

**INDEX – exemplu laborator**

```
create database C
go
use C
go

CREATE TABLE Ceaiuri(
    Cid INT PRIMARY KEY IDENTITY,
    Denumire VARCHAR(50),
    Pret INT)
```

La crearea unui tabel ce are cheie primară – se crează automat indexul clustered în acel tabel (pe cheia primară).

Pe un tabel – maxim 1 index clustered (pe cheia primară) ai oricâți non-clustered.

Index-ii se pot verifica în tabelul **sys.indexes** (fiecare tabel are 152 index-i + cei pe care îi definim noi)

```
SELECT * FROM sys.indexes

SELECT name FROM sys.indexes

select * from sys.indexes
where name like 'N%'
```

Verificarea utilității utilizării index-ilor se poate verifica folosind Include Live Query Statistics.

Primul index va fi cel mai efficient.

Index-ii non-clustered se crează pe câmpurile utilizate în clauzele WHERE, ORDER BY, JOIN-uri (pe cheile externe – foreign keys) și atunci când utilitatea nu se observă se poate recurge la adăugarea mai multor înregistrări în tabele.

**ORDER BY**

La **ORDER BY** NU contează câmpurile care sunt incluse în **SELECT** la utilizarea index-ilor (ci doar ceea ce este în ORDER BY).

