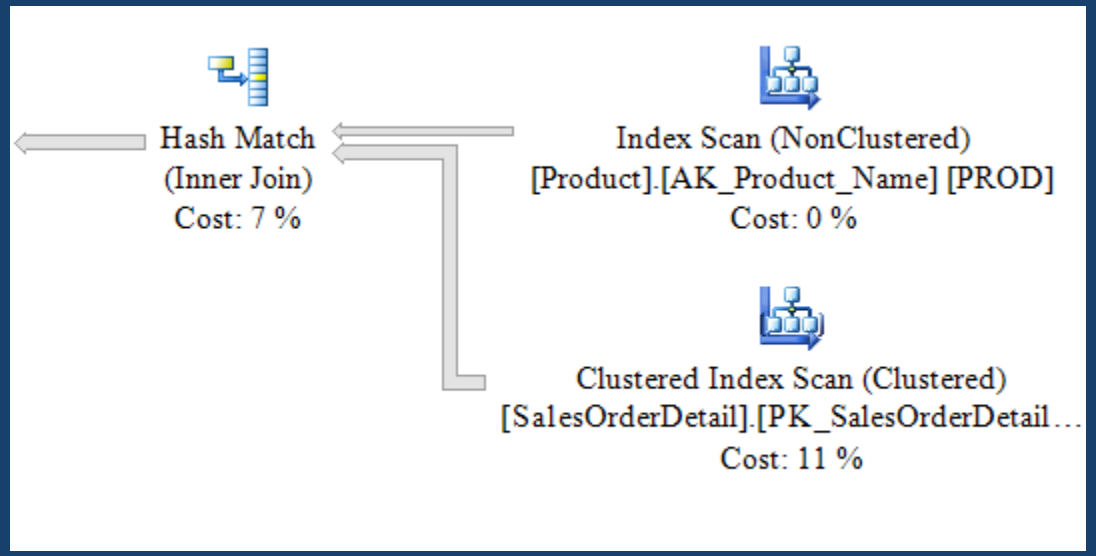


- Both inputs must be sorted in the same order
- Each row is compared

Photos: Becka Spence https://www.flickr.com/photos/nevrlndtink/1920763302/in/photolist-7JHijY-6qbipm-6qbid7-6qbhQw-6qbhAd-6q798t-3VJq1E-85i1X5-nGdiLA-iTgJrA-iTeFbc-pk979P-caMKg5-bYXuxd-bpJYsn-5LYb4h-aJZj32-c93xaQ-c93wXj-du1XBt-q3HPiz-2Air87-2Air49-2AiqXw-8peEYs-gsUJq-53AgoC-8t1YnK-28zraz-bWzyjD-bWzuYH-9MkCBU-9MhP5a-9MkCd7-asMJZp-oWjePn-oWjfv2-oDUaRR-oWjgSF-oDQ246-oEifhQ-oWL6a3-oWL5xb-c7UTKL-2tGgh-oDPa24-oUuvGC-oWotJi-oW7NLe-oW7N2D

Micky** https://www.flickr.com/photos/emzee/278221145/in/photolist-qzXsi-w6vki-5LevK5-5Lagb2-75F9J-5Lagwv-5Lestu-5LevpN-8xYKsp-5LabCT-bzC3Dx-6j3DXv-6fJnht-8xYKL6-8y2NoW-5WVFpH-51CWGy-6vm3Mg-5Lerd5-55py8R-6gV2i5-5LerZU-6PmiNj-vr6uX-czhtWQ-7r91yj-5V59yU-siZBN-4RzTQj-bijRqT-dJ85Xp-6iSfjp-zBuHk-aEvew1-akQJb2-6iWpy7-5LewWj-5Lev3j-5LaeKn-5Leuju-5LetQS-5LetuU-5Lad9x-5LesLC-9rbGRt-6KAFkY-p3gag6-6wvjpw-7oDn6M-5czFkp

Hash



- Best for large data sets
- Hash tables are built in memory – build and probe
- Values from probe compared to build

