

Seminar 5

SQL Server - *Tuning*-ul performanței -

Tuning-ul interogărilor - metodologie

- Identificarea așteptărilor (*bottleneck*) la nivel de server
 - I/O latches
 - Update log
 - Blocare
 - Altele
- Corelare așteptări - cozi (*queues*)
- Restrângere la nivel de bază de date / fișier
- Restrângere la nivel de proces
- *Tuning*-ul interogărilor problematice

Tuning-ul interogărilor - metodologie

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

Identificare aşteptări

- `sys.dm_os_wait_stats`:

- Tabel returnat:

- `wait_type`

- Aşteptări resursă (blocări, latches, reţea, I/O)
 - Aşteptări *queue*
 - Aşteptări externe

- `waiting_tasks_count`

- `wait_time_ms`

- `max_wait_time_ms`

- `signal_wait_time_ms`

- Resetarea *counter*-elor:

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

Corelare aşteptări - *queues*

- **sys.dm_os_performance_counters**
 - *object_name* - categoria *counter*-ului
 - *counter_name* - numele *counter*-ului
 - *instance_name* - numele instanţei specifice a *counter*-ului; adesea conţine numele bazei de date
 - *cntr_value* - valoarea curentă a *counter*-ului
 - *cntr_type* - tipul *counter*-ului definit de Performance Monitor

Corelare aşteptări - *queues*

- **sys.dm_os_performance_counters**
- > 500 countere: 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* conţine valoarea efectivă
- *cntr_type* = 537003264 → *cntr_value* conţine rezultate în timp real; trebuie împărţite la o "bază" pentru a obţine valoarea efectivă; singure, doar ele însele, sunt inutile ...
 - valoarea trebuie împărţită la o valoare "bază" pentru a obţine un raport; rezultatul se poate înmulţi cu 100.0 pentru a-l exprima în procente

Corelare aşteptări - *queues*

- **sys.dm_os_performance_counters**
- *cntr_type* = 272696576 → *cntr_value* conţine valoarea de bază
 - *counterele* sunt bazate pe timp
 - *counterele* sunt cumulative
 - se utilizează un tabel secundar pentru stocarea valorilor intermediare pentru statistici
- *cntr_type* = 1073874176 şi *cntr_type* = 1073939712
- → se obţine atât valoarea (1073874176) cât şi valoarea de bază (1073939712)
- se obţin ambele valori din nou (de ex, după 15 secunde) ☺
- pentru a obţine rezultatul dorit, se calculează:

$$\text{UnitatiPeSec} = (\text{cv2} - \text{cv1}) / (\text{bv2} - \text{bv1}) / 15.0$$

Restrângere la nivel de bază de date/fișier

■ `sys.dm_io_virtual_file_stats`

- returnează statistici I/O pentru fișierele de date și loguri

■ Parametri:

- `database_ID` (NULL = toate bazele de date), funcție utilă: `DB_ID`
- `file_ID` (NULL = toate fișierele), funcție utilă: `FILE_IDEX`

■ Tabel returnat:

- `database_ID`
- `file_ID`
- `sample_ms` - # de milisecunde de la pornirea calculatorului
- `num_of_reads` - numărul de citiri fizice realizate
- `num_of_bytes_read` - numărul de octeți citiți
- `io_stall_read_ms` - timpul total de așteptare al utilizatorilor pentru citiri
- `num_of_writes` - numărul de scrieri
- `num_of_bytes_written` - numărul total de octeți scriși
- `io_stall_write_ms` - timpul total de așteptare al utilizatorilor pentru finalizarea scrierilor
- `io_stall` - timpul total de așteptare al utilizatorilor pentru finalizarea operațiilor I/O (ms)
- `file_handle`

Restrângere la nivel de proces

- Filtrarea după durată/IO izolează doar procese individuale (batch/proc/interogare)
- E mai importantă agregarea datelor la nivel de șablon de interogare
 - Când se utilizează proceduri stocate, identificarea șablonului e ușoară
 - Când nu se utilizează proceduri stocate:
 - *Quick and dirty*: LEFT(query string, n)
 - Utilizarea unui *parser* pentru identificarea șablonului unei interogări

Indecși

- Sunt printre principalii factori care influențează performanța interogărilor
 - Impact asupra: filtrării, join-ului, sortării, grupării; pot evita blocările și dead-lock-ul etc
 - Efect în modificări: efect pozitiv în localizarea rândurilor; efect negativ - costul modificărilor în index
- Înțelegerea indecșilor și a mecanismelor interne ale acestora
 - Clustered/nonclustered, 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 optimizatorul să îi utilizeze pe primii pentru a acoperi o interogare
- *View-urile indexate* au un cost asociat de întreținere mai ridicat decât indecșii standard
 - opțiunea WITH SCHEMABINDING este obligatorie

