Optimizarea performanței în MS SQL Server

Seminar 5

Optimizarea interogărilor - metodologie

- Identificarea așteptărilor (bottleneck) la nivel de server
  - I/O latches
  - Log update
  - Blocare
  - Altele
- Corelare așteptări – cozi (queues)
- Restrângere la nivel de bază de date/fișier
- Optimizarea interogărilor problematice

Latch

- **Latch**: un **tip special de blocare sistem low-level** care este menținută pe întreaga durată a unei **operații fizice** asupra unei **pagini din memorie**, ce are scopul de a proteja **consistența memoriei**
- **Latch**-urile sunt **un mecanism intern al SQL Server** care are scopul de a proteja resursele de memorie partajate (cum ar fi paginile sau structurile de date din memorie aflate în buffer pool) și de a coordona accesul la aceste resurse
- Deoarece latch-urile sunt un mecanism intern al SQL Server, care nu este expus în afara SQLOS (SQL Server Operating System), ele nu pot fi administrate de către utilizatori, spre deosebire de blocările tranzacționale (locks), care pot fi administrate cu ajutorul hint-urilor NO LOCK
- **Latch-urile I/O** sunt obținute atunci când se citesc sau se scriu date pe disc
- **Latch-urile buffer** sunt obținute atunci când se accesează pagini din memorie

Identificarea așteptărilor

DMV (Dynamic Management Views) returnează informații despre starea server-ului care pot fi folosite pentru monitorizarea stării server-ului, diagnosticarea problemelor și reglarea performanței

Dynamic management view-ul sistem *sys.dm_os_wait_stats* returnează informații despre toate așteptările întâmpinate de thread-urile care s-au executat

- **wait_type –** numele tipului de așteptare
  - Așteptări resursă (blocări, latches, rețea, I/O)
  - Așteptări queue
  - Așteptări externe (au loc atunci când un SQL Server worker așteaptă terminarea unui eveniment extern, cum ar fi apelul unei proceduri stocate extended sau o interogare linked server)
- **waiting_tasks_count –** numărul de așteptări pentru tipul de așteptare

Identificarea așteptărilor

- **wait_time_ms –** timpul total de așteptare pentru tipul de așteptare în milisecunde (acest timp include signal_wait_time_ms)
- **max_wait_time_ms –** timpul maxim de așteptare pentru tipul de așteptare
- **signal_wait_time_ms –** diferența dintre momentul în care waiting thread a fost semnalat și momentul în care a început să ruleze

Resetarea counter-elor:

```
DBCC SQLPERF ('sys.dm_os_wait_stats', CLEAR);
```

Corelare așteptări - queues

Dynamic management view-ul sistem *sys.dm_os_performance_counters* returnează câte o înregistrare pentru fiecare **performance counter** menținut de server

- **object_name –** categoria de care aparține counter-ul
- **counter_name –** numele counter-ului relativ la categorie (se poate suprapune pentru diverse valori object_name)
- **instance_name –** numele instanței counter-ului (de multe ori conține numele bazei de date)
- **cntr_value –** valoarea înregistrată sau calculată a counter-ului
- **cntr_type –** tipul counter-ului definit de Performance Monitor

Corelare așteptări - queues

Sunt disponibile peste 500 de counter-e:

- Access Methods, User Settable, Buffer Manager, Broker Statistics, SQL Errors, Latches, Buffer Partition, SQL Statistics, Locks, Buffer Node, Plan Cache, Cursor Manager by Type, Memory Manager, General Statistics, Databases, Catalog Metadata, Broker Activation, Broker/DBM Transport, Transactions, Cursor Manager Total, Exec Statistics, Wait Statistics etc.
- cntr_type=65792 → cntr_value conține valoarea efectivă
- cntr_type=537003264 → cntr_value conține rezultate în timp real care trebuie împărțite la o “bază” pentru a obține valoarea efectivă (altfel, sunt inutile)
  - valoarea trebuie împărțită la o valoare "bază" pentru a obține un raport, iar rezultatul se poate înmulți cu 100 pentru a-l exprima în procente