Photos: Justin Ennis https://www.flickr.com/photos/averain/4039312969/in/photolist-79Wx24-6eL9Qd-tF4Y-5zPpQL-bYXxwY-au4JnJ-2Ae4vR-qZmJ9g-8derqL-qZvKzt-bUUh5i-7JHijY-6qbipm-6qbid7-6qbhQw-6qbhAd-6q798t-3VJq1E-85i1X5-nGdiLA-iTgJrA-iTeFbc-pk979P-caMKg5-bYXuxd-bpJYsn-5LYb4h-aJZj32-c93xaQ-c93wXj-du1XBt-q3HPiz-2Air87-2Air49-2AiqXw-8peEYs-gsUJq-53AgoC-8t1YnK-28zraz-bWzyjD-bWzuYH-9MkCBU-9MhP5a-9MkCd7-asMJZp-oWjePn-oWjfv2-oDUaRR-oWjgSF

HISHAM BINSUWAIF https://www.flickr.com/photos/4444/340569115/in/photolist-w6vki-9tVrew-5LevK5-5Lagb2-75F9J-5Lagwv-5Lestu-5LevpN-8xYKsp-5LabCT-bzC3Dx-6j3DXv-6fJnht-8xYKL6-8y2NoW-5WVFpH-51CWGy-6vm3Mg-5Lerd5-55py8R-6gV2i5-5LerZU-6PmiNj-vr6uX-czhtWQ-7r91yj-5V59yU-siZBN-4RzTQj-bijRqT-7kyAtD-dJ85Xp-6iSfjp-zBuHk-aEvew1-akQJb2-6iWpy7-5LewWj-5Lev3j-5LaeKn-5Leuju-5LetQS-5LetuU-5Lad9x-5LesLC-9rbGRt-6KAFkY-p3gag6-6wvjpw-7oDn6M

Execution plans are trying to help you!

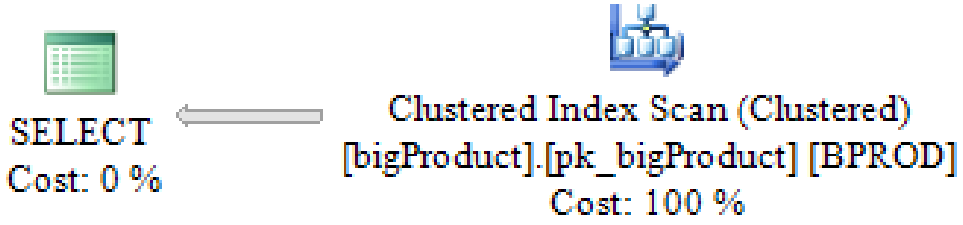
Missing indexes

```
SELECT Name, Color, Size, Weight
FROM bigProduct BPROD
WHERE Class = 'L'
```

Query 1: Query cost (relative to the batch): 100%

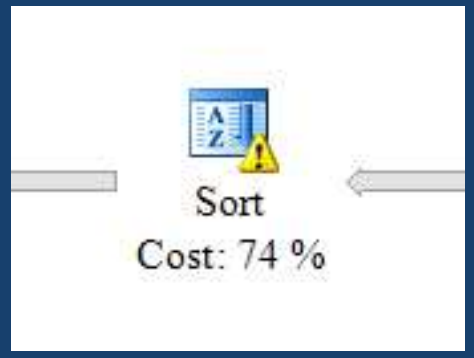
SELECT [Name],[Color],[Size],[Weight] FROM [bigProduct] [BPROD] WHERE [Class]=@1