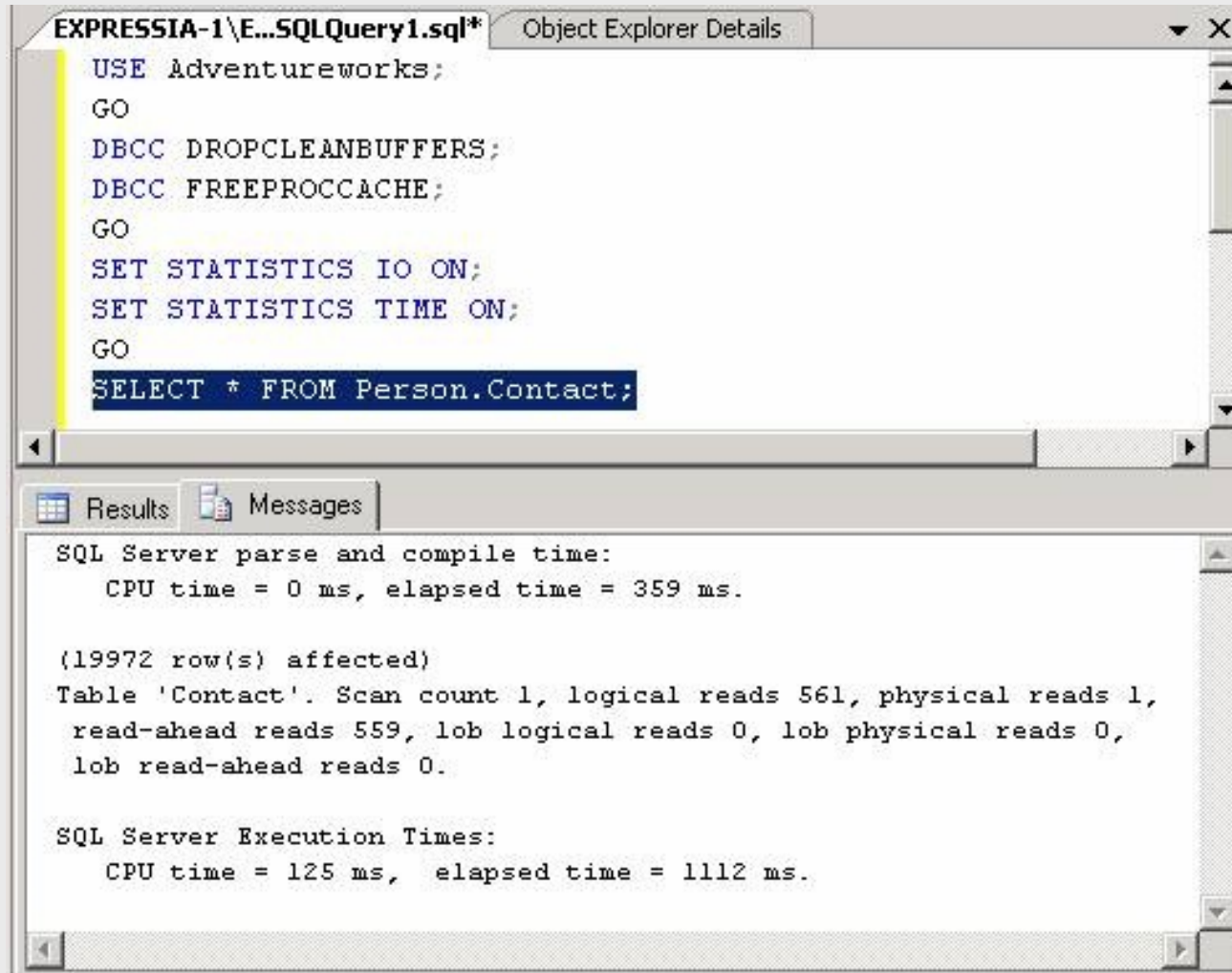
Analiza performanței interogărilor - unelte

- 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
- STATISTICS XML: informații detaliate legate de performanța efectivă în format XML
- SHOWPLAN_XML: informații detaliate legate de performanța estimată în format XML

Optimizarea interogărilor

- Evaluarea planurilor de execuție
 - Secvență de operații fizice/logice
- Factori - optimizare:
 - Predicatul de căutare utilizat
 - Tabelele implicate în join-uri
 - Condițiile de join
 - Dimensiunea rezultatului
 - Lista de indecși
- Scop: evitarea celor mai slabe planuri pentru interogări
- SQL Server utilizează un optimizator de interogări bazat pe cost

STATISTICS IO și STATISTICS TIME



The screenshot shows a SQL Server Enterprise Manager window titled 'EXPRESSIA-1\E...SQLQuery1.sql*'. The query window contains the following SQL code:

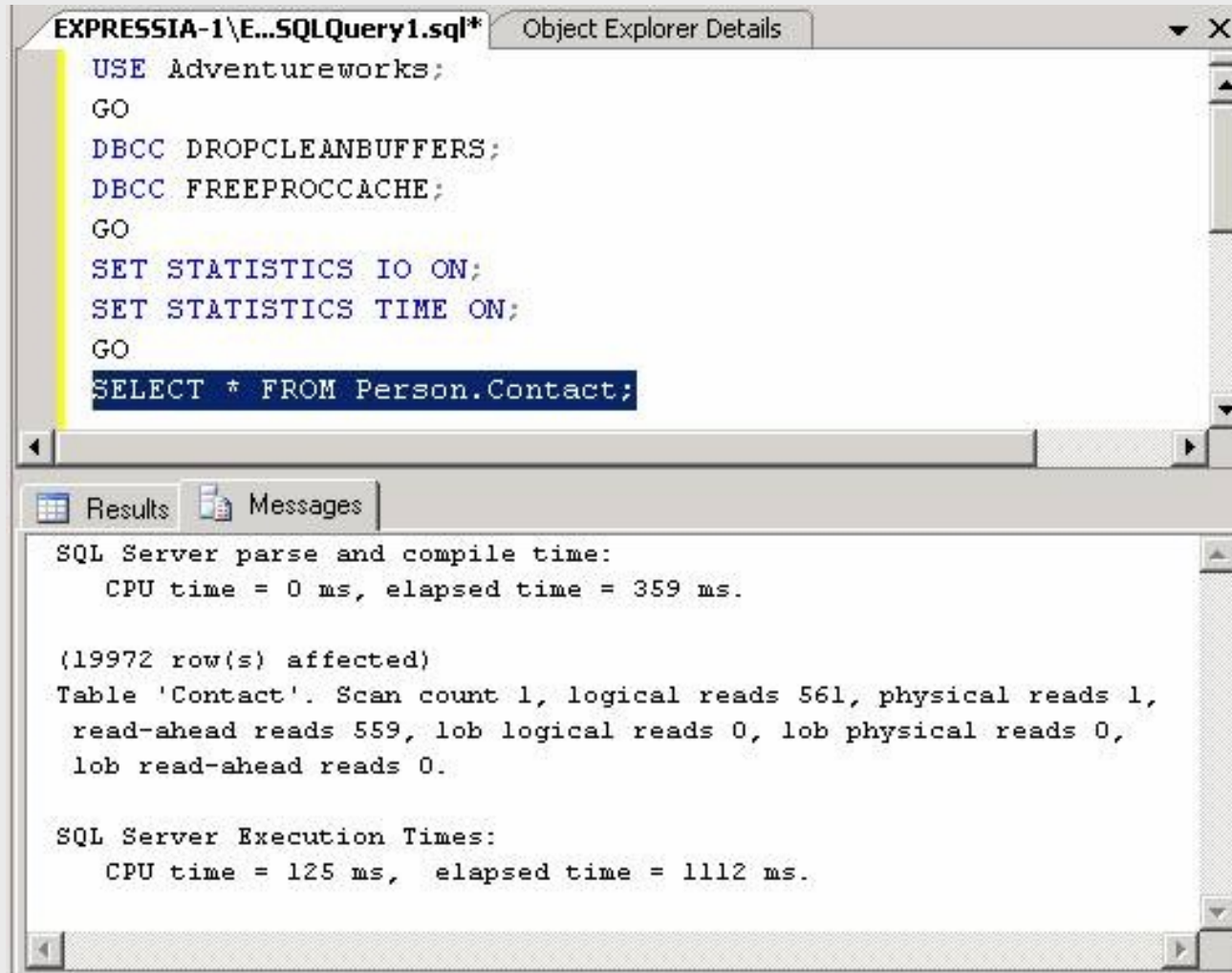
```
USE Adventureworks;  
GO  
DBCC DROPCLEANBUFFERS;  
DBCC FREEPROCCACHE;  
GO  
SET STATISTICS IO ON;  
SET STATISTICS TIME ON;  
GO  
SELECT * FROM Person.Contact;
```

The 'Results' tab is selected, displaying the following output:

```
SQL Server parse and compile time:  
    CPU time = 0 ms, elapsed time = 359 ms.  
  
(19972 row(s) affected)  
Table 'Contact'. Scan count 1, logical reads 561, physical reads 1,  
    read-ahead reads 559, lob logical reads 0, lob physical reads 0,  
    lob read-ahead reads 0.  
  
SQL Server Execution Times:  
    CPU time = 125 ms,  elapsed time = 1112 ms.
```

- DBCC DROPCLEANBUFFERS – golește datele SQL Server
- DBCC FREEPROCCACHE – golește cache-ul de planuri

STATISTICS IO și STATISTICS TIME



The screenshot shows a SQL Server Enterprise Manager window titled 'EXPRESSIA-1\E...SQLQuery1.sql*'. The query window contains the following SQL code:

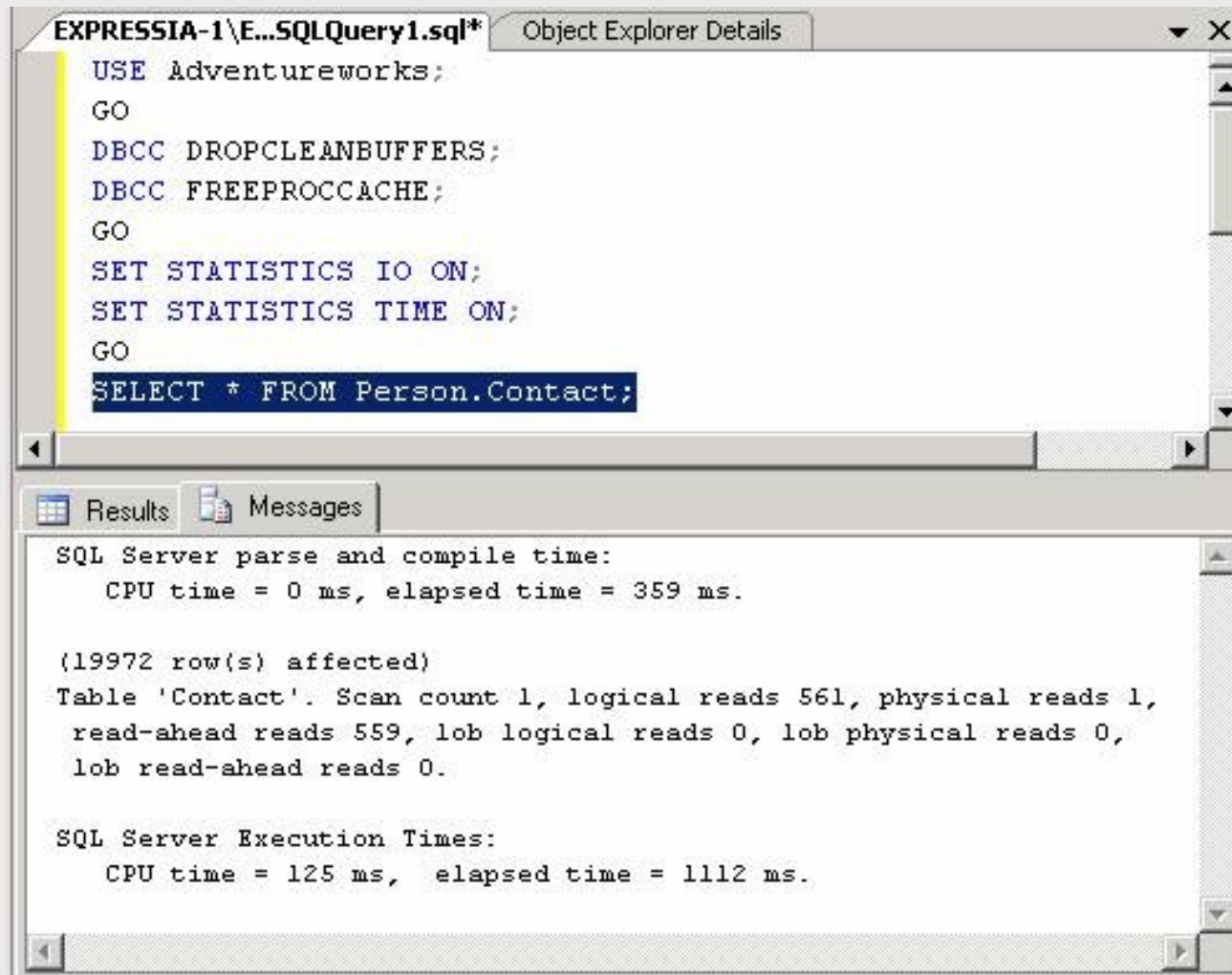
```
USE Adventureworks;  
GO  
DBCC DROPCLEANBUFFERS;  
DBCC FREEPROCCACHE;  
GO  
SET STATISTICS IO ON;  
SET STATISTICS TIME ON;  
GO  
SELECT * FROM Person.Contact;
```