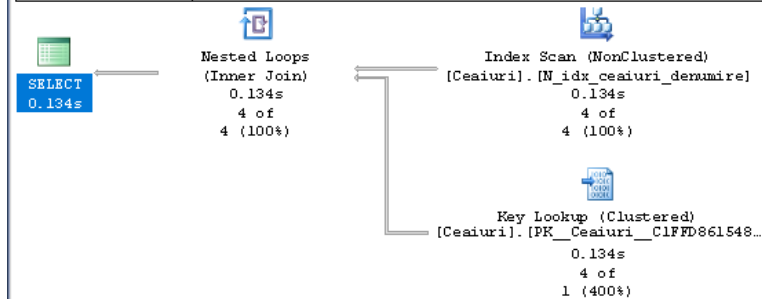
<pre>-- on primary key SELECT * FROM Ceaiuri ORDER BY Cid</pre>	<p>Query 1: Query cost (relative to the batch): 100% progress:100% SELECT * FROM Ceaiuri ORDER BY Cid</p> <p>Clustered Index Scan (Clustered) [Ceaiuri].[PK_Ceaiuri_C1FFD861548...] 0 of 1 (0%)</p>																																																												
<pre>-- with index clustered SELECT * FROM Ceaiuri ORDER BY Denumire</pre>	<p>Query 1: Query cost (relative to the batch): 100% progress:100% SELECT * FROM Ceaiuri ORDER BY Denumire</p> <p>Sort 0 of 1 (0%)</p> <p>Clustered Index Scan (Clustered) [Ceaiuri].[PK_Ceaiuri_C1FFD861548...] 0 of 1 (0%)</p> <p>E folosit tot index-ul clustered.. ca să îmbunătățim vom crea un index non-clustered pe câmpul Denumire</p>																																																												
<pre>IF EXISTS (SELECT NAME FROM sys.indexes WHERE name='N_idx_ceaiuri_denumire') DROP INDEX N_idx_ceaiuri_denumire ON Ceaiuri CREATE NONCLUSTERED INDEX N_idx_ceaiuri_denumire ON Ceaiuri(Denumire)</pre>																																																													
<pre>-- with non-clustered index on Denumire SELECT * FROM Ceaiuri ORDER BY Denumire</pre>	<p>Query 1: Query cost (relative to the batch): 100% progress:100% -- with non-clustered index on Denumire SELECT * FROM Ceaiuri ORDER BY Denumire</p> <p>Nested Loops (Inner Join) 0 of 1 (0%)</p> <p>Index Scan (NonClustered) [Ceaiuri].[N_idx_ceaiuri_denumire] 0 of 1 (0%)</p> <p>Key Lookup (Clustered) [Ceaiuri].[PK_Ceaiuri_C1FFD861548...] 0 of 1 (0%)</p> <div style="display: flex; justify-content: space-between;"> <div data-bbox="542 1150 1003 1772"> <p><b>Index Scan (NonClustered)</b> Scan a nonclustered index, entirely or only a range. Estimated operator progress: 100%</p> <table border="1"> <tr><td>Physical Operation</td><td>Index Scan</td></tr> <tr><td>Logical Operation</td><td>Index Scan</td></tr> <tr><td>Estimated Execution Mode</td><td>Row</td></tr> <tr><td>Storage</td><td>RowStore</td></tr> <tr><td>Actual Number of Rows</td><td>0</td></tr> <tr><td>Estimated Operator Cost</td><td>0.0032831 (50%)</td></tr> <tr><td>Estimated I/O Cost</td><td>0.003125</td></tr> <tr><td>Estimated CPU Cost</td><td>0.0001581</td></tr> <tr><td>Estimated Subtree Cost</td><td>0.0032831</td></tr> <tr><td>Number of Executions</td><td>1</td></tr> <tr><td>Estimated Number of Executions</td><td>1</td></tr> <tr><td>Estimated Number of Rows</td><td>1</td></tr> <tr><td>Estimated Row Size</td><td>40 B</td></tr> <tr><td>Ordered</td><td>True</td></tr> <tr><td>Node ID</td><td>1</td></tr> </table> <p><b>Object</b> [C].[dbo].[Ceaiuri].[N_idx_ceaiuri_denumire]</p> <p><b>Output List</b> [C].[dbo].[Ceaiuri].Cid, [C].[dbo].[Ceaiuri].Denumire</p> </div> <div data-bbox="1024 1115 1451 1772"> <p><b>Key Lookup (Clustered)</b> Uses a supplied clustering key to lookup on a table that has a clustered index. Estimated operator progress: 100%</p> <table border="1"> <tr><td>Physical Operation</td><td>Key Lookup</td></tr> <tr><td>Logical Operation</td><td>Key Lookup</td></tr> <tr><td>Estimated Execution Mode</td><td>Row</td></tr> <tr><td>Storage</td><td>RowStore</td></tr> <tr><td>Actual Number of Rows</td><td>0</td></tr> <tr><td>Estimated Operator Cost</td><td>0.0032831 (50%)</td></tr> <tr><td>Estimated I/O Cost</td><td>0.003125</td></tr> <tr><td>Estimated CPU Cost</td><td>0.0001581</td></tr> <tr><td>Estimated Subtree Cost</td><td>0.0032831</td></tr> <tr><td>Number of Executions</td><td>0</td></tr> <tr><td>Estimated Number of Executions</td><td>1</td></tr> <tr><td>Estimated Number of Rows</td><td>1</td></tr> <tr><td>Estimated Row Size</td><td>11 B</td></tr> <tr><td>Ordered</td><td>True</td></tr> <tr><td>Node ID</td><td>3</td></tr> </table> <p><b>Object</b> [C].[dbo].[Ceaiuri].[PK_Ceaiuri_C1FFD86154891985]</p> <p><b>Output List</b> [C].[dbo].[Ceaiuri].Pret</p> <p><b>Seek Predicates</b> Seek Keys[1]: Prefix: [C].[dbo].[Ceaiuri].Cid = Scalar Operator([C].[dbo].[Ceaiuri].[Cid])</p> </div> </div> <p>Estimated Cost Operator 50% - 50%</p>	Physical Operation	Index Scan	Logical Operation	Index Scan	Estimated Execution Mode	Row	Storage	RowStore	Actual Number of Rows	0	Estimated Operator Cost	0.0032831 (50%)	Estimated I/O Cost	0.003125	Estimated CPU Cost	0.0001581	Estimated Subtree Cost	0.0032831	Number of Executions	1	Estimated Number of Executions	1	Estimated Number of Rows	1	Estimated Row Size	40 B	Ordered	True	Node ID	1	Physical Operation	Key Lookup	Logical Operation	Key Lookup	Estimated Execution Mode	Row	Storage	RowStore	Actual Number of Rows	0	Estimated Operator Cost	0.0032831 (50%)	Estimated I/O Cost	0.003125	Estimated CPU Cost	0.0001581	Estimated Subtree Cost	0.0032831	Number of Executions	0	Estimated Number of Executions	1	Estimated Number of Rows	1	Estimated Row Size	11 B	Ordered	True	Node ID	3
Physical Operation	Index Scan																																																												
Logical Operation	Index Scan																																																												
Estimated Execution Mode	Row																																																												
Storage	RowStore																																																												
Actual Number of Rows	0																																																												
Estimated Operator Cost	0.0032831 (50%)																																																												
Estimated I/O Cost	0.003125																																																												
Estimated CPU Cost	0.0001581																																																												
Estimated Subtree Cost	0.0032831																																																												
Number of Executions	1																																																												
Estimated Number of Executions	1																																																												
Estimated Number of Rows	1																																																												
Estimated Row Size	40 B																																																												
Ordered	True																																																												
Node ID	1																																																												
Physical Operation	Key Lookup																																																												
Logical Operation	Key Lookup																																																												
Estimated Execution Mode	Row																																																												
Storage	RowStore																																																												
Actual Number of Rows	0																																																												
Estimated Operator Cost	0.0032831 (50%)																																																												
Estimated I/O Cost	0.003125																																																												
Estimated CPU Cost	0.0001581																																																												
Estimated Subtree Cost	0.0032831																																																												
Number of Executions	0																																																												
Estimated Number of Executions	1																																																												
Estimated Number of Rows	1																																																												
Estimated Row Size	11 B																																																												
Ordered	True																																																												
Node ID	3																																																												
<p>Inseram înregistrări pentru a verifica mai bine utilitatea creării index-ului non-clustered pe câmpul Denumire din tabela Ceaiuri</p>																																																													

```
INSERT INTO Ceiuri VALUES('Menta', 10), ('Tei', 11), ('Fructe', 8), ('Ghimbir', 15)
```

```
-- after adding records
SELECT * FROM Ceiuri
ORDER BY Denumire
```