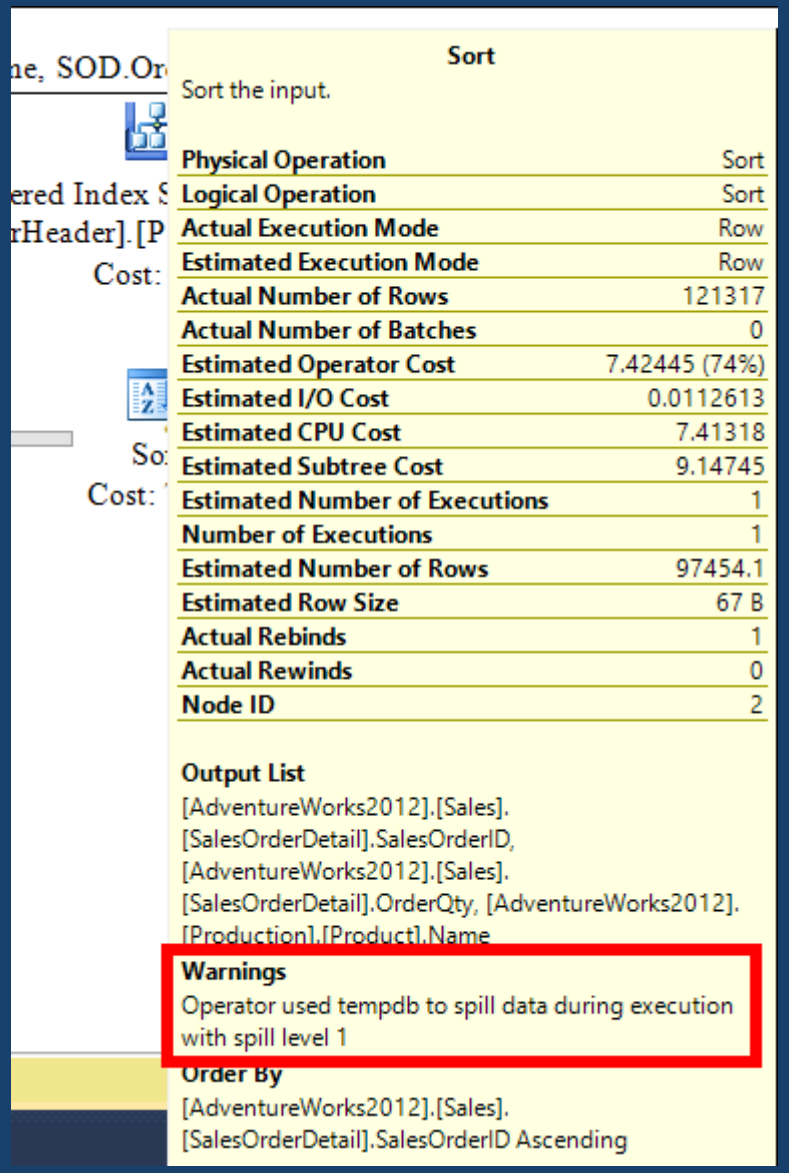
Missing Index (Impact 81.8037): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] (




```
/*
Missing Index Details from Execution Plans.sql
The Query Processor estimates that implementing the following index could improve the query
cost by 81.8037%.
*/

/*
USE [AdventureWorks2012]
GO
CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>]
ON [dbo].[bigProduct] ([Class])
INCLUDE ([Name],[Color],[Size],[Weight])
GO
*/
```

Warnings





Missing Index DMVs

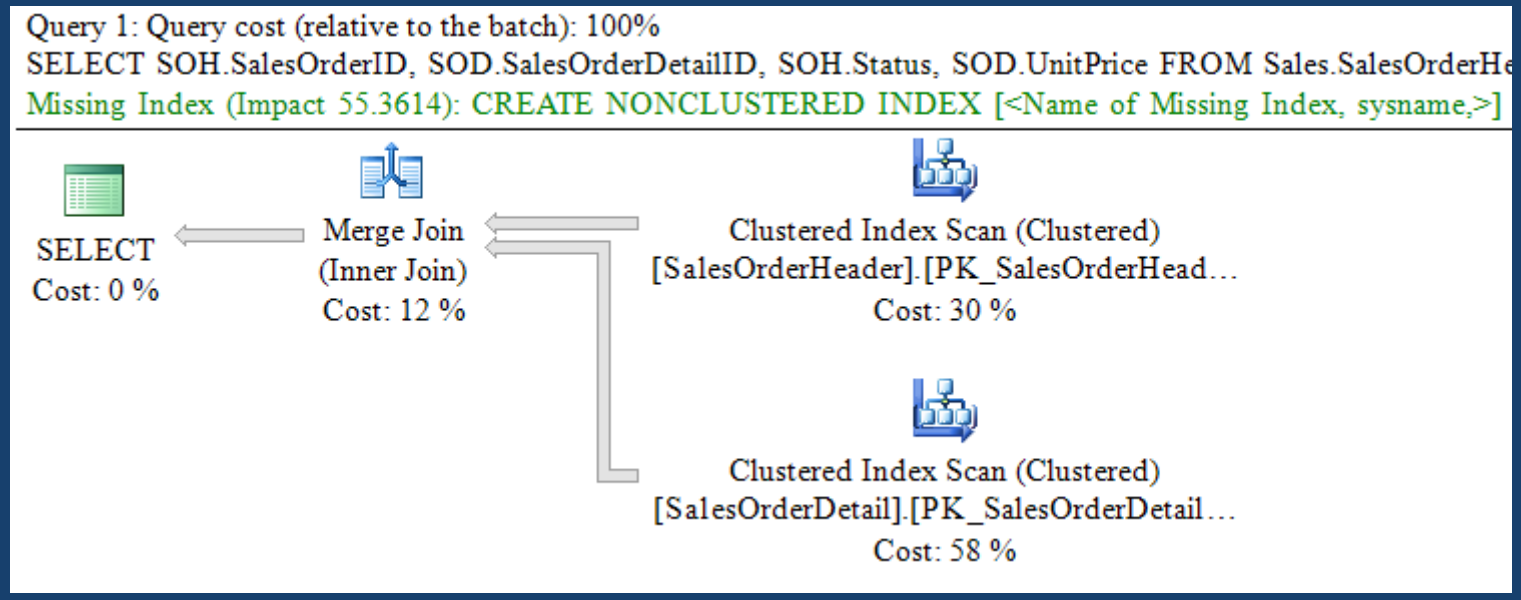
- There are four missing index DMVs
- Combine them to find missing index requests & impact
- http://blog.sqlauthority.com/2011/01/03/sql-server-2008-missing-index-script-download/

