Time statistics

IMPROVING QUERY PERFORMANCE IN SQL SERVER

Dean Smith
Founder, Atamai Analytics

SQL Server Management Studio

STATISTICS TIME in SSMS

```
SELECT UNStatisticalRegion,
        CountryName,
        Capital
FROM Nations
WHERE Capital IN
        (SELECT CityName
        FROM Cities)
            AND Capital IN
                (SELECT CityName
                    FROM Cities
                    WHERE Pop2017 > 1000000);
```

```
SQL Server parse and compile time:
   CPU time = 16 ms, elapsed time = 21 ms.

(88 rows affected)

 SQL Server Execution Times:
   CPU time = 390 ms,  elapsed time = 382 ms.
```

`STATISTICS TIME` - command reports number of milliseconds, required to parse, compile, and execute a query.

SQL Server Execution Times

```
SQL Server Execution Times:
  CPU time = 390 ms,  elapsed time = 382 ms.
```

- **CPU time**: time taken by server processors to process the query
- **Elapsed time**: total duration of the query

Example: query 1

```
SELECT UNStatisticalRegion,
       CountryName,
       Capital
FROM Nations
WHERE Capital IN
        (SELECT CityName -- 1st sub-query
        FROM Cities)
            AND Capital IN
                (SELECT CityName -- 2nd sub-query
                 FROM Cities
                 WHERE Pop2017 > 1000000);
```

Example: query 1

```
SET STATISTICS TIME ON
```

```
SELECT UNStatisticalRegion,
       CountryName,
       Capital
FROM Nations
WHERE Capital IN
        (SELECT CityName -- 1st sub-query
        FROM Cities)
            AND Capital IN
                (SELECT CityName -- 2nd sub-query
                 FROM Cities
                 WHERE Pop2017 > 1000000);
```

```
SQL Server Execution Times:
   CPU time = 391 ms,  elapsed time = 381 ms.
```

Example: query 2

```
SELECT UNStatisticalRegion,
       CountryName,
       Capital
FROM Nations n
WHERE EXISTS
    (SELECT 1
     FROM  Cities c
     WHERE n.Capital = c.CityName
        AND Pop2017 > 1000000);
```

```
SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 2 ms.
```

Example: query 2

```
SELECT UNStatisticalRegion,
       CountryName,
       Capital
FROM Nations n
WHERE EXISTS
    (SELECT 1
     FROM  Cities c
     WHERE n.Capital = c.CityName
        AND Pop2017 > 1000000);
```

```
SET STATISTICS TIME OFF
```

```
SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 2 ms.
```

Comparing queries

- First query containing two sub-queries

```
SELECT UNStatisticalRegion,
       CountryName,
       Capital
FROM Nations
WHERE Capital IN
        (SELECT CityName
        FROM Cities)
            AND Capital IN
                (SELECT CityName
                 FROM Cities
                 WHERE Pop2017 > 1000000);
```

```
SQL Server Execution Times:
   CPU time = 391 ms,  elapsed time = 381 ms.
```

- Second query using `EXISTS`

```
SELECT UNStatisticalRegion,
       CountryName,
       Capital
FROM Nations n
WHERE EXISTS
    (SELECT 1
     FROM  Cities c
     WHERE n.Capital = c.CityName
        AND Pop2017 > 1000000);
```

```
SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 2 ms.
```

Elapsed time vs. CPU time

Elapsed time

- May be variable when analyzing query time statistics
- Time best time statistics measure for the fastest running query

CPU time

- Should vary little when analyzing query time statistics
- May not be a useful measure if server processors are running in parallel

Taking an average

- Don't rely on one measurement, take an average.

```
elapsed time = 2032 ms.

elapsed time = 2060 ms.

elapsed time = 1915 ms.

elapsed time = 4009 ms.

elapsed time = 3511 ms.
```

```
Average elapsed time = 2705 ms.
```

Let's practice!

IMPROVING QUERY PERFORMANCE IN SQL SERVER

Page read statistics

IMPROVING QUERY PERFORMANCE IN SQL SERVER

Dean Smith
Founder, Atamai Analytics

Table data pages

- All data, in either memory or on the disk, is stored in 8 kilobyte size "pages"
- One page can store many rows or one value could span multiple pages
- A page can only belong to one table
- SQL Server works with pages cached in memory
- If a page is not cached in memory it is read from disk and cached in memory

Customers: data pages

Page 1

| ALFKI | Alfreds Futterkiste | Maria Anders | Sales Representative | …
| ANATR | Ana Trujillo Emparedados y helados | Ana Trujillo | Owner | …
| ANTON | Antonio Moreno Taqueria | Antonio Moreno | Owner | …
| AROUT | Around the Horn | Thomas Hardy | Sales Representative | …
| BERGS | Berglunds snabbkop | Christina Berglund | Order Administrator | …
| BLAUS | Blauer See Delikatessen | Hanna Moos | Sales Representative | …
| BLONP | Blondesddsl pere et fils | Frederique Citeaux | Marketing Manager | …

Page 2

| BOLID | Bolido Comidas preparadas | Martin Sommer | Owner | …
| BONAP | Bon app | Laurence Lebihan | Owner | …
| BOTTM | Bottom-Dollar Markets | Elizabeth Lincoln | Accounting Manager | …
| BSBEV | B s Beverages | Victoria Ashworth | Sales Representative | …
| CACTU | Cactus Comidas para llevar | Patricio Simpson | Sales Agent | …
| CENTC | Centro comercial Moctezuma | Francisco Chang | Marketing Manager | …
| CHOPS | Chop-suey Chinese | Yang Wang | Owner | …

Page 3

| COMMI | Comercio Mineiro | Pedro Afonso | Sales Associate | …
| CONSH | Consolidated Holdings | Elizabeth Brown | Sales Representative | …
| DRACD | Drachenblut Delikatessen | Sven Ottlieb | Order Administrator | …
| DUMON | Du monde entier | Janine Labrune | Owner | …
| EASTC | Eastern Connection | Ann Devon | Sales Agent | …
| ERNSH | Ernst Handel | Roland Mendel | Sales Manager | …
| FAMIA | Familia Arquibaldo | Aria Cruz | Marketing Assistant | …