The 'Results' tab is selected, displaying the following output:

```
SQL Server parse and compile time:  
    CPU time = 0 ms, elapsed time = 359 ms.  
  
(19972 row(s) affected)  
Table 'Contact'. Scan count 1, logical reads 561, physical reads 1,  
    read-ahead reads 559, lob logical reads 0, lob physical reads 0,  
    lob read-ahead reads 0.  
  
SQL Server Execution Times:  
    CPU time = 125 ms,  elapsed time = 1112 ms.
```

- *CPU time* – resursele CPU utilizate pentru a executa interogarea
- *elapsed time* – cât timp a durat execuția interogării

STATISTICS IO și STATISTICS TIME



The screenshot shows a SQL Server Enterprise Manager window titled 'EXPRESSIA-1\E...SQLQuery1.sql*'. The query window contains the following SQL code:

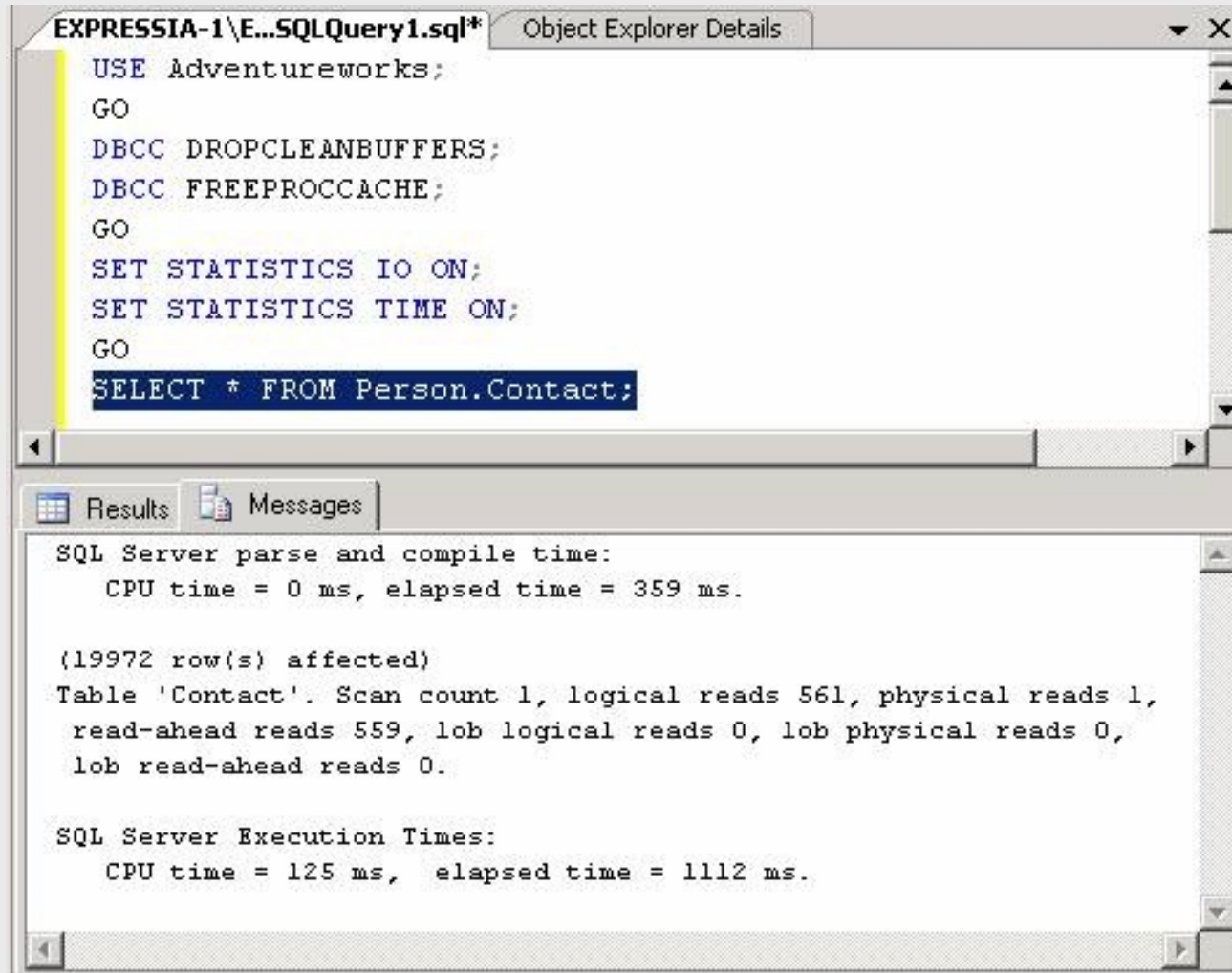
```
USE Adventureworks;  
GO  
DBCC DROPCLEANBUFFERS;  
DBCC FREEPROCCACHE;  
GO  
SET STATISTICS IO ON;  
SET STATISTICS TIME ON;  
GO  
SELECT * FROM Person.Contact;
```