What the execution plan *won't* tell you

Multiple missing indexes

A query with two predicates

```
SELECT SOH.SalesOrderID,
SOD.SalesOrderDetailID, SOH.Status,
SOD.UnitPrice
FROM Sales.SalesOrderHeader SOH
INNER JOIN Sales.SalesOrderDetail SOD ON
SOD.SalesOrderID = SOH.SalesOrderID
WHERE SOH.Status = 5
    AND SOD.UnitPrice > 500
```



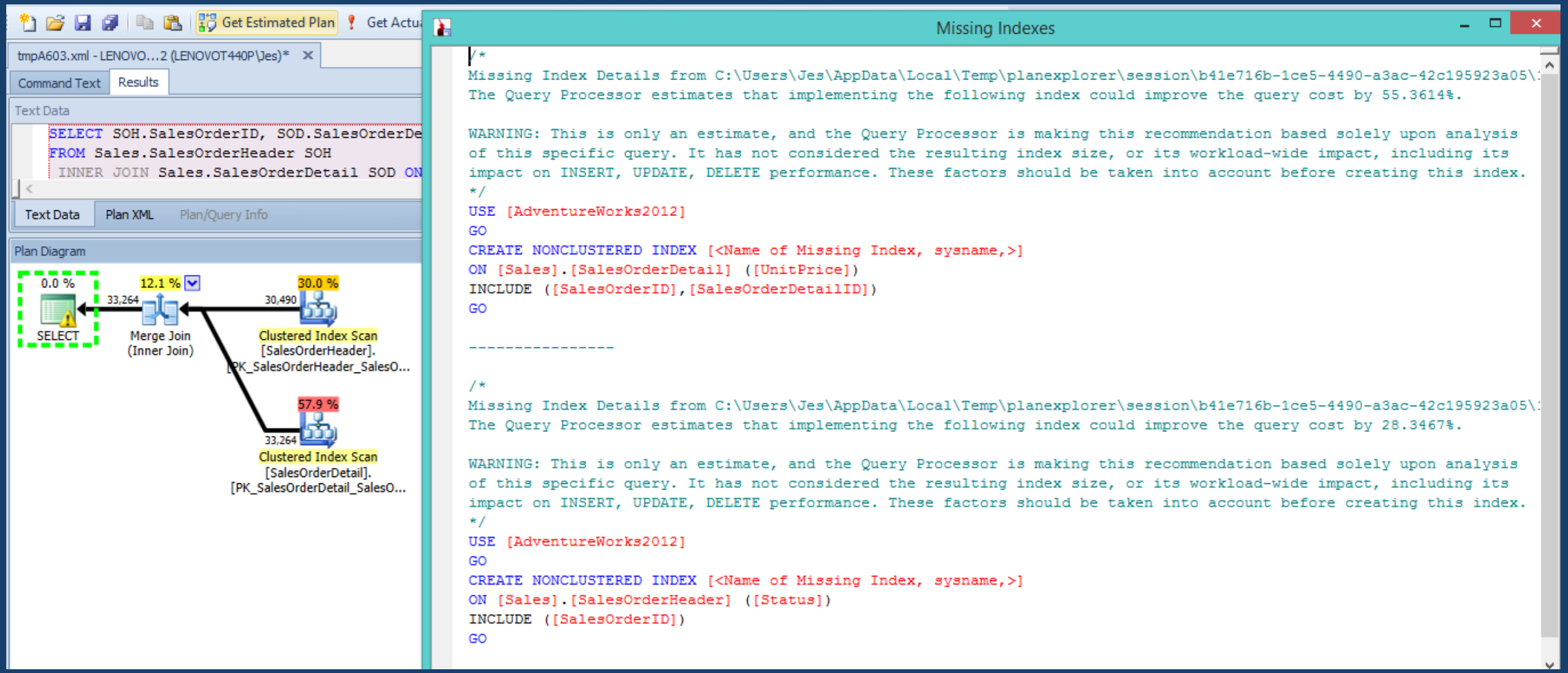

```
/*
Missing Index Details from Execution Plans.sql
The Query Processor estimates that implementing the following index could improve the query cost by
55.3614%.
*/

/*
USE [AdventureWorks2012]
GO
CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>]
ON [Sales].[SalesOrderDetail] ([UnitPrice])
INCLUDE ([SalesOrderID],[SalesOrderDetailID])
GO
*/
```