Page 4

| FISSA | FISSA Fabrica Inter. Salchichas S.A. | Diego Roel | Accounting Manager | …
| FOLIG | Folies gourmandes | Martine Rance | Assistant Sales Agent | …
| FOLKO | Folk och fa HB | Maria Larsson | Owner | …
| FRANK | Frankenversand | Peter Franken | Marketing Manager | …
| FRANR | France restauration | Carine Schmitt | Marketing Manager | …
| FRANS | Franchi S.p.A. | Paolo Accorti | Sales Representative | …
| FURIB | Furia Bacalhau e Frutos do Mar | Lino Rodriguez | Sales Manager | …

STATISTICS IO in SSMS

```
SET STATISTICS IO ON
```

```
SELECT UNStatisticalRegion,
       CountryName,
       Capital
       FROM Nations
       WHERE Capital IN
         (SELECT CityName
          FROM Cities)
            AND Capital IN
               (SELECT CityName
                FROM Cities
                WHERE Pop2017 > 1000000);
```

```
(88 rows affected)
Table 'Worktable'. Scan count 1, logical reads 104568, physical reads 0, read-ahead reads 0, lob logical reads 0, ...
Table 'Cities'. Scan count 2, logical reads 548, physical reads 0, read-ahead reads 281, lob logical reads 0, lob physical reads 0, ...
Table 'Nations'. Scan count 1, logical reads 3, physical reads 1, read-ahead reads 0, lob logical reads 0, lob physical reads 0, ...
```

Logical reads

```
(88 rows affected)
Table 'Cities'. ... , logical reads 548, ... , ...
Table 'Nations'. ... , logical reads 3, ... , ...
```

logical reads: number of 8 kilobyte pages read per table

Example: query 1

```
SELECT UNStatisticalRegion,
       CountryName,
       Capital,
          (SELECT MAX(Magnitude)
          FROM Earthquakes e
          WHERE n.Capital = e.NearestPop
              AND n.Code2 = e.Country) MaxMagnitude
FROM Nations n
WHERE Capital IN
        (SELECT NearestPop
         FROM Earthquakes e
         WHERE n.Capital = e.NearestPop
            AND n.Code2 = e.Country)
                AND Capital IN
                   (SELECT NearestPop
                    FROM Earthquakes
                    WHERE Magnitude >= 7.5);
```

Example: query 1

```
SELECT UNStatisticalRegion,
       CountryName,
       Capital,
          (SELECT MAX(Magnitude) -- 1st sub-query
          FROM Earthquakes e
          WHERE n.Capital = e.NearestPop
              AND n.Code2 = e.Country) MaxMagnitude
FROM Nations n
WHERE Capital IN
        (SELECT NearestPop
         FROM Earthquakes e
         WHERE n.Capital = e.NearestPop
            AND n.Code2 = e.Country)
                AND Capital IN
                   (SELECT NearestPop
                    FROM Earthquakes
                    WHERE Magnitude >= 7.5);
```

Example: query 1

```
SELECT UNStatisticalRegion,
       CountryName,
       Capital,
          (SELECT MAX(Magnitude) -- 1st sub-query
          FROM Earthquakes e
          WHERE n.Capital = e.NearestPop
              AND n.Code2 = e.Country) MaxMagnitude
FROM Nations n
WHERE Capital IN
        (SELECT NearestPop -- 2nd sub-query
         FROM Earthquakes e
         WHERE n.Capital = e.NearestPop
            AND n.Code2 = e.Country)
                AND Capital IN
                   (SELECT NearestPop
                    FROM Earthquakes
                    WHERE Magnitude >= 7.5);
```

Example: query 1

```
SELECT UNStatisticalRegion,
       CountryName,
       Capital,
          (SELECT MAX(Magnitude) -- 1st sub-query
          FROM Earthquakes e
          WHERE n.Capital = e.NearestPop
              AND n.Code2 = e.Country) MaxMagnitude
FROM Nations n
WHERE Capital IN
        (SELECT NearestPop -- 2nd sub-query
         FROM Earthquakes e
         WHERE n.Capital = e.NearestPop
            AND n.Code2 = e.Country)
                AND Capital IN
                   (SELECT NearestPop -- 3rd sub-query
                    FROM Earthquakes
                    WHERE Magnitude >= 7.5);
```

Example: query 1

```
SET STATISTICS IO ON
```

```
SELECT UNStatisticalRegion,
       CountryName,
       Capital,
          (SELECT MAX(Magnitude) -- 1st sub-query
          FROM Earthquakes e
          WHERE n.Capital = e.NearestPop
              AND n.Code2 = e.Country) MaxMagnitude
FROM Nations n
WHERE Capital IN
        (SELECT NearestPop -- 2nd sub-query
         FROM Earthquakes e
         WHERE n.Capital = e.NearestPop
            AND n.Code2 = e.Country)
                AND Capital IN
                   (SELECT NearestPop -- 3rd sub-query
                    FROM Earthquakes
                    WHERE Magnitude >= 7.5);
```

```
Table 'Earthquakes'. ..., logical reads 54, ...
'Nations'. ..., logical reads 3, ...
```

Example: query 2

```
SELECT n.UNStatisticalRegion,
       n.CountryName,
       n.Capital,
       MAX(e.Magnitude)
FROM Nations n
INNER JOIN Earthquakes e
     ON n.Capital = e.NearestPop
         AND n.Code2 = e.Country
WHERE e.Magnitude >=7.5
GROUP BY n.UNStatisticalRegion,
        n.CountryName,
        n.Capital
```

Example: query 2

```
SELECT n.UNStatisticalRegion,
       n.CountryName,
       n.Capital,
       MAX(e.Magnitude)
FROM Nations n
INNER JOIN Earthquakes e
     ON n.Capital = e.NearestPop
         AND n.Code2 = e.Country
WHERE e.Magnitude >=7.5
GROUP BY n.UNStatisticalRegion,
        n.CountryName,
        n.Capital
```

```
SET STATISTICS IO OFF
```

```
Table 'Earthquakes'. ..., logical reads 18, ...
'Nations'. ..., logical reads 3, ...
```

Comparing queries

- First query containing three sub-queries

```
Table 'Earthquakes'. ..., logical reads 54, ...
'Nations'. ..., logical reads 3, ...
```

```
SQL Server parse and compile time:
   CPU time = 29 ms, elapsed time = 29 ms.
```

- Second query using an `INNER JOIN`

```
Table 'Earthquakes'. ..., logical reads 18, ...
'Nations'. ..., logical reads 3, ...
```

```
SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 3 ms.
```

Let's practice!