Corelare așteptări - queues

- cntr_type=272696576 → cntr_value conține valoarea de bază
  - Counter-ele sunt bazate pe timp
  - Counter-ele sunt cumulative
  - Se utilizează un tabel secundar pentru stocarea valorilor intermediare pentru statistici
- cntr_type=1073874176 și cntr_type=1073939712
- Se obține atât valoarea (1073874176) cât și valoarea de bază (1073939712)
- Se obțin ambele valori din nou (după 15 secunde)
- Pentru a obține rezultatul vizat, se împart diferențele între ele:

UnitsPerSec = (cv2 –cv1) / (bv2 –bv1) / 15.0

Restrângere la nivel de bază de date/fișier

Dynamic management view-ul sistem *sys.dm_io_virtual_file_stats* returnează statistici I/O pentru fișierele de date și loguri

- **Parametri**:
  - database_id (NULL=toate bazele de date), funcția DB_ID este utilă
  - file_id (NULL=toate fișierele), funcția FILE_IDEX este utilă
- **Tabel returnat**:
  - **database_id**
  - **file_id**
  - **sample_ms** – numărul de milisecunde de la pornirea calculatorului
  - **num_of_reads** – numărul de citiri fizice realizate
  - **num_of_bytes_read** – numărul total de octeți citiți

Restrângere la nivel de bază de date/fișier

- **io_stall_read_ms** – timpul total de așteptare al utilizatorilor pentru citiri
- **num_of_writes** – numărul de scrieri efectuate
- **num_of_bytes_written** – numărul total de octeți scriși
- **io_stall_write_ms** – timpul total de așteptare al utilizatorilor pentru finalizarea scrierilor
- **io_stall** – suma io_stall_read_ms și io_stall_write_ms
- **file_handle** – Windows file handle pentru fișier
- **size_on_disk_bytes** – numărul total de octeți folosiți pe disc pentru fișier

- Exemplu:

```
SELECT * FROM sys.dm_io_virtual_file_stats(DB_ID('SGBDIR'),NULL);
```

Indecși

Sunt printre **principalii factori** care influențează **performanța interogărilor**

- **Efect asupra**: filtrării, join-ului, sortării, grupării, evitării blocării și a deadlock-ului, etc.
- **Efect în modificări**: efect **pozitiv** în localizarea înregistrărilor și efect **negativ** al costului modificărilor în index

Înțelegerea indecșilor și a mecanismelor interne ale acestora

- Clustered/nonclustered
- Indecși cu una sau mai multe coloane
- View-uri indexate și indecși pe coloane calculate
- Scenarii de acoperire
- Intersecție

Indecși

- În funcție de mediu și de raportul dintre interogările SELECT și modificările datelor, trebuie să apreciați în ce măsură costul adițional de mentenanță a indecșilor se justifică prin îmbunătățirea performanței interogărilor
- Indecșii cu mai multe coloane tind să fie mult mai utili decât indecșii cu o coloană
- E mai probabil ca query optimizer-ul să utilizeze indecși cu mai multe coloane pentru a acoperi o interogare
- View-urile indexate au un cost de întreținere mai ridicat decât indecșii standard
- Opțiunea WITH SCHEMABINDING este obligatorie pentru crearea view-urilor indexate

Unelte pentru analiza performanței interogărilor

- **Plan de execuție grafic**
- **STATISTICS IO**: număr de scanări, citiri logice, citiri fizice, citiri read ahead
- **STATISTICS TIME**: durată și timp CPU net
- **SHOWPLAN_TEXT**: plan estimat
- **SHOWPLAN_ALL**: plan estimat detaliat
- **STATISTICS PROFILE**: plan efectiv detaliat
- **SET STATISTICS XML**: informații detaliate despre performanța efectivă în format XML
- **SET SHOWPLAN_XML**: informații detaliate despre performanța estimată în format XML

Optimizarea interogărilor

Evaluarea planurilor de execuție

- Secvență de operații fizice/logice

Factori de optimizare:

- Predicatul de căutare utilizat
- Tabelele implicate în join
- Condițiile de join
- Dimensiunea rezultatului
- Lista de indecși

Scop: evitarea celor mai slabe planuri pentru interogări

SQL Server utilizează un query optimizer bazat pe cost

STATISTICS IO și STATISTICS TIME

```
USE AdventureWorks2014;
GO
DBCC DROPCLEANBUFFERS;
DBCC FREEPROCCACHE;
GO
SET STATISTICS IO ON;
SET STATISTICS TIME ON;
GO
SELECT * FROM Person.BusinessEntityContact;
```