Results Messages Live Query Statistics

Estimated query progress: 100% Query 1: Query cost (relative to the batch): 100%  
-- after adding records SELECT \* FROM Ceiuri ORDER BY Denumire



Index Scan (NonClustered)	
Scan a nonclustered index, entirely or only a range.	
Estimated operator progress: 100%	
Physical Operation	Index Scan
Logical Operation	Index Scan
Estimated Execution Mode	Row
Storage	RowStore
Actual Number of Rows	4
Estimated Operator Cost	0.0032864 (47%)
Estimated I/O Cost	0.003125
Estimated CPU Cost	0.0001614
Estimated Subtree Cost	0.0032864
Number of Executions	1
Estimated Number of Executions	1
Estimated Number of Rows	4
Estimated Row Size	40 B
Ordered	True
Node ID	1
<b>Object</b>	
[C].[dbo].[Ceiuri].[N_idx_ceiuri_denumire]	
<b>Output List</b>	
[C].[dbo].[Ceiuri].Cid, [C].[dbo].[Ceiuri].Denumire	

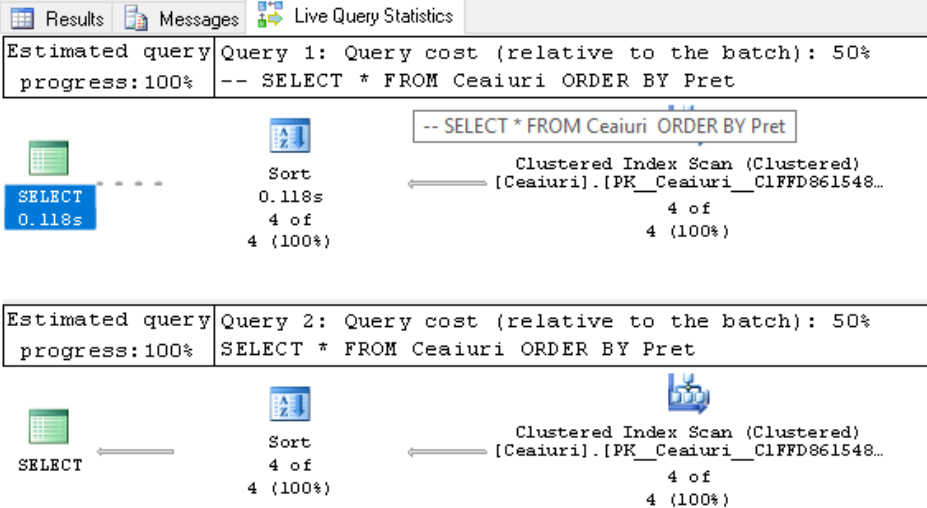
Key Lookup (Clustered)	
Uses a supplied clustering key to lookup on a table that has a clustered index.	
Estimated operator progress: 100%	
Physical Operation	Key Lookup
Logical Operation	Key Lookup
Estimated Execution Mode	Row
Storage	RowStore
Actual Number of Rows	4
Estimated Operator Cost	0.0037574 (53%)
Estimated I/O Cost	0.003125
Estimated CPU Cost	0.0001581
Estimated Subtree Cost	0.0037574
Number of Executions	4
Estimated Number of Executions	4
Estimated Number of Rows	1
Estimated Row Size	11 B
Ordered	True
Node ID	3
<b>Object</b>	
[C].[dbo].[Ceiuri].[PK_Ceiuri_C1FFD86154891985]	
<b>Output List</b>	
[C].[dbo].[Ceiuri].Pret	
<b>Seek Predicates</b>	
Seek Keys[1]: Prefix: [C].[dbo].[Ceiuri].Cid = Scalar Operator([C].[dbo].[Ceiuri].[Cid])	