IMPROVING QUERY PERFORMANCE IN SQL SERVER

Indexes

IMPROVING QUERY PERFORMANCE IN SQL SERVER

Dean Smith
Founder, Atamai Analytics

datacamp

What is an index?

- Structure to improve speed of accessing data from a table
- Used to locate data quickly without having to scan the entire table
- Useful for improving performance of queries with filter conditions
- Applied to table columns
- Typically added by a database administrator

Clustered and nonclustered indexes

Clustered Index

- Analogy : dictionary
- Table data pages are ordered by the column(s) with the index
- Only one allowed per table
- Speeds up search operations

Clustered and nonclustered indexes

Clustered Index

- Analogy : dictionary
- Table data pages are ordered by the column(s) with the index
- Only one allowed per table
- Speeds up search operations

Non-clustered Index

- Analogy: text book with an index at the back
- Structure contains an ordered layer of index pointers to unordered table data pages
- A table can have more than one
- Improves insert and update operations

Clustered index: B-tree structure

- ROOT NODE
- BRANCH NODES
- PAGE NODES

Clustered index: B-tree structure

```
ROOT NODE:
                                    A G O W
```

```
BRANCH NODES:
                A B E F                 G H J K                 O P S T
```

```
PAGE NODES:
          Page 1                  Page 2                  Page 3                  Page 4
      Index Column |...       Index Column |...       Index Column | ...      Index Column | ...
      A | ...                 E | ...                 I | ...                 M | ...
      B | ...                 F | ...                 J | ...                 N | ...
      C | ...                 G | ...                 K | ...                 O | ...
      D | ...                 H | ...                 L | ...                 P | ...
      ...                     ...                     ...                     ...
```

Customers table without clustered index

```
SELECT *
FROM Customers
WHERE CustomerID = "PARIS"
```

```
PAGE NODES:
            Page 1
         CustomerID |...
         ALFKI | ...
         ANATR | ...
         BLONP | ...
         BSBEV | ...
         ...
```

Customers table without clustered index

```
SELECT *
FROM Customers
WHERE CustomerID = "PARIS"
```

```
PAGE NODES:
            Page 1               Page 2
         CustomerID |...     CustomerID|...
         ALFKI | ...         FOLIG | ...
         ANATR | ...         FRANK | ...
         BLONP | ...         GALED | ...
         BSBEV | ...         GREAL | ...
         ...                 ...
```

Customers table without clustered index

```
SELECT *
FROM Customers
WHERE CustomerID = "PARIS"
```

```
PAGE NODES:
            Page 1                Page 2                Page 3
         CustomerID |...      CustomerID|...       CustomerID | ...
         ALFKI | ...          FOLIG | ...          LILAS | ...
         ANATR | ...          FRANK | ...          LINOD | ...
         BLONP | ...          GALED | ...          MEREP | ...
         BSBEV | ...          GREAL | ...          MORGK | ...
         ...                  ...                  ...
```

Customers table without clustered index

```
SELECT *
FROM Customers
WHERE CustomerID = "PARIS"
```

```
PAGE NODES:
            Page 1              Page 2              Page 3              Page 4
         CustomerID |...     CustomerID|...      CustomerID | ...     CustomerID | ...
         ALFKI | ...         FOLIG | ...         LILAS | ...         OCEAN | ...
         ANATR | ...         FRANK | ...         LINOD | ...         PARIS | ...
         BLONP | ...         GALED | ...         MEREP | ...
         BSBEV | ...         GREAL | ...         MORGK | ...
         ...                 ...                 ...                 ...
```

Customers table with clustered index

```
SELECT *
FROM Customers
WHERE CustomerID = "PARIS"
```

```
ROOT NODE:
                                        ALFKI FOLIG OLDWO WOLZA
```

```
BRANCH NODES:
              ALFKI BONAP DRACD FISSA    FOLIG GALED LILAS NORTS    OCEAN OLDWO QUICK WOLZA
```

```
PAGE NODES:
                    Page 1              Page 2              Page 3              Page 4
                 CustomerID |...     CustomerID|...      CustomerID | ...     CustomerID | ...
                 ALFKI | ...         FOLIG | ...         LILAS | ...         OCEAN | ...
                 ANATR | ...         FRANK | ...         LINOD | ...         PARIS | ...
                 BLONP | ...         GALED | ...         MEREP | ...         PICCO | ...
                 BSBEV | ...         GREAL | ...         MORGK | ...         QUICK | ...
                 ...                 ...                 ...                 ...
```

Customers table with clustered index

```
SELECT *
FROM Customers
WHERE CustomerID = "PARIS"
```

```
ROOT NODE:
                                                OLDWO WOLZA
```

```
BRANCH NODES:
              ALFKI BONAP DRACD FISSA    FOLIG GALED LILAS NORTS    OCEAN OLDWO QUICK WOLZA
```

```
PAGE NODES:
                    Page 1              Page 2              Page 3              Page 4
                CustomerID |...     CustomerID|...      CustomerID | ...     CustomerID | ...
                ALFKI | ...         FOLIG | ...         LILAS | ...         OCEAN | ...
                ANATR | ...         FRANK | ...         LINOD | ...         PARIS | ...
                BLONP | ...         GALED | ...         MEREP | ...         PICCO | ...
                BSBEV | ...         GREAL | ...         MORGK | ...         QUICK | ...
                ...                 ...                 ...                 ...
```

Customers table with clustered index

```
SELECT *
FROM Customers
WHERE CustomerID = "PARIS"
```

```
ROOT NODE:
                                                OLDWO WOLZA
```

```
BRANCH NODES:
                                                                OLDWO QUICK
```

```
PAGE NODES:
                  Page 1              Page 2              Page 3              Page 4
                CustomerID |...     CustomerID|...     CustomerID | ...     CustomerID | ...
                ALFKI | ...         FOLIG | ...         LILAS | ...         OCEAN | ...
                ANATR | ...         FRANK | ...         LINOD | ...         PARIS | ...
                BLONP | ...         GALED | ...         MEREP | ...         PICCO | ...
                BSBEV | ...         GREAL | ...         MORGK | ...         QUICK | ...
                ...                 ...                 ...                 ...
```