The 'Results' tab is selected, displaying the following output:

```
SQL Server parse and compile time:  
    CPU time = 0 ms, elapsed time = 359 ms.  
  
(19972 row(s) affected)  
Table 'Contact'. Scan count 1, logical reads 561, physical reads 1,  
    read-ahead reads 559, lob logical reads 0, lob physical reads 0,  
    lob read-ahead reads 0.  
  
SQL Server Execution Times:  
    CPU time = 125 ms,  elapsed time = 1112 ms.
```

- *physical reads* - numărul de pagini citite de pe disc
- *read-ahead reads* - numărul de pagini plasate în cache pentru interogare 16

STATISTICS IO și STATISTICS TIME



The screenshot shows a SQL Server Enterprise Manager window titled 'EXPRESSIA-1\E...SQLQuery1.sql*'. The query window contains the following SQL code:

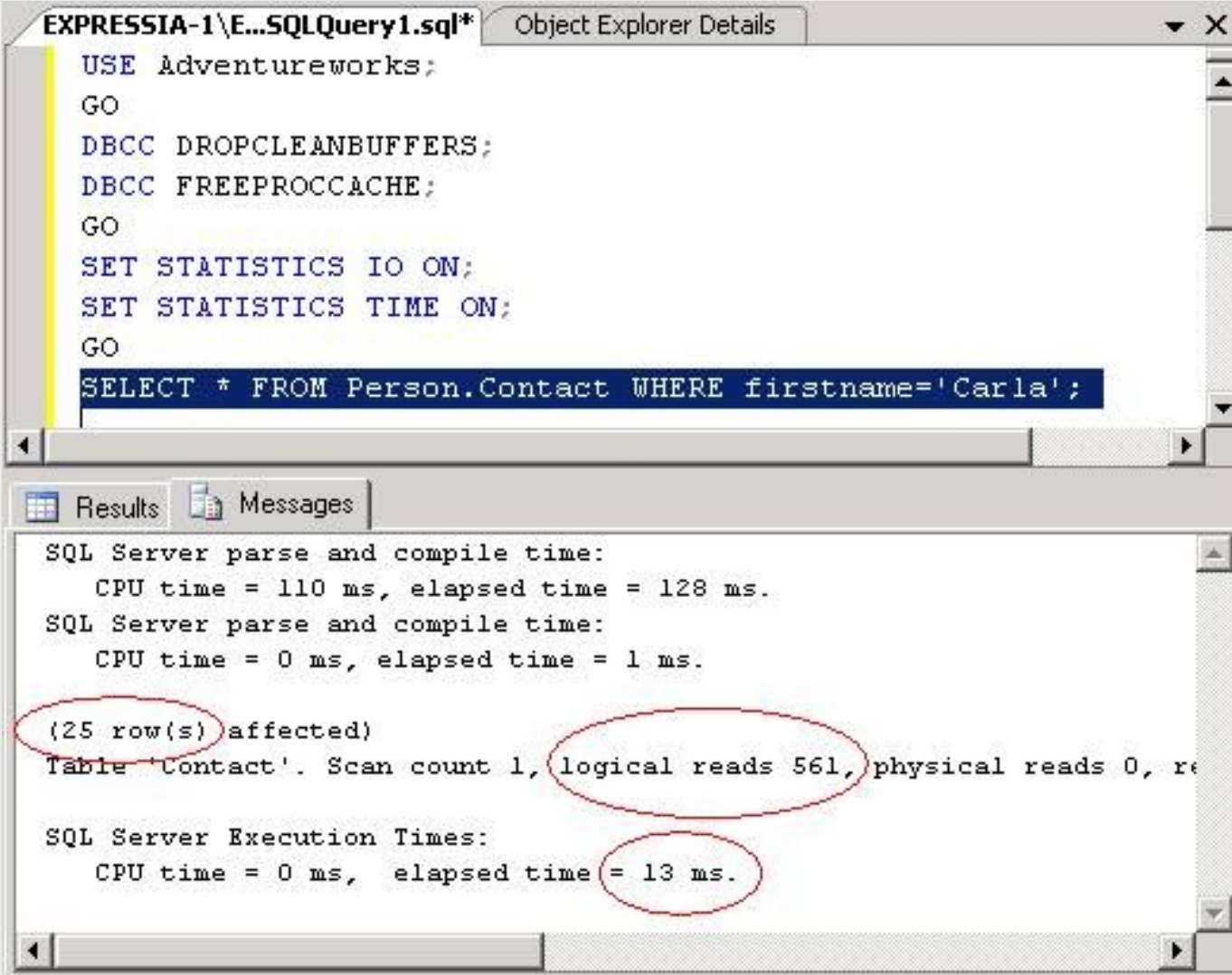
```
USE Adventureworks;  
GO  
DBCC DROPCLEANBUFFERS;  
DBCC FREEPROCCACHE;  
GO  
SET STATISTICS IO ON;  
SET STATISTICS TIME ON;  
GO  
SELECT * FROM Person.Contact;
```