Estimated Cost Operator = 47% (pentru index-ul non-clustered)

Estimated Cost Operator = 53% (pentru index-ul clustered)

- Deci, eficient cel non-clustered

Dacă în ORDER BY sunt incluse și câmpuri ce nu au index-i (Clustered/Non-Clustered) pe ele, atunci se va folosi index-ul Clustered.

<pre>-- use a field what has no index on it (Pret) and the used index will be the Clustered one SELECT * FROM Ceaiuri ORDER BY Denumire, Pret  -- SELECT * FROM Ceaiuri ORDER BY Pret</pre>	 <p>Estimated query progress:100% Query 1: Query cost (relative to the batch): 50% -- SELECT * FROM Ceaiuri ORDER BY Pret</p> <p>SELECT 0.118s</p> <p>Sort 0.118s 4 of 4 (100%)</p> <p>Clustered Index Scan (Clustered) [Ceaiuri].[PK_Ceaiuri_C1FFD861548...] 4 of 4 (100%)</p> <p>Estimated query progress:100% Query 2: Query cost (relative to the batch): 50% SELECT * FROM Ceaiuri ORDER BY Pret</p> <p>SELECT</p> <p>Sort 4 of 4 (100%)</p> <p>Clustered Index Scan (Clustered) [Ceaiuri].[PK_Ceaiuri_C1FFD861548...] 4 of 4 (100%)</p>
---	---

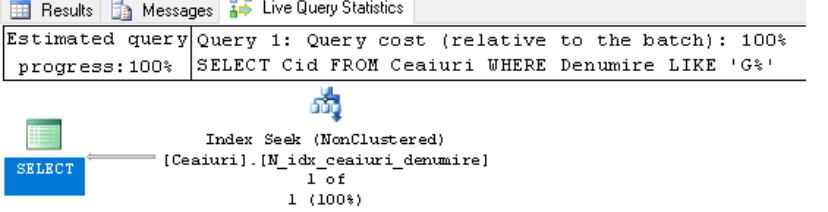
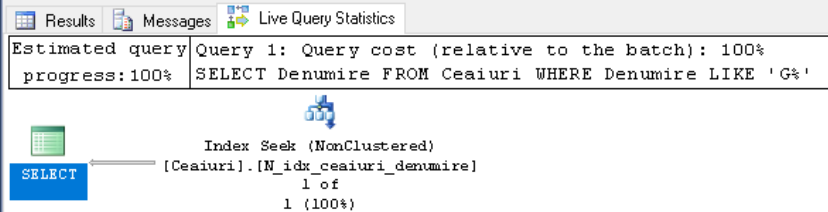
## WHERE

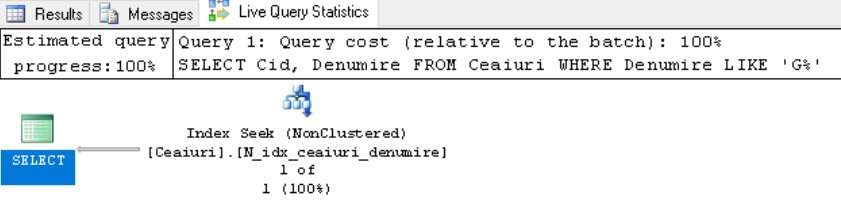
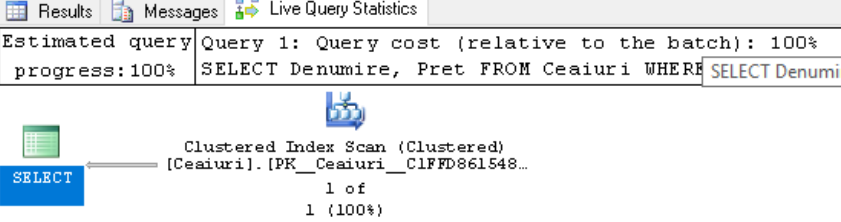
Index-ii se pot verifica și dacă sunt incluși într-un view.