Dig into the XML…

```
 9            <MissingIndexes>
10              <MissingIndexGroup Impact="55.3614">
11                <MissingIndex Database="[AdventureWorks2012]" Schema="[Sales]"
    Table="[SalesOrderDetail]">
12                  <ColumnGroup Usage="INEQUALITY">
13                    <Column Name="[UnitPrice]" ColumnId="7" />
14                  </ColumnGroup>
15                  <ColumnGroup Usage="INCLUDE">
16                    <Column Name="[SalesOrderID]" ColumnId="1" />
17                    <Column Name="[SalesOrderDetailID]" ColumnId="2" />
18                  </ColumnGroup>
19                </MissingIndex>
20              </MissingIndexGroup>
21              <MissingIndexGroup Impact="28.3467">
22                <MissingIndex Database="[AdventureWorks2012]" Schema="[Sales]"
    Table="[SalesOrderHeader]">
23                  <ColumnGroup Usage="EQUALITY">
24                    <Column Name="[Status]" ColumnId="6" />
25                  </ColumnGroup>
26                  <ColumnGroup Usage="INCLUDE">
27                    <Column Name="[SalesOrderID]" ColumnId="1" />
28                  </ColumnGroup>
29                </MissingIndex>
30              </MissingIndexGroup>
31            </MissingIndexes>
```