Customers table with clustered index

```
SELECT *
FROM Customers
WHERE CustomerID = "PARIS"
```

```
ROOT NODE:
                                                    OLDWO WOLZA
```

```
BRANCH NODES:
                                                                OLDWO QUICK
```

```
PAGE NODES:
                                                                    Page 4
                                                                CustomerID | ...
                                                                OCEAN | ...
                                                                PARIS | ...
                                                                PICCO | ...
                                                                QUICK | ...
                                                                ...
```

Customers table with clustered index

```
SELECT *
FROM Customers
WHERE CustomerID = "PARIS"
```

```
ROOT NODE:
                                                    OLDWO WOLZA
```

```
BRANCH NODES:
                                                              OLDWO QUICK
```

```
PAGE NODES:
                                                                  Page 4
                                                              CustomerID | ...



                                                              PARIS | ...
```

Clustered index: example

```
SET STATISTICS IO ON
```

```
SELECT *
FROM PlayerStats
WHERE Team = 'OKC'
```

`PlayerStats` table with no index

```
Table 'PlayerStats'. ..., logical reads 12, ...
```

`PlayerStats` table with **clustered index** on `Team`

```
Table 'PlayerStats'. ..., logical reads 2, ...
```

Let's practice!

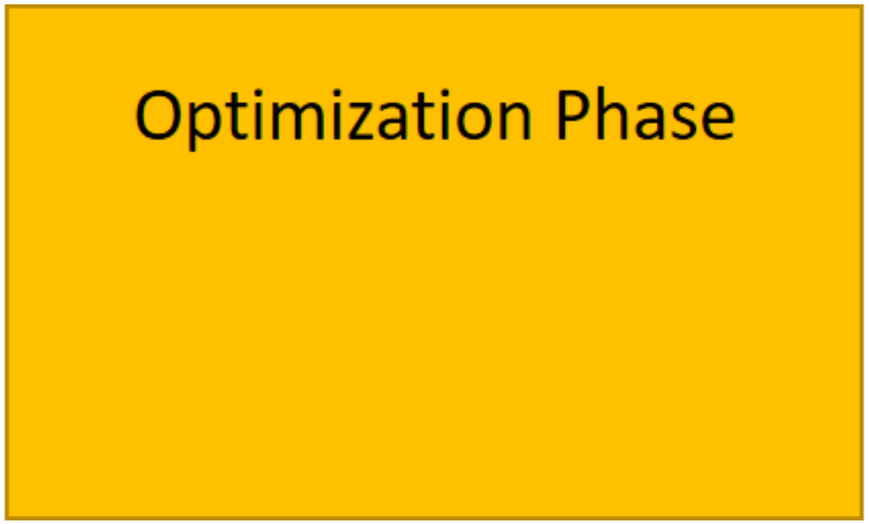
IMPROVING QUERY PERFORMANCE IN SQL SERVER

Execution plans

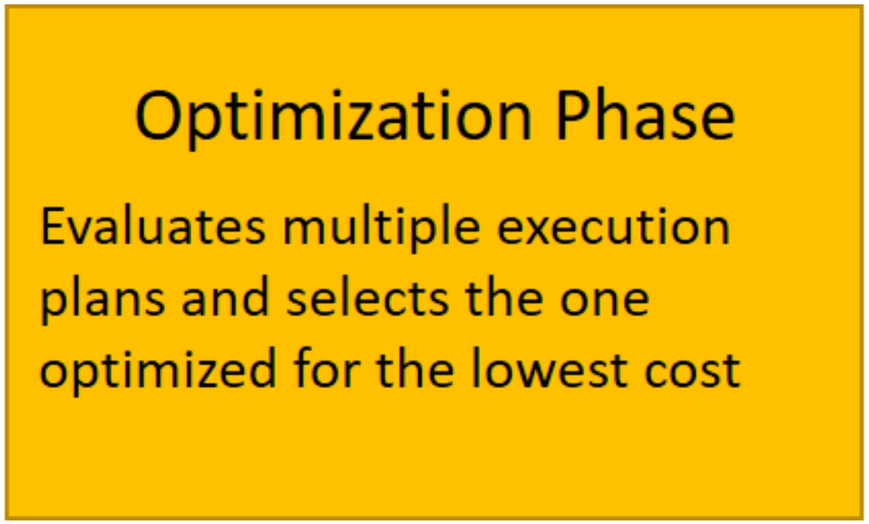
IMPROVING QUERY PERFORMANCE IN SQL SERVER

Dean Smith
Founder, Atamai Analytics

Optimization phase



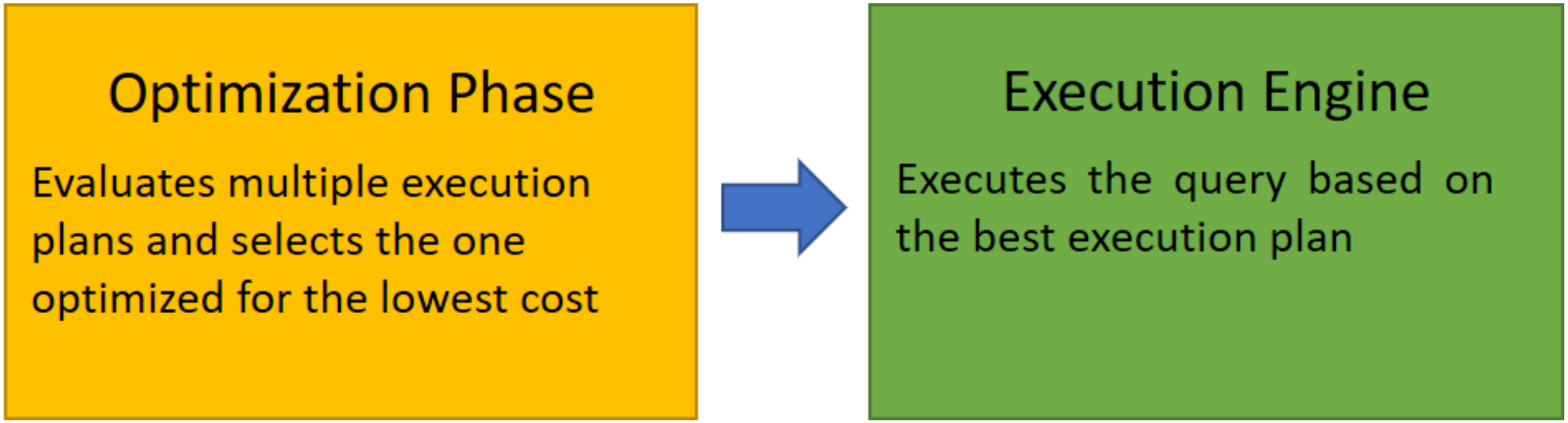
Optimization phase



Cost parameters evaluated include:

- Processor usage
- Memory usage
- Data page reads

Optimization phase



Cost parameters evaluated include;

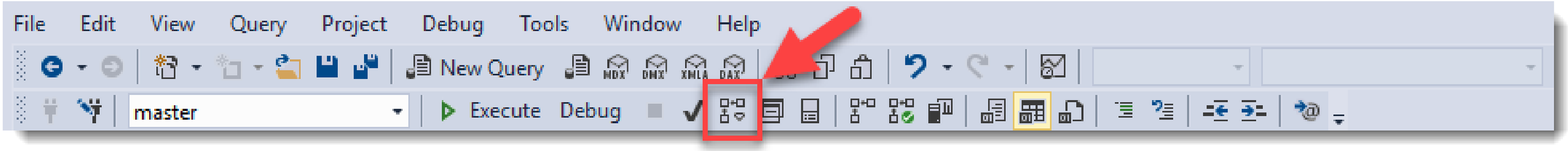
- Processor usage
- Memory usage
- Data page reads

Information from execution plans

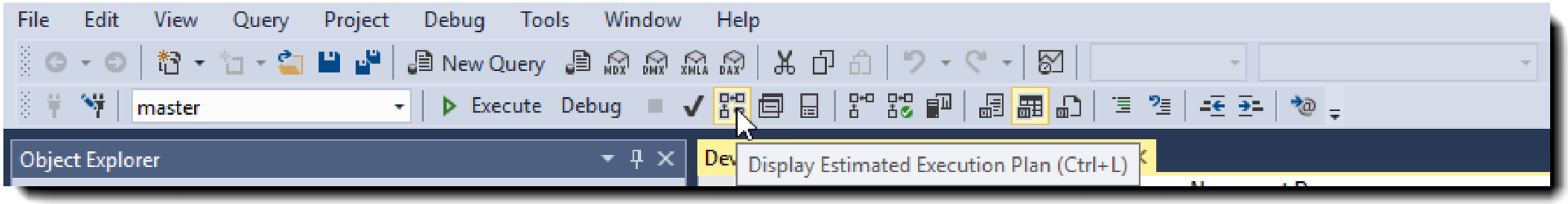
Execution plans can provide information on:

- Whether indexes were used
- Types of joins used
- Location and relative costs of:
  - filter conditions
  - sorting
  - aggregations