<pre>-- WHERE CREATE VIEW vCeaiuri AS     SELECT *     FROM Ceaiuri     WHERE Denumire LIKE 'G%' GO  SELECT * FROM vCeaiuri</pre>	
---	--

La **WHERE** contează câmpurile care sunt incluse în **SELECT** la utilizarea index-ilor.

Se vor folosi index-ii non-clustered doar atunci când avem în **SELECT** câmpuri pe care există index-i (clustered/non-clustered).

Există Index Clustered pe Cid si Index Non-Clustered pe Denumire	
<pre>SELECT Cid FROM Ceaiuri WHERE Denumire LIKE 'G%' -- nonclustered</pre>	 <p>Estimated query progress:100% Query 1: Query cost (relative to the batch): 100% SELECT Cid FROM Ceaiuri WHERE Denumire LIKE 'G%'</p> <p>SELECT</p> <p>Index Seek (NonClustered) [Ceaiuri].[N_idx_ceaiuri_denumire] 1 of 1 (100%)</p>
<pre>SELECT Denumire FROM Ceaiuri WHERE Denumire LIKE 'G%' -- nonclustered</pre>	 <p>Estimated query progress:100% Query 1: Query cost (relative to the batch): 100% SELECT Denumire FROM Ceaiuri WHERE Denumire LIKE 'G%'</p> <p>SELECT</p> <p>Index Seek (NonClustered) [Ceaiuri].[N_idx_ceaiuri_denumire] 1 of 1 (100%)</p>

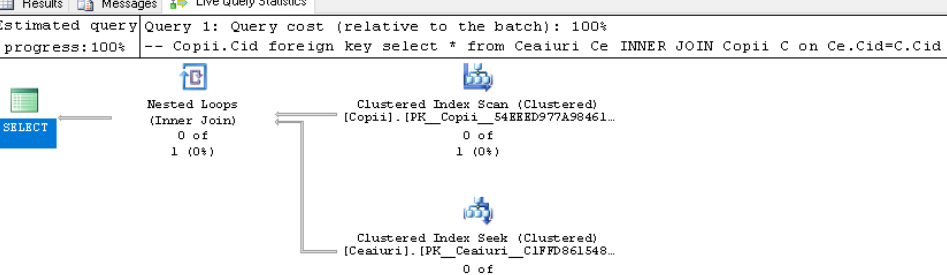
<pre>SELECT Cid, Denumire FROM Ceaiuri WHERE Denumire LIKE 'G%' -- nonclustered</pre>	 <p>Query 1: Query cost (relative to the batch): 100% progress:100% SELECT Cid, Denumire FROM Ceaiuri WHERE Denumire LIKE 'G%'</p> <p>Index Seek (NonClustered) [Ceaiuri].[N_idx_ceaiuri_denumire] 1 of 1 (100%)</p>
<pre>SELECT Denumire, Pret FROM Ceaiuri WHERE Denumire LIKE 'G%' -- clustered , because on the field Pret does not exist an index</pre>	 <p>Query 1: Query cost (relative to the batch): 100% progress:100% SELECT Denumire, Pret FROM Ceaiuri WHERE Denumire LIKE 'G%'</p> <p>Clustered Index Scan (Clustered) [Ceaiuri].[PK_Ceaiuri_C1FFD861548...] 1 of 1 (100%)</p>

## JOIN-uri

La **JOIN** contează câmpurile care sunt incluse în **SELECT** la utilizarea index-ilor.