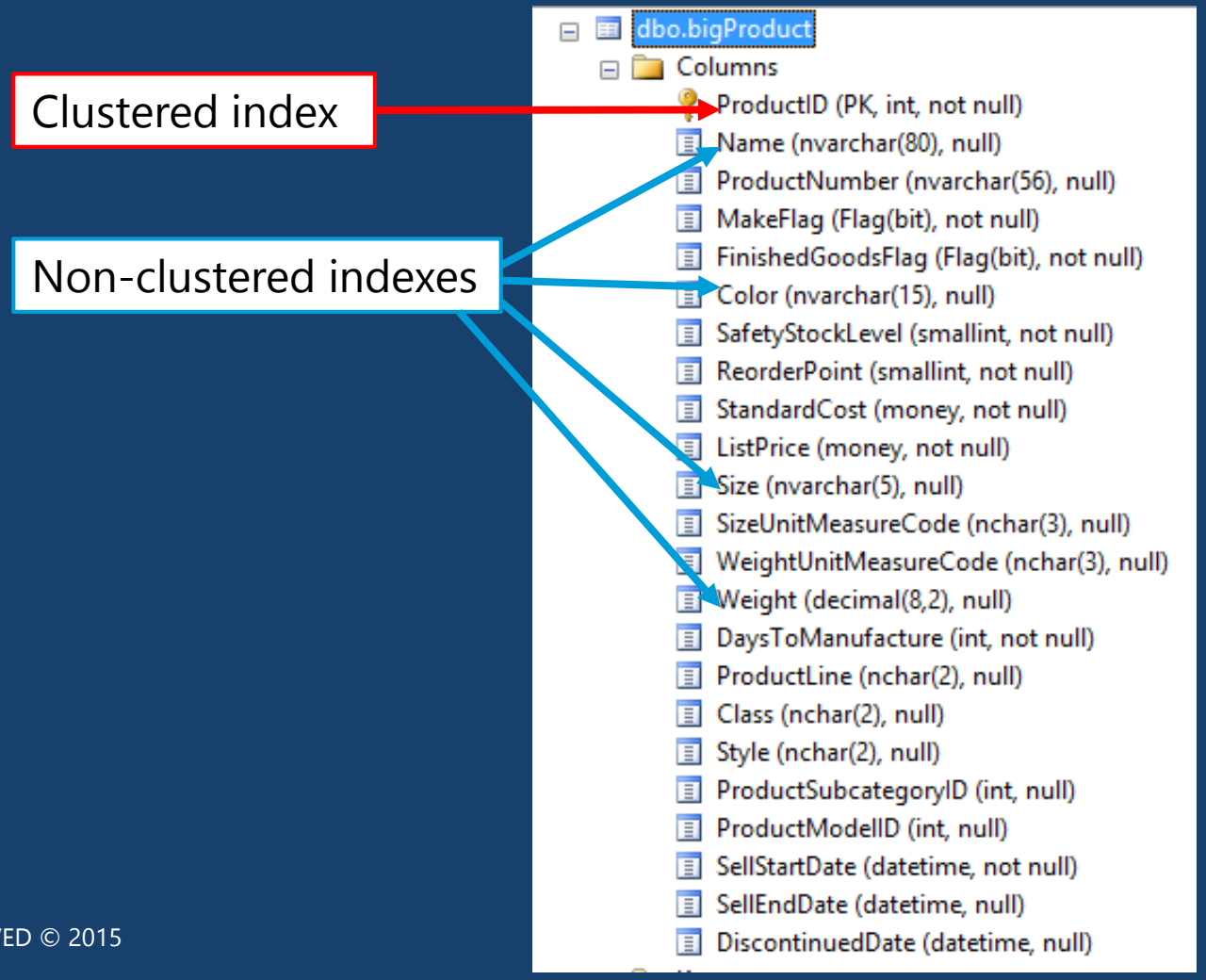
Solutions

- Use the missing index DMVs to find all missing indexes
- Review the XML
- Use a (free!) tool called SQL Sentry Plan Explorer to view the execution plan



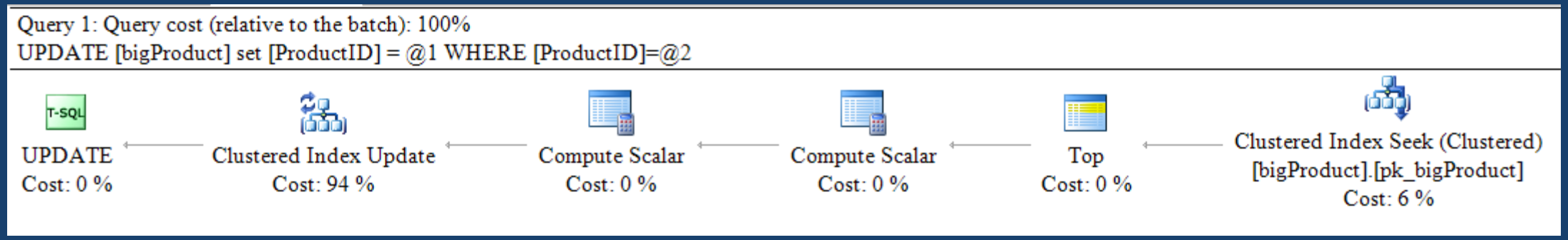
How many indexes are affected?