The 'Results' tab is selected, displaying the following output:

```
SQL Server parse and compile time:  
    CPU time = 0 ms, elapsed time = 359 ms.  
  
(19972 row(s) affected)  
Table 'Contact'. Scan count 1, logical reads 561, physical reads 1,  
    read-ahead reads 559, lob logical reads 0, lob physical reads 0,  
    lob read-ahead reads 0.  
  
SQL Server Execution Times:  
    CPU time = 125 ms,  elapsed time = 1112 ms.
```

- *scan count* - de câte ori au fost accesate tabelele
- *logical reads* - numărul de pagini citite din data cache

STATISTICS IO și STATISTICS TIME



The screenshot displays the SQL Server Enterprise Manager interface. The top pane, titled 'EXPRESSIA-1\E...SQLQuery1.sql*', contains the following T-SQL script:

```
USE Adventureworks;  
GO  
DBCC DROPCLEANBUFFERS;  
DBCC FREEPROCCACHE;  
GO  
SET STATISTICS IO ON;  
SET STATISTICS TIME ON;  
GO  
SELECT * FROM Person.Contact WHERE firstname='Carla';
```

The bottom pane, titled 'Results', shows the execution output:

```
SQL Server parse and compile time:  
  CPU time = 110 ms, elapsed time = 128 ms.  
SQL Server parse and compile time:  
  CPU time = 0 ms, elapsed time = 1 ms.  
(25 row(s) affected)  
Table 'Contact'. Scan count 1, logical reads 561, physical reads 0, re  
  
SQL Server Execution Times:  
  CPU time = 0 ms, elapsed time = 13 ms.
```

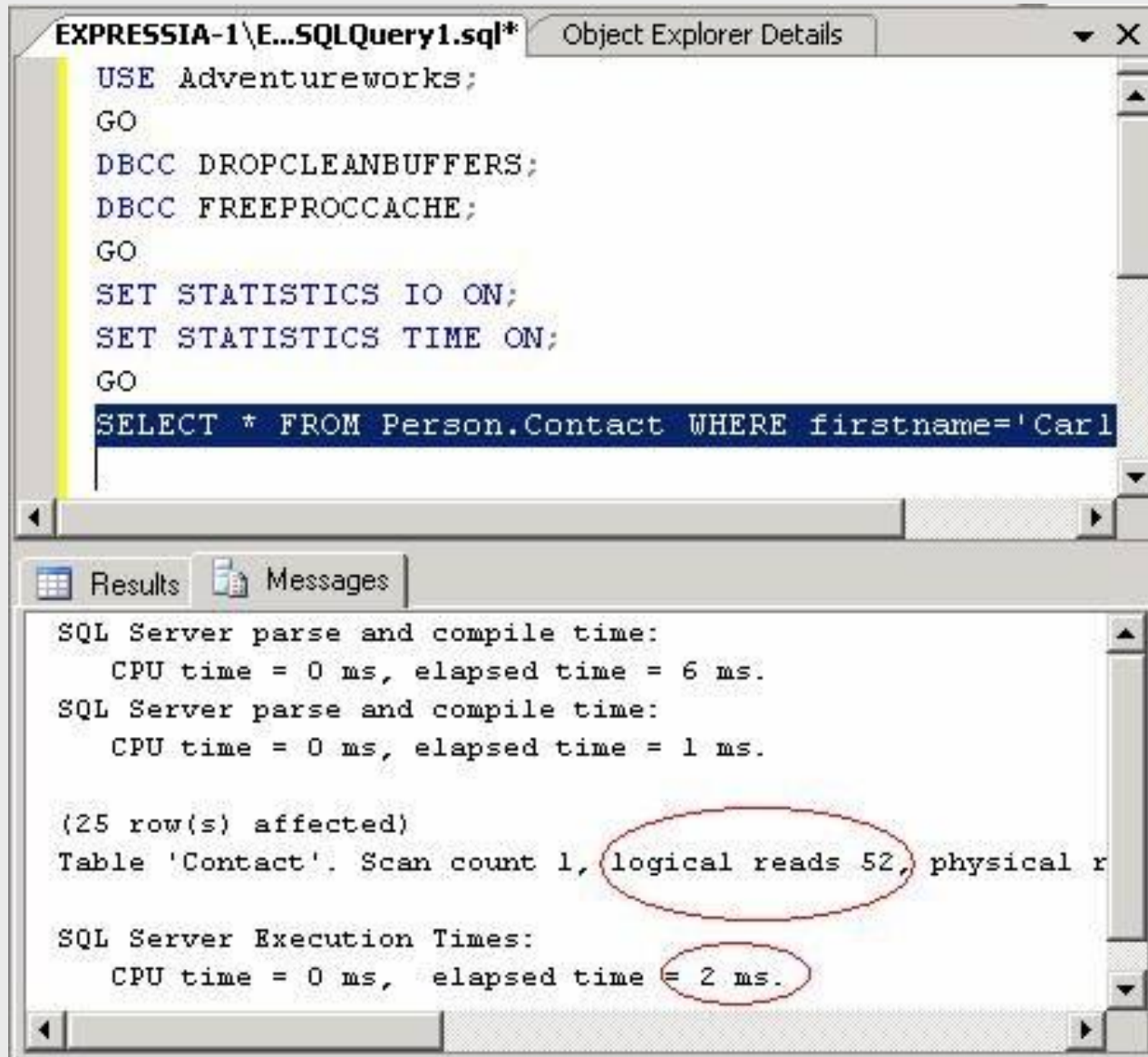
Red circles highlight the following values in the output:

- (25 row(s) affected)
- logical reads 561
- elapsed time = 13 ms.