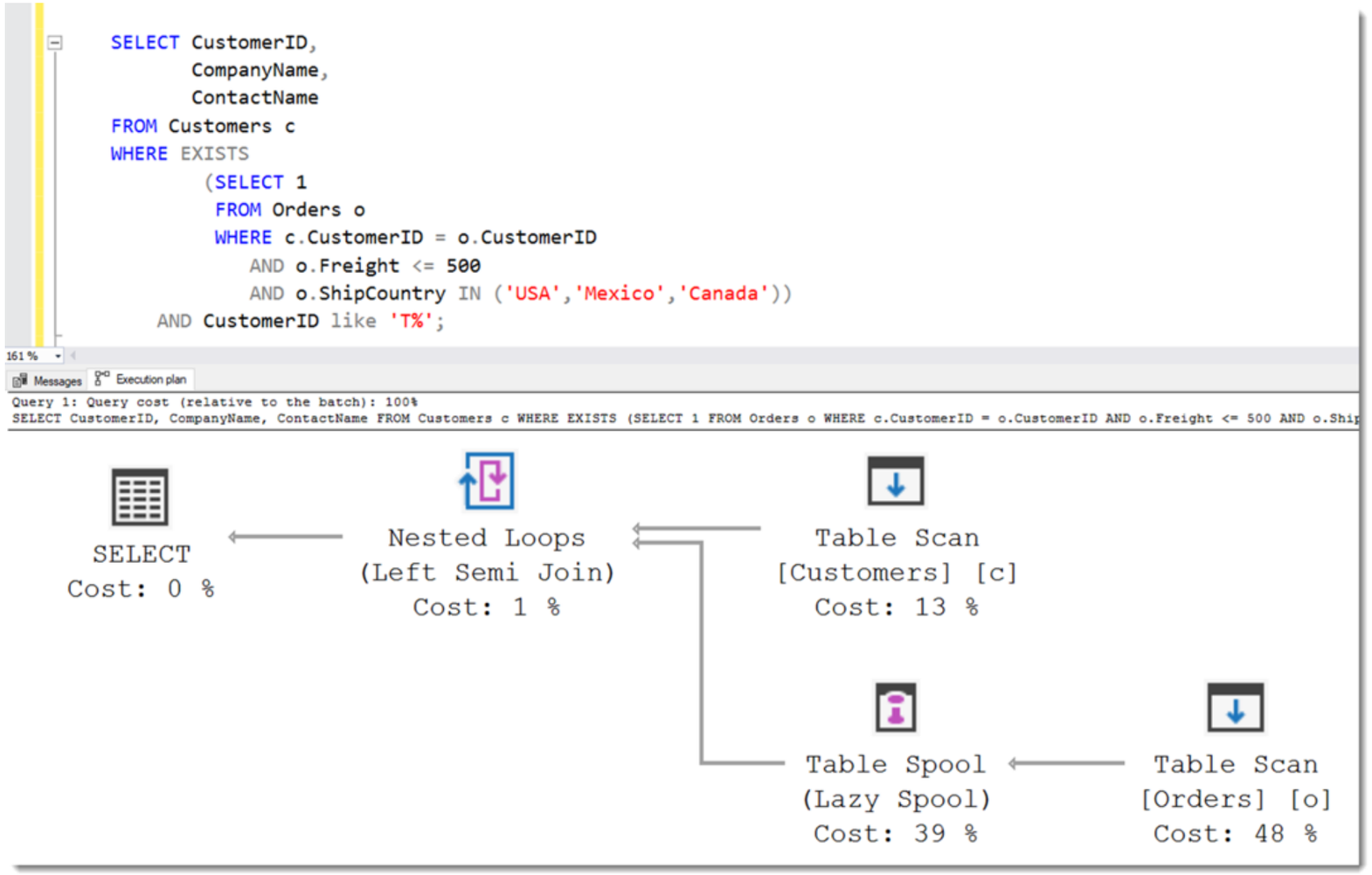
Estimated execution plan in SSMS



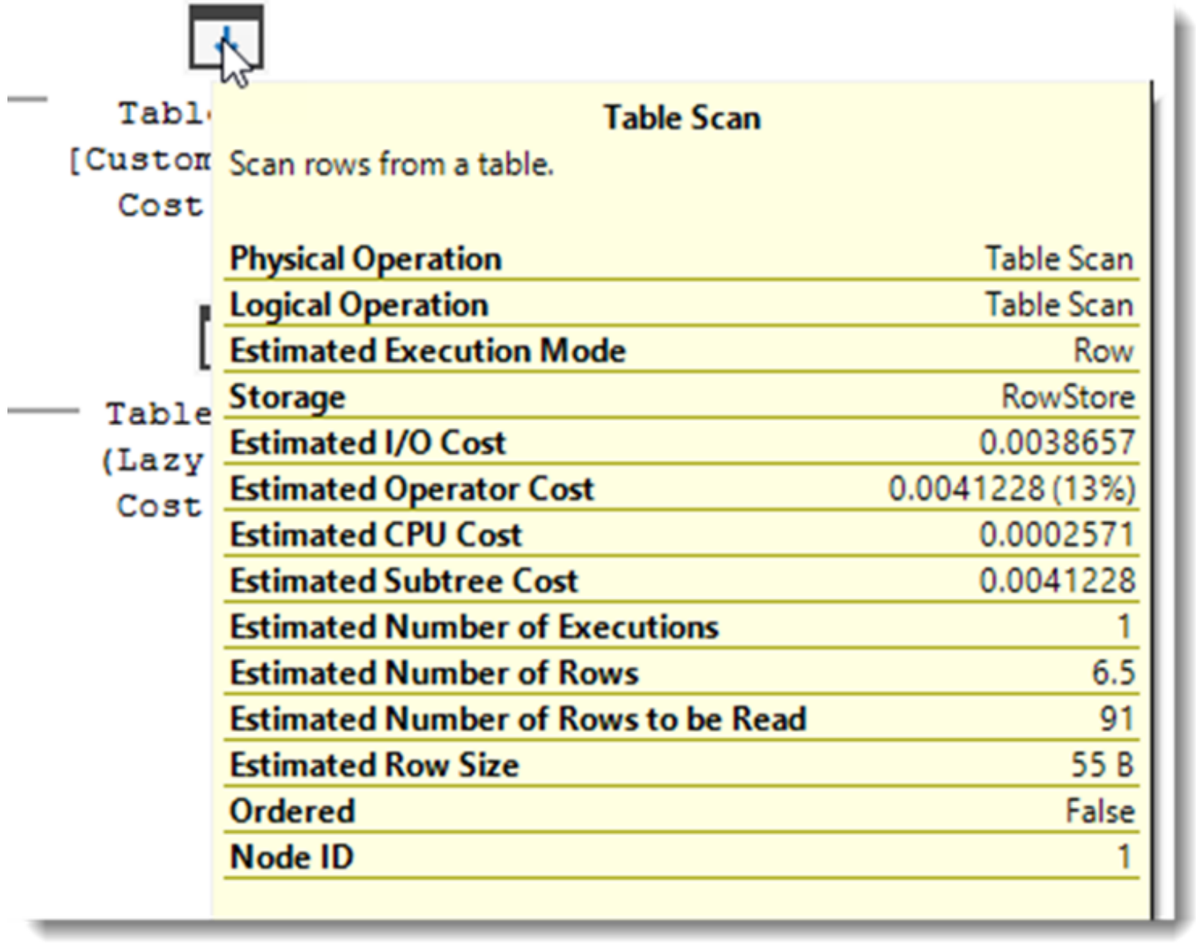
Estimated execution plan in SSMS



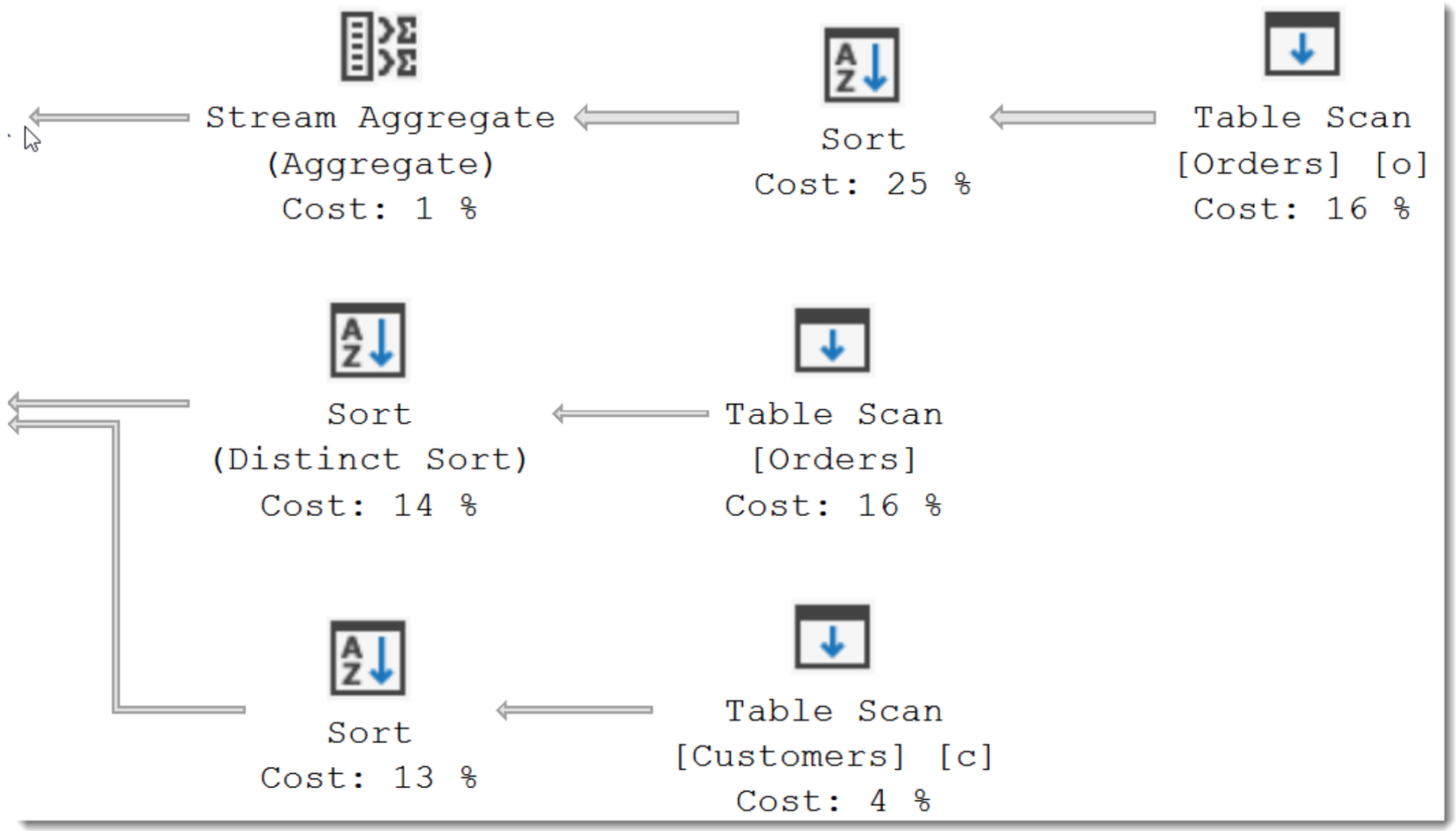
Viewing executions plans in SSMS



Operator statistics