INSERTS, UPDATES, DELETES

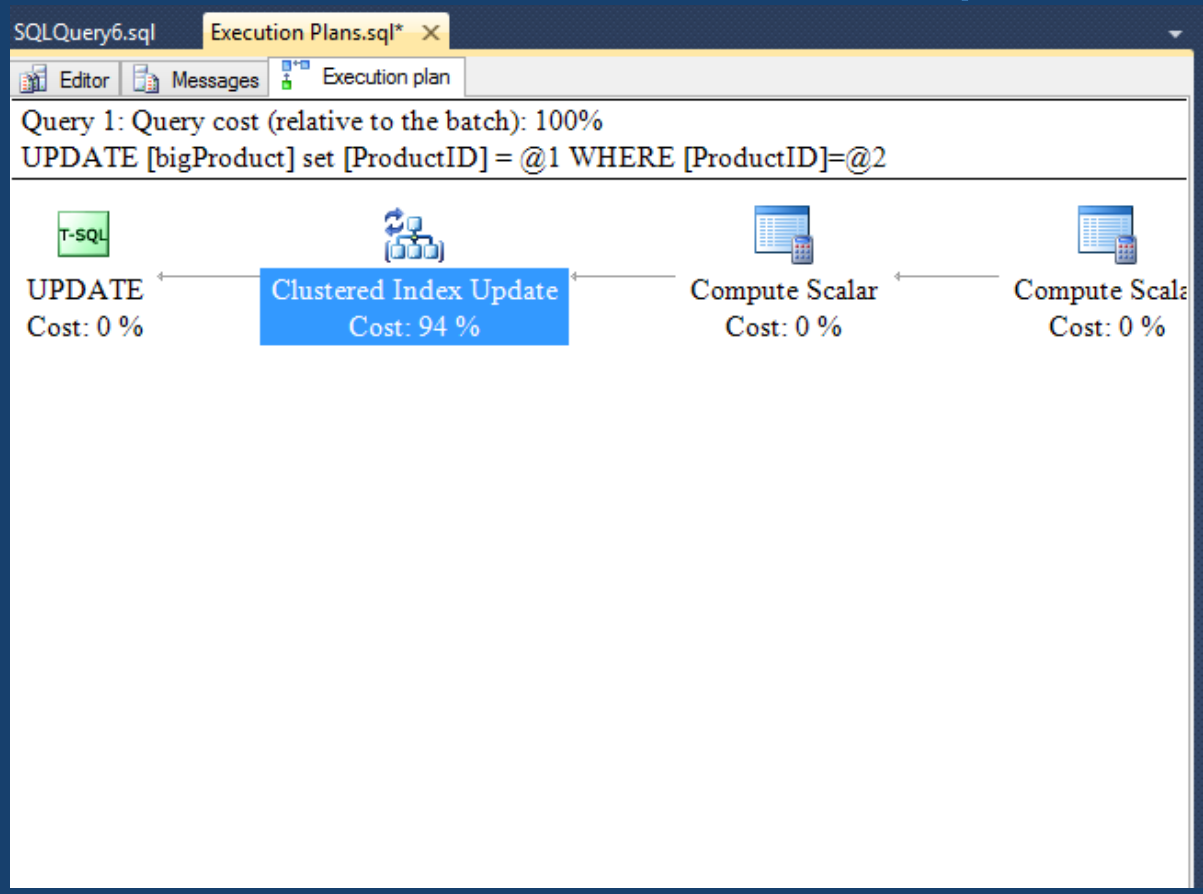


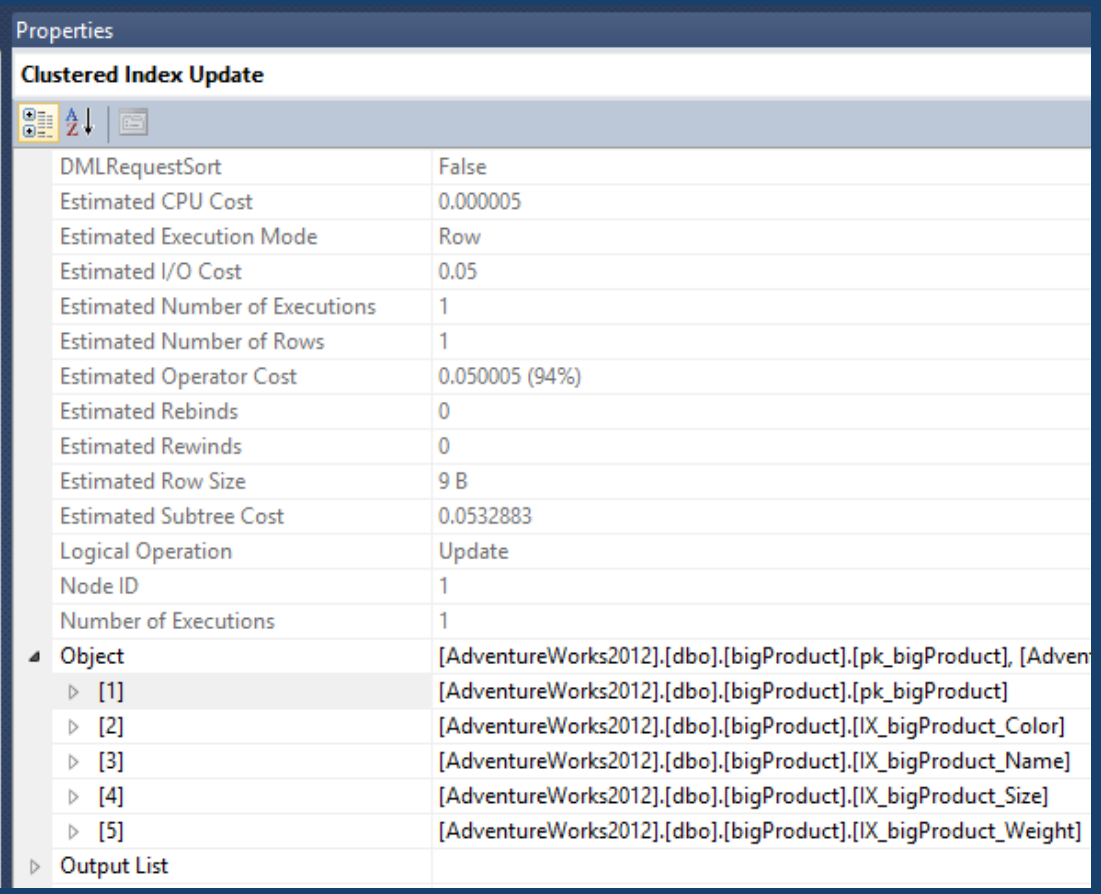
A simple UPDATE

```
UPDATE bigProduct
SET ProductID = 1005
WHERE ProductID = 1004
```