STATISTICS IO *și* STATISTICS TIME

```
USE [AdventureWorks] GO
CREATE NONCLUSTERED INDEX [IDX_firstname]
ON [Person].[Contact]
(
    [FirstName] ASC
)
GO
```

STATISTICS IO *și* STATISTICS TIME



The screenshot shows a SQL Server Enterprise Manager window with a query editor and a results pane. The query editor contains the following SQL code:

```
USE Adventureworks;
GO
DBCC DROPLEANBUFFERS;
DBCC FREEPROCCACHE;
GO
SET STATISTICS IO ON;
SET STATISTICS TIME ON;
GO
SELECT * FROM Person.Contact WHERE firstname='Carl'
```

The results pane shows the execution statistics for the query:

```
SQL Server parse and compile time:
  CPU time = 0 ms, elapsed time = 6 ms.
SQL Server parse and compile time:
  CPU time = 0 ms, elapsed time = 1 ms.

(25 row(s) affected)
Table 'Contact'. Scan count 1, logical reads 52, physical r

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

The values "logical reads 52" and "elapsed time = 2 ms." are circled in red in the original image.

SHOWPLAN_ALL

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

Results

StmtText

```
-----  
SELECT COUNT(*) cRows  
FROM HumanResources.Shift;  
  |--Compute Scalar(DEFINE:([Expr1003]=CONVERT_IMPLICIT(int,[Expr1004],0)))  
  |--Stream Aggregate(DEFINE:([Expr1004]=Count(*)))  
    |--Index Scan(OBJECT:([master].[HumanResources].[Shift].[AK_Shift])  
  
(4 row(s) affected)
```

Plan de execuție grafic

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

Results Execution plan

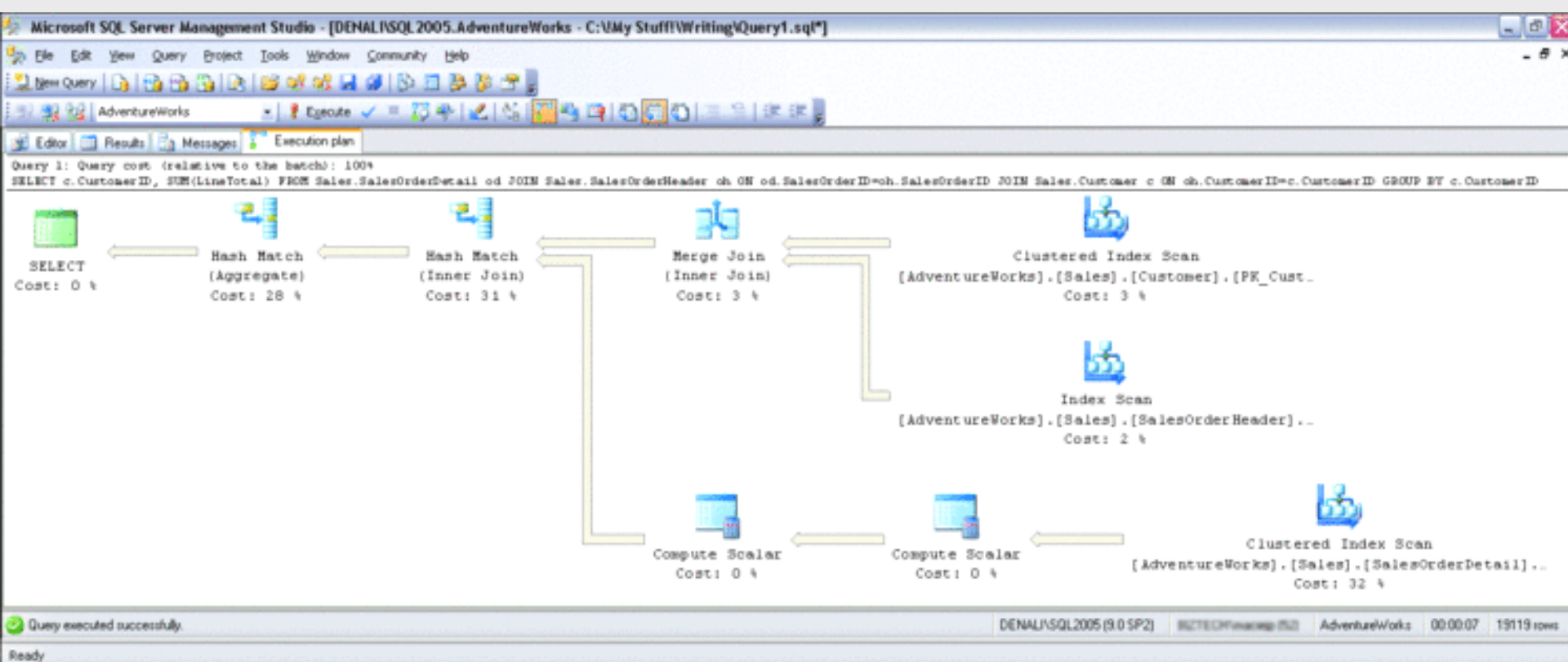
Query 1: Query cost (relative to the batch): 100%
SELECT COUNT(*) cRows FROM HumanResources.Shift;



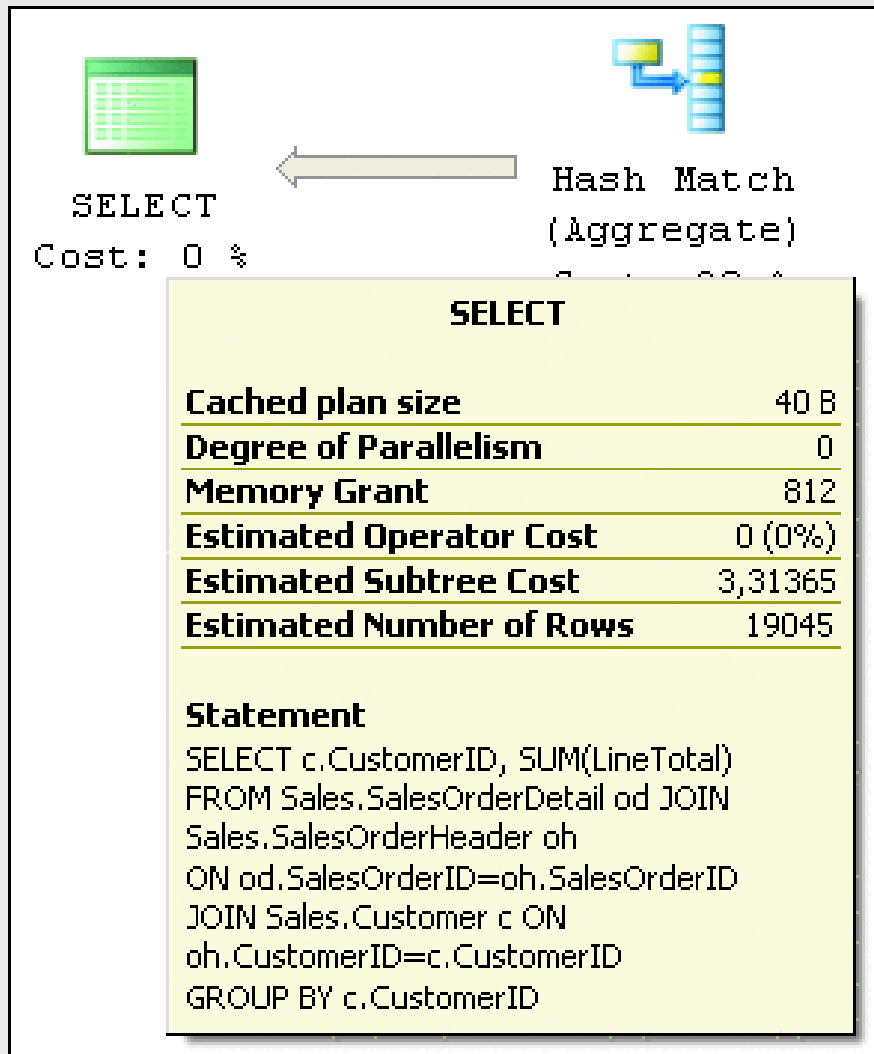

Plan de execuție grafic