Mai adăugăm un nou tabel, Copii, în care Cid va fi cheia străină (foreign key)

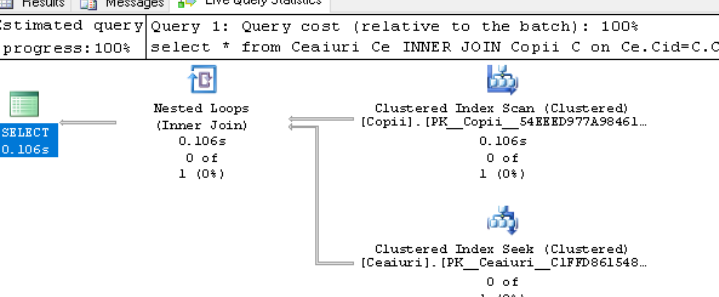
```
create table Copii(
CopiiId int primary key identity,
Nume varchar(50),
Varsta int,
Cid int foreign key references Ceaiuri(Cid))
```

<pre>-- Copii.Cid foreign key select * from Ceaiuri Ce INNER JOIN Copii C on Ce.Cid=C.Cid -- index Scan (Clustered) + Index Seek (Clustered)</pre>	 <p>Query 1: Query cost (relative to the batch): 100% progress:100% -- Copii.Cid foreign key select * from Ceaiuri Ce INNER JOIN Copii C on Ce.Cid=C.Cid</p> <p>Nested Loops (Inner Join) 0 of 1 (0%)</p> <p>Clustered Index Scan (Clustered) [Copii].[PK_Copii_548EED977A98461...] 0 of 1 (0%)</p> <p>Clustered Index Seek (Clustered) [Ceaiuri].[PK_Ceaiuri_C1FFD861548...] 0 of 1 (0%)</p>
--	---

Vom crea un index non-clustered pe tabela Copii pe Cid (cheia străină – foreign key)

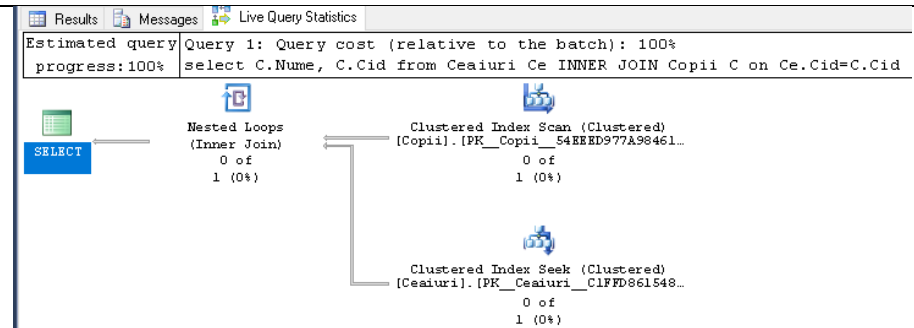
```
IF EXISTS (SELECT NAME FROM sys.indexes WHERE name='N_idx_copii_cid')
DROP INDEX N_idx_copii_cid ON Copii
CREATE NONCLUSTERED INDEX N_idx_copii_cid ON Copii(Cid)
```

```
select *
from Ceaiuri Ce INNER JOIN Copii C
on Ce.Cid=C.Cid
-- index Scan (Clustered) + Index
Seek (Clustered)
```

<pre>SELECT 0.106s</pre>	 <p>Query 1: Query cost (relative to the batch): 100% progress:100% select * from Ceaiuri Ce INNER JOIN Copii C on Ce.Cid=C.Cid</p> <p>Nested Loops (Inner Join) 0.106s 0 of 1 (0%)</p> <p>Clustered Index Scan (Clustered) [Copii].[PK_Copii_548EED977A98461...] 0.106s 0 of 1 (0%)</p> <p>Clustered Index Seek (Clustered) [Ceaiuri].[PK_Ceaiuri_C1FFD861548...] 0 of 1 (0%)</p>
--------------------------	--

Încă nu se vede nici o diferență

```
select C.Nume, C.Cid
from Ceiuri Ce INNER JOIN Copii C
on Ce.Cid=C.Cid
-- index Scan (Clustered) + Index
Seek (Clustered)
```