150 %

Results | Messages

```
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 249 ms.

(909 rows affected)
Table 'BusinessEntityContact'. Scan count 1, logical reads 8, physical reads 1,
read-ahead reads 8, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 424 ms.
```

- **DBCC DROPCLEANBUFFERS** – elimină toate clean buffers din buffer pool și obiectele columnstore din columnstore object pool
- **DBCC DROPCLEANBUFFERS** se poate folosi pentru a testa interogări utilizând un cold buffer cache fără a da shut down și restart server-ului
- **DBCC FREEPROCCACHE** – șterge toate elementele din plan cache

STATISTICS IO și STATISTICS TIME

```
USE AdventureWorks2014;
GO
DBCC DROPCLEANBUFFERS;
DBCC FREEPROCCACHE;
GO
SET STATISTICS IO ON;
SET STATISTICS TIME ON;
GO
SELECT * FROM Person.BusinessEntityContact;
```

150 %

Results | Messages

```
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 249 ms.

(909 rows affected)
Table 'BusinessEntityContact'. Scan count 1, logical reads 8, physical reads 1,
read-ahead reads 8, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 424 ms.
```

- **CPU time** – resursele CPU utilizate pentru a executa o interogare
- **Elapsed time** – cât timp a durat execuția interogării

STATISTICS IO și STATISTICS TIME

```
USE AdventureWorks2014;
GO
DBCC DROPCLEANBUFFERS;
DBCC FREEPROCCACHE;
GO
SET STATISTICS IO ON;
SET STATISTICS TIME ON;
GO
SELECT * FROM Person.BusinessEntityContact;
```

150 %

Results | Messages

```
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 249 ms.

(909 rows affected)
Table 'BusinessEntityContact'. Scan count 1, logical reads 8, physical reads 1,
read-ahead reads 8, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 424 ms.
```

- **Physical reads** – numărul de pagini citite de pe disc
- **Read-ahead reads** – numărul de pagini plasate în cache pentru interogare
- **Scan count** – de câte ori au fost accesate tabelele
- **Logical reads** – numărul de pagini citite din data cache

STATISTICS IO și STATISTICS TIME

```
USE AdventureWorks2014;
GO
DBCC DROPCLEANBUFFERS;
DBCC FREEPROCCACHE;
GO
SET STATISTICS IO ON;
SET STATISTICS TIME ON;
GO
SELECT ReorderPoint FROM Production.Product WHERE ReorderPoint>375;
```

150 %

Results | Messages

```
SQL Server parse and compile time:
   CPU time = 16 ms, elapsed time = 376 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

(181 rows affected)
Table 'Product'. Scan count 1, logical reads 15, physical reads 1, read-ahead re

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 30 ms
```

STATISTICS IO și STATISTICS TIME

```
--Se definește un index pentru a optimiza interogarea

USE [AdventureWorks2014];

GO

CREATE NONCLUSTERED INDEX [IX_Product_ReorderPoint_ASC]

ON [Production].[Product]

(ReorderPoint ASC);

GO
```

STATISTICS IO și STATISTICS TIME

```
USE AdventureWorks2014;
GO
DBCC DROPCLEANBUFFERS;
DBCC FREEPROCCACHE;
GO
SET STATISTICS IO ON;
SET STATISTICS TIME ON;
GO
SELECT ReorderPoint FROM Production.Product WHERE ReorderPoint>375;
```

150 %

Results | Messages

```
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 105 ms.
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

(181 rows affected)
Table 'Product'. Scan count 1, logical reads 2, physical reads 1, read-ahead rea

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 9 ms.
```

SHOWPLAN_ALL

- Dacă **SHOWPLAN_ALL** este setat pe ON, Microsoft SQL Server nu execută instrucțiunile Transact-SQL, ci returnează informații detaliate despre cum sunt executate instrucțiunile și oferă estimări ale cerințelor de resurse pentru instrucțiuni
- **SHOWPLAN_ALL** returnează informații sub forma unui set de înregistrări care formează un hierarchical tree ce reprezintă pașii efectuați de SQL Server query processor pe măsură ce execută fiecare instrucțiune
- Fiecare instrucțiune reflectată în output conține o singură înregistrare cu textul instrucțiunii, urmată de mai multe înregistrări cu detaliile pașilor de execuție
- Sintaxa:

```
SET SHOWPLAN_ALL { ON | OFF }
```