```
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

Clustered Index Scan
[AdventureWorks].[Sales].[SalesOrderDetail] ...
Cost: 32 %

Clustered Index Scan	
Scanning a clustered index, entirely or only a range.	
Physical Operation	Clustered Index Scan
Logical Operation	Clustered Index Scan
Actual Number of Rows	121317
Estimated I/O Cost	0,915718
Estimated CPU Cost	0,133606
Estimated Operator Cost	1,04932 (32%)
Estimated Subtree Cost	1,04932
Estimated Number of Rows	121317
Estimated Row Size	29 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	False
Node ID	8

Object
[AdventureWorks].[Sales].[SalesOrderDetail].
[PK_SalesOrderDetail_SalesOrderID_SalesOrderDetailID]
[od]

Output List
[AdventureWorks].[Sales].
[SalesOrderDetail].SalesOrderID; [AdventureWorks].
[Sales].[SalesOrderDetail].OrderQty; [AdventureWorks].
[Sales].[SalesOrderDetail].UnitPrice; [AdventureWorks].
[Sales].[SalesOrderDetail].UnitPriceDiscount

Plan de execuție grafic

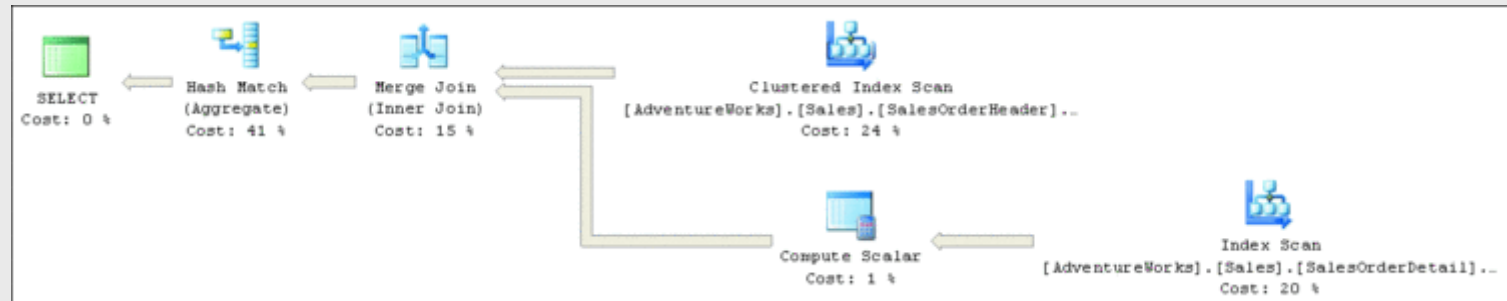
```
SELECT oh.CustomerID, SUM(LineTotal)
FROM Sales.SalesOrderDetail od
JOIN Sales.SalesOrderHeader oh ON
    od.SalesOrderID=oh.SalesOrderID
GROUP BY oh.CustomerID
```

Plan de execuție grafic



Plan de execuție grafic

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



Query Governor

- `SET QUERY_GOVERNOR_COST_LIMIT:` optimizatorul estimează numărul de secunde necesare pentru execuția unei interogări; dacă acesta depășește limita stabilită, interogarea nu va fi executată; dacă valoarea stabilită este 0, toate interogările vor fi rulate