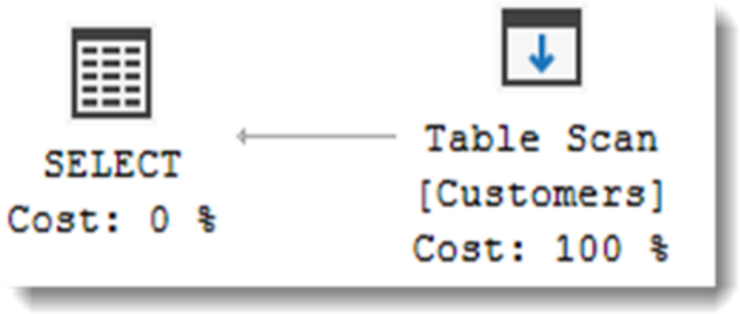


Reading execution plans

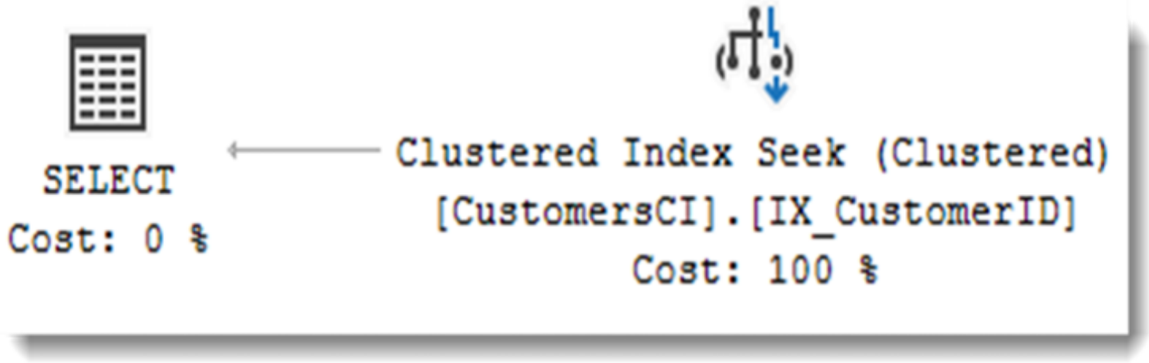


Index example

```
SELECT *
FROM Customers
WHERE CustomerID = 'PARIS';
```

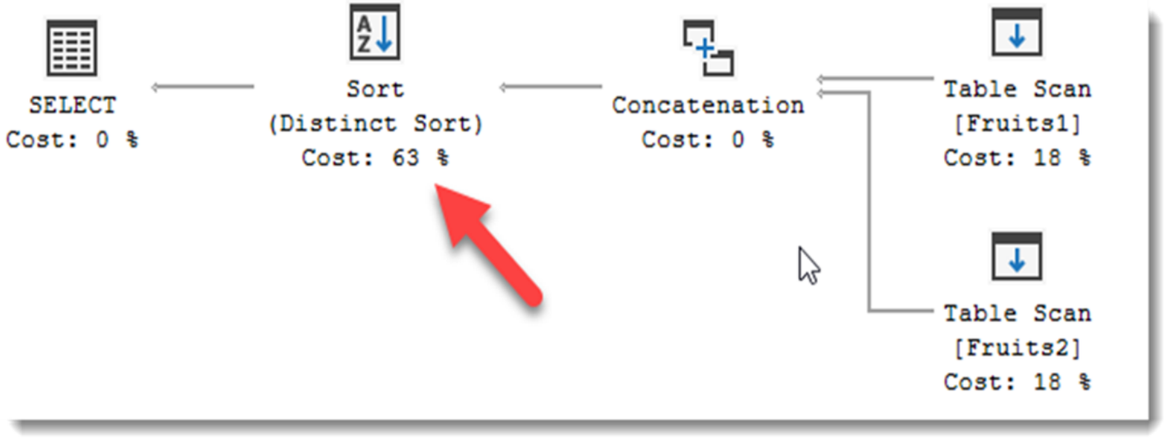



```
SELECT *
FROM CustomersCI
WHERE CustomerID = 'PARIS';
```