SHOWPLAN_ALL

- Exemplu:

```
SET SHOWPLAN_ALL ON;
GO
SELECT COUNT(*) cRows FROM HumanResources.Shift;
GO
SET SHOWPLAN_ALL OFF;
GO
```

100 %

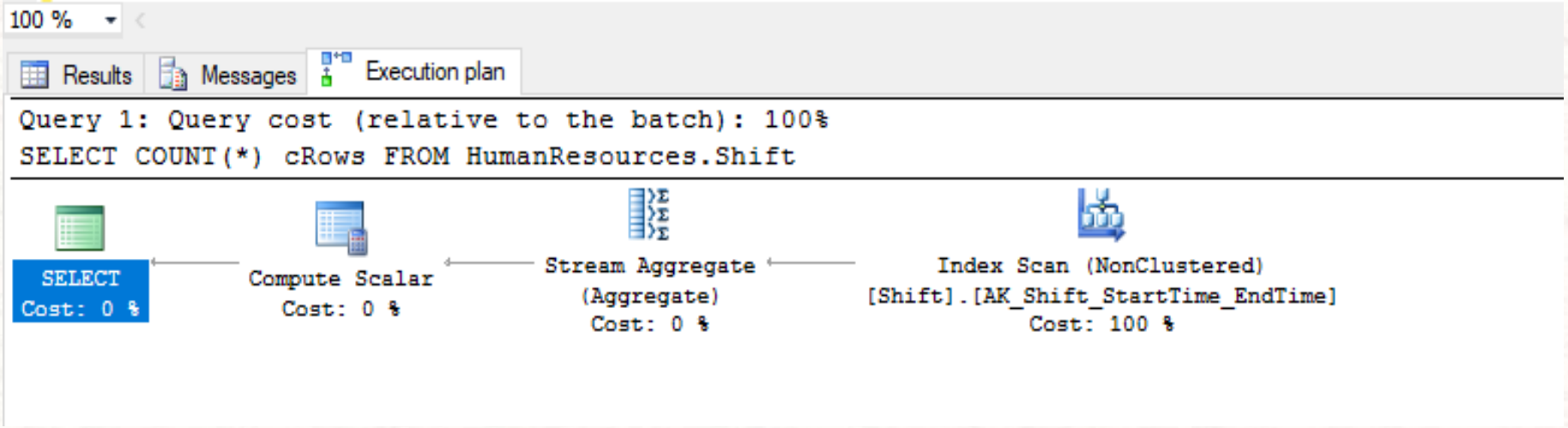
Results | Messages

	StmtText
1	SELECT COUNT(*) cRows FROM HumanResources.Shift;
2	|--Compute Scalar(DEFINE:([Expr1002]=CONVERT_IMPLICIT(int,[Expr1003],0)))
3	|--Stream Aggregate(DEFINE:([Expr1003]=Count(*)))
4	|--Index Scan(OBJECT:([AdventureWorks2012].[HumanResources].[Shift].[AK_Shift_StartTime_EndTime]))

Plan de execuție grafic

- Exemplu:

```
USE AdventureWorks2014
GO
SELECT COUNT(*) cRows FROM HumanResources.Shift;
GO
```