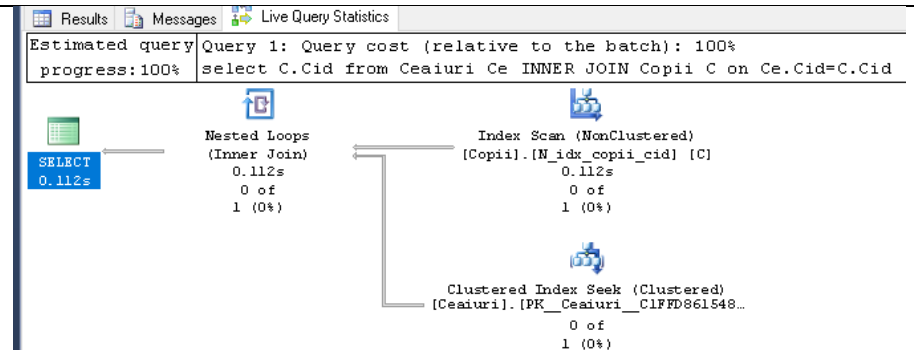


Încă nu se vede nici o diferență

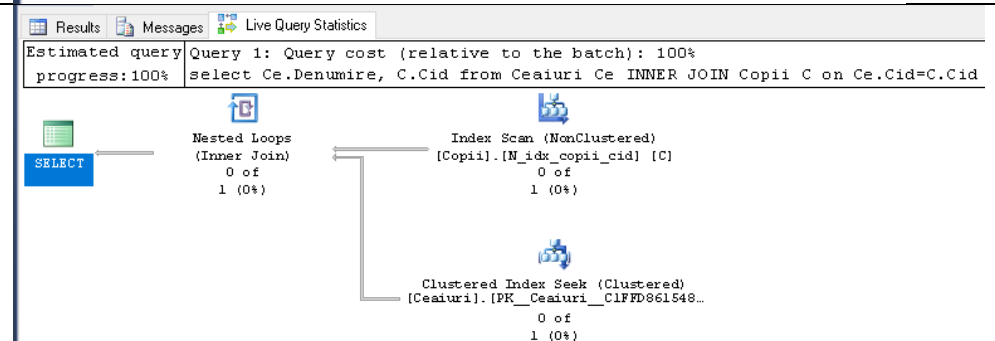
Utilitatea index-ilor (și folosirea index-ului non-clustered) se poate observa doar atunci când sunt implicate în SELECT câmpurile ce au index-i (clustered și/sau non-clustered) creați pe ele (și nu alte câmpuri)

-- the non-clustered index (on a field) is used only when it is combined with another clustered/non-clustered index (on another field(s))

```
select C.Cid
from Ceiuri Ce INNER JOIN Copii C
on Ce.Cid=C.Cid
-- index Scan (NonClustered) +
Index Seek (Clustered)
```



```
select Ce.Denumire, C.Cid
from Ceiuri Ce INNER JOIN Copii C
on Ce.Cid=C.Cid
-- index Scan (NonClustered) +
Index Seek (Clustered)
```



Observație:

Crearea unui index non-clustered pe un câmp ce nu este folosit in Select, Where, Join's, Order by, este inutilă și trebuie evitată.

-- creating a non-clustered index on a field that will not be used, it is useless

-- we have a non-clustered index on Ceiuri in the field Denumire, but we do not use the field Denumire in our queries

<pre>select Cid, Pret from Ceaiuri order by Pret -- clustered index</pre>	<p>The execution plan shows a 'SELECT' operation connected to a 'Sort' operation (4 of 4, 100%), which is connected to a 'Clustered Index Scan (Clustered)' operation on '[Ceaiuri].[PK_Ceaiuri_ClFFD8618DB...]' (4 of 4, 100%). The query statistics indicate a query cost of 100% and 100% progress.</p>
<pre>select Cid, Pret from Ceaiuri where Pret&gt;10 -- clustered index</pre>	<p>The execution plan shows a 'SELECT' operation connected to a 'Clustered Index Scan (Clustered)' operation on '[Ceaiuri].[PK_Ceaiuri_ClFFD8618DB...]' (2 of 2, 100%). The query statistics indicate a query cost of 100% and 100% progress.</p>
<pre>select C.Cid, Pret from Ceaiuri Ce inner join Copii C on Ce.Cid=C.Cid -- clustered index -- the non-clustered index used is on the Copii.Cid field created</pre>	<p>The execution plan shows a 'SELECT' operation connected to a 'Nested Loops (Inner Join)' operation (0 of 1, 0%). The 'Nested Loops' operation is connected to an 'Index Scan (NonClustered)' operation on '[Copii].[N_idx_copii_cid] [C]' (0 of 1, 0%), which is connected to a 'Clustered Index Seek (Clustered)' operation on '[Ceaiuri].[PK_Ceaiuri_ClFFD8618DB...]' (0 of 1, 0%). The query statistics indicate a query cost of 100% and 100% progress.</p>