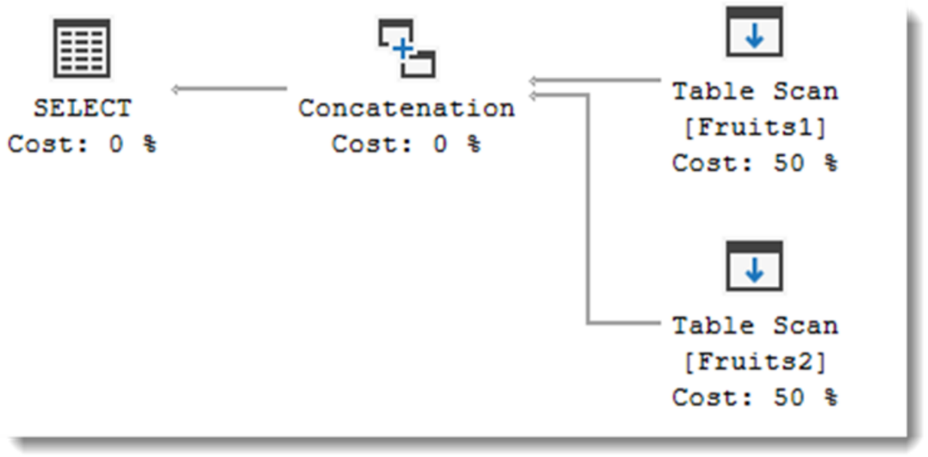


Sort operator example

```
SELECT FruitName, FruitType
FROM Fruits1

UNION

SELECT FruitName, FruitType
FROM Fruits2;
```

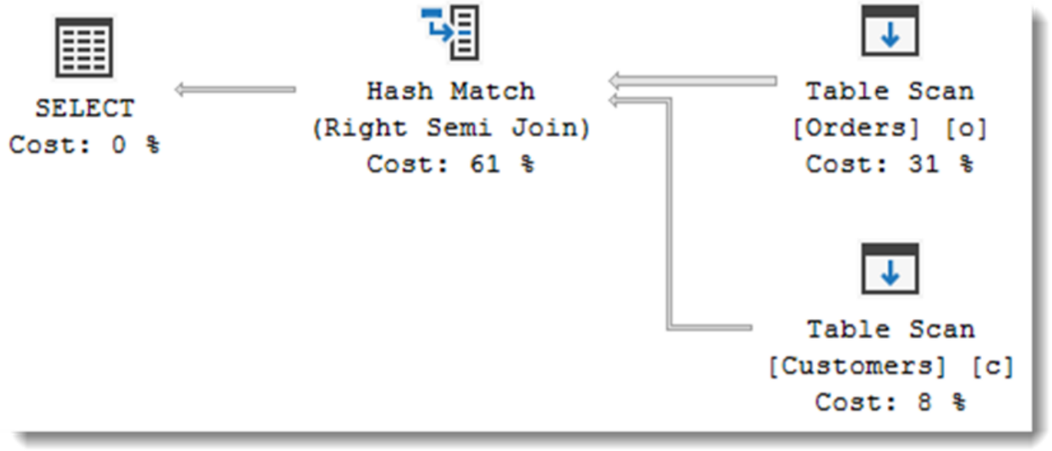



```
SELECT FruitName, FruitType
FROM Fruits1

UNION ALL

SELECT FruitName, FruitType
FROM Fruits2;
```

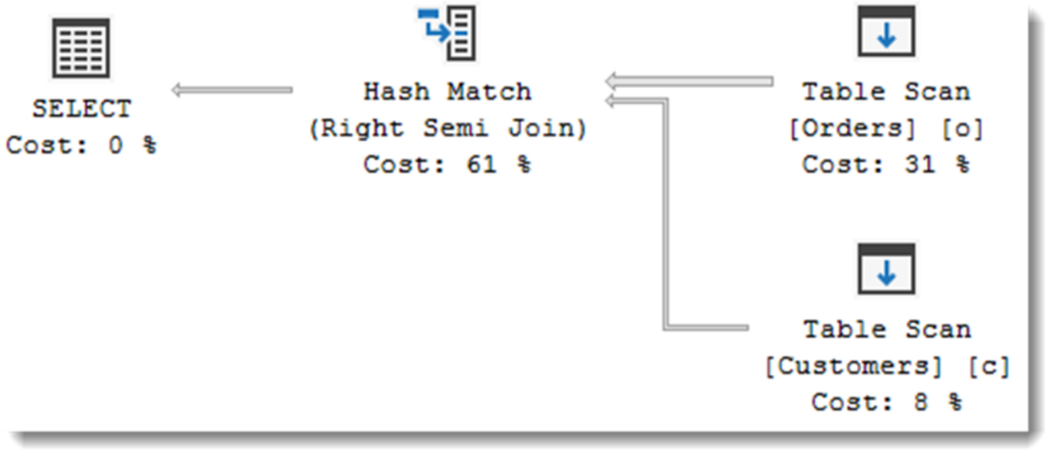


The same execution plan?

```
SELECT *
FROM Customers
WHERE CustomerID IN
    (SELECT CustomerID
     FROM Orders);
```




```
SELECT *
FROM Customers c
WHERE EXISTS
    (SELECT 1
     FROM Orders o
     WHERE c.CustomerID = o.CustomerID);
```



Let's practice!

IMPROVING QUERY PERFORMANCE IN SQL SERVER

Query performance tuning: final notes

IMPROVING QUERY PERFORMANCE IN SQL SERVER

Dean Smith
Founder, Atamai Analytics

Final notes

```
SQL Server Execution Times:
  CPU time = 390 ms,  elapsed time = 382 ms.
```

- Time statistics examples and exercises presented in this chapter are reported in milliseconds.

- In the real world, it would not be uncommon to work with large complex queries that run for ten minutes, one hour or more.

Final notes

- Query statistics, indexes, and execution plans are all advanced topics with many books and websites devoted to each one.

- Communicate with your database administrator regarding the permissions required to use query performance tuning tools and commands in SQL Server.

- Don't rely on one tool or command for query performance tuning. They can often complement one another.

Conclusion

IMPROVING QUERY PERFORMANCE IN SQL SERVER