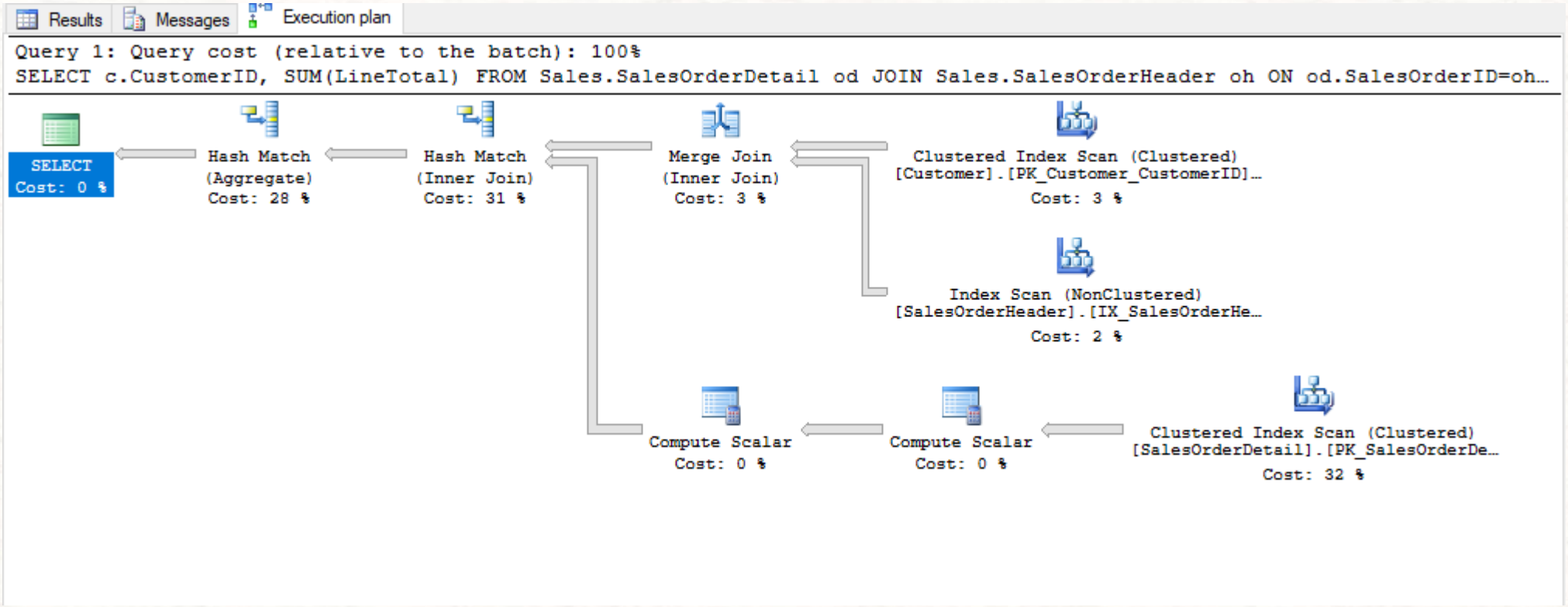


Plan de execuție grafic

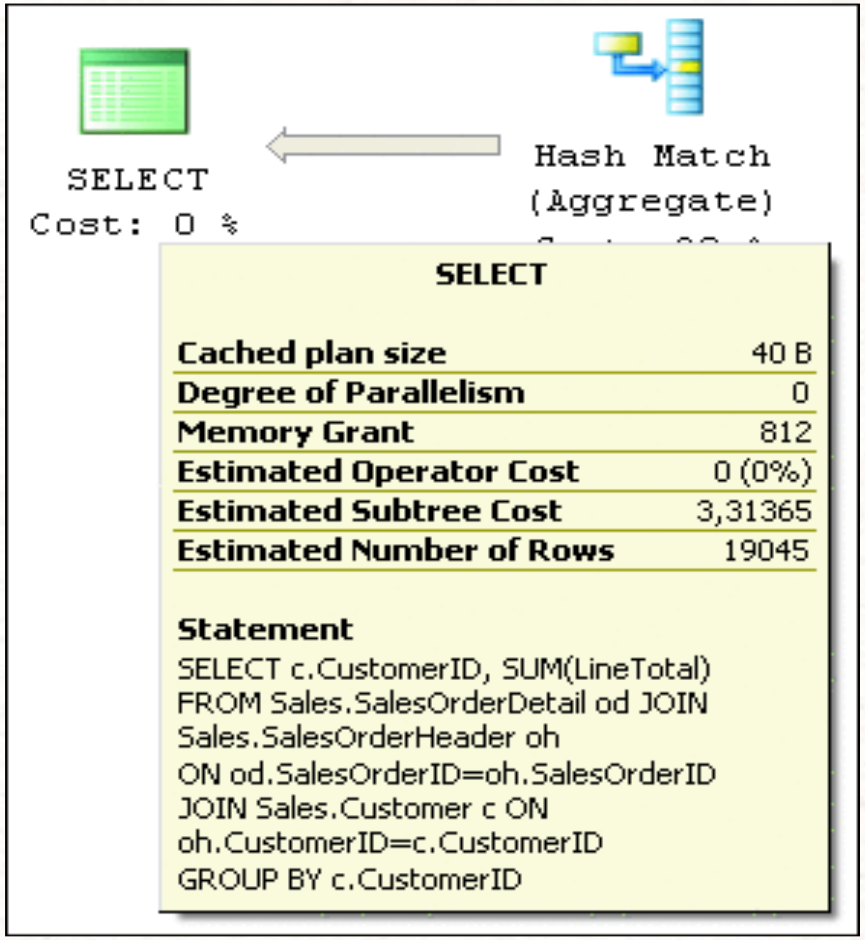
- Exemplu:

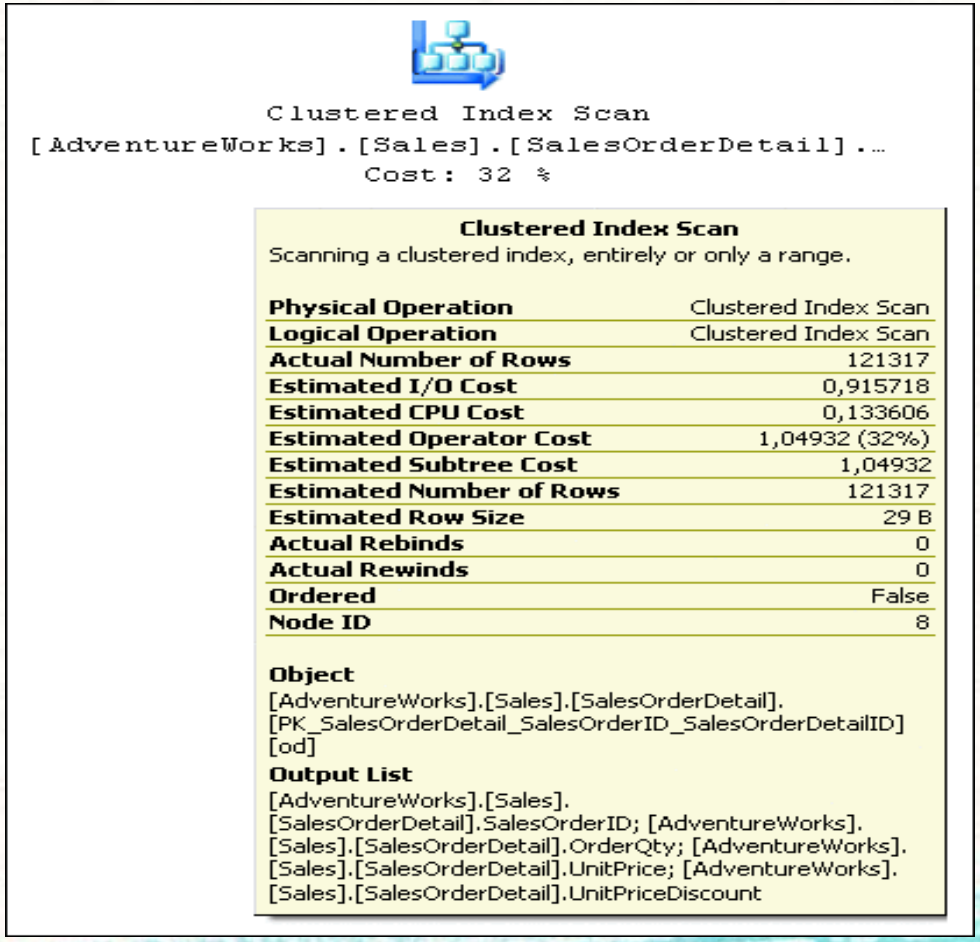
```
SELECT c.CustomerID, SUM(LineTotal)
FROM Sales.SalesOrderDetail od
JOIN Sales.SalesOrderHeader oh ON
od.SalesOrderID=oh.SalesOrderID
JOIN Sales.Customer c ON
oh.CustomerID=c.CustomerID
GROUP BY c.CustomerID;
```