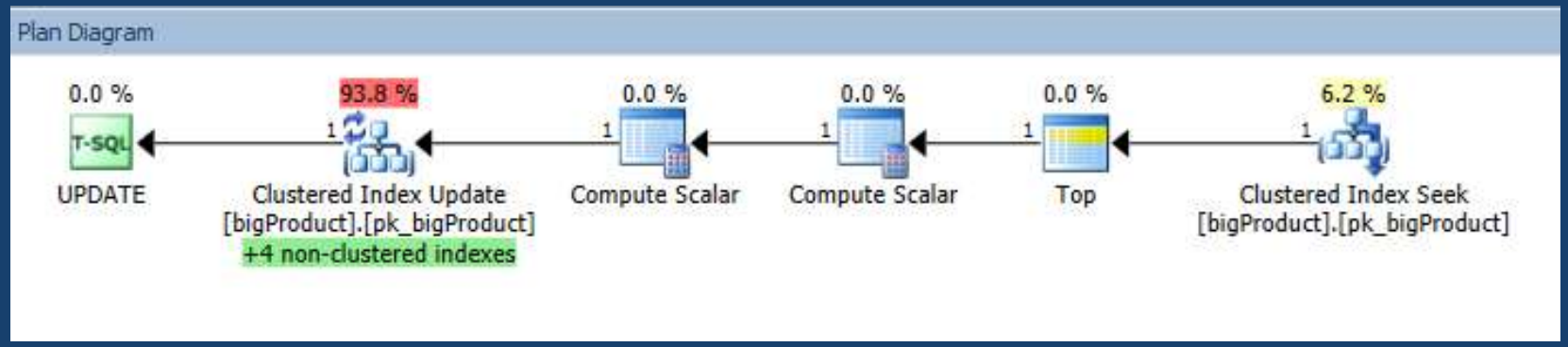


Look at Properties of the Clustered Index Update





In SQL Sentry Plan Explorer



Resources

SQL Sentry Plan Explorer

- http://sqlsentry.net
- FREE! (But check out the Pro version - extra awesome!)

Books and videos

- Don't Fear the Execution Plan
  https://www.youtube.com/watch?v=I-jjgZ51_Sw
- Spotting Trouble – and Help! – in Execution Plans
  https://www.youtube.com/watch?v=9lwpULbFK4E
- What the Execution Plan Doesn't Tell You
  https://www.youtube.com/watch?v=mbaLmVEmXbI
- SQL Server Execution Plans, Second Edition, by Grant Fritchey
  https://www.simple-talk.com/books/sql-books/sql-server-execution-plans,-second-edition,-by-grant-fritchey/
- Can You Dig It? Plan Cache Series by Jason Strate
  http://www.jasonstrate.com/2011/02/can-you-dig-it-plan-cache-series/

Concurrency

Contact me

- jborland@concurrency.com
- @grrl_geek
- http://lessthandot.com