Plan de execuție grafic



Plan de execuție grafic



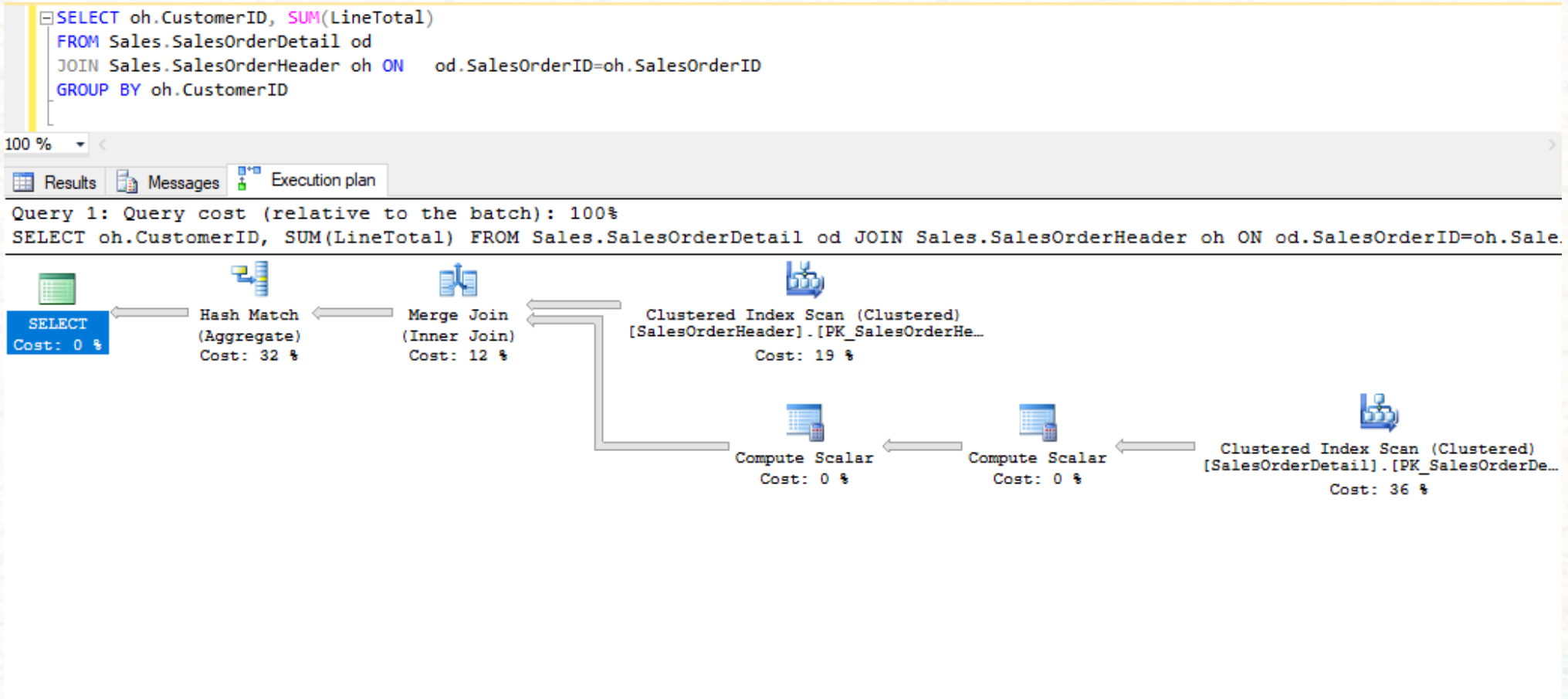


Plan de execuție grafic

- Exemplu:

```
SELECT oh.CustomerID, SUM(LineTotal)
FROM Sales.SalesOrderDetail od
JOIN Sales.SalesOrderHeader oh ON
od.SalesOrderID=oh.SalesOrderID
GROUP BY oh.CustomerID;
```

Plan de execuție grafic



Plan de execuție grafic

```
CREATE INDEX IDX_OrderDetail_OrderID_TotalLine
ON Sales.SalesOrderDetail(SalesOrderID)
INCLUDE (LineTotal);


SELECT oh.CustomerID, SUM(LineTotal)
FROM Sales.SalesOrderDetail od
JOIN Sales.SalesOrderHeader oh ON
od.SalesOrderID=oh.SalesOrderID
GROUP BY oh.CustomerID;
```

Plan de execuție grafic

- Dacă interogarea este executată din nou după crearea indexului, putem vedea că indexul este folosit: