SParser User Guide

Matthias Lüken

January 12, 2021

Contents

1 Quickstart

SParser parses SQL Server time and io statisics to a csv-format and provides commands for adding the corresponding tables to your LaTeX document. Here is how:

1. Add `\input{preamble/sparser.tex}` right before `\begin{document}`.
2. Copy and paste output from the messages tab in SSMS[1] into a text file.
3. Parse the file using the `\sparse{data.txt}` command.
4. You will find new text files in the same folder. Insert tables with...
   (a) time statistics via `\sparsertime{output_time.txt}`.

[1] Microsoft® SQL Server Management Studio

(b) io statistics via `\sparserio{output_io_b1s1.txt}` or compare two queries using `\sparserio{output_io_b1s1.txt}[output_io_b1s2.txt]`.

2 Detailed example

Let us take a look at the following query. The actual code is not important, but notice, that we have two batches (separated by `GO`-commands on lines 13 and 22). The first one consists of seven statements (three `SET`, two `PRINT` and two `SELECT` statements), the second one consists of three statements (two `PRINT` and one `SELECT`-statement).

Statistic	Batch	Statement
Time	Parse and compile time	Execution times
IO		Scan and read counts*
Count		Rows affected*

Table 1: Values collected by SSMS for a given statistic (time, io, count) and level (batch, statement). Notice that the number of affected rows is outputted by default and `SET NOCOUNT ON` has to be used to deactivate it. Values tagged with a * are only given when applicable.

```
1  SET NOCOUNT OFF
2  SET STATISTICS TIME ON
3  SET STATISTICS IO ON
4
5  PRINT '-----------------------'
6  PRINT 'First Batch'
7
8  SELECT *
9  FROM Sales.vSalesPerson
10
11 SELECT *
12 FROM Sales.vStoreWithAddresses
13 GO
14
15 PRINT '-----------------------'
16 PRINT 'Second Batch'
17
18 SELECT *
19 FROM Sales.vSalesPerson AS sp
20 JOIN Sales.vStoreWithAddresses AS swa
21         ON sp.PostalCode = swa.PostalCode
22 GO
```

When executing the query, SSMS will do its best to collect and output the statistics in the messages tab according to table 1, but the presentation is not exactly what one would call encouraging:

```
SQL Server parse and compile time:
   CPU time = 188 ms, elapsed time = 404 ms.

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

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

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

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.
First Batch

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

(17 rows affected)
Table 'PhoneNumberType'. Scan count 1, logical reads 35, physical reads 1,
↪  read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead
↪  reads 0.
Table 'PersonPhone'. Scan count 17, logical reads 34, physical reads 2, read-ahead
↪  reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'EmailAddress'. Scan count 17, logical reads 34, physical reads 2, read-ahead
↪  reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'SalesTerritory'. Scan count 1, logical reads 35, physical reads 1,
↪  read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead
↪  reads 0.
Table 'CountryRegion'. Scan count 0, logical reads 34, physical reads 3, read-ahead
↪  reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'StateProvince'. Scan count 0, logical reads 34, physical reads 3, read-ahead
↪  reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'Employee'. Scan count 0, logical reads 34, physical reads 2, read-ahead
↪  reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'Address'. Scan count 0, logical reads 34, physical reads 3, read-ahead reads
↪  0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'Person'. Scan count 0, logical reads 51, physical reads 3, read-ahead reads
↪  0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'BusinessEntityAddress'. Scan count 17, logical reads 34, physical reads 2,
↪  read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead
↪  reads 0.
Table 'SalesPerson'. Scan count 1, logical reads 2, physical reads 1, read-ahead
↪  reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 12 ms.

(712 rows affected)
Table 'Workfile'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads
↪  0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
```

```
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'Address'. Scan count 1, logical reads 216, physical reads 1, read-ahead reads 211, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'BusinessEntityAddress'. Scan count 6, logical reads 61, physical reads 1, read-ahead reads 48, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'AddressType'. Scan count 1, logical reads 2, physical reads 1, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'Store'. Scan count 1, logical reads 103, physical reads 1, read-ahead reads 101, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'StateProvince'. Scan count 1, logical reads 4, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'CountryRegion'. Scan count 1, logical reads 4, physical reads 1, read-ahead reads 16, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 298 ms.
SQL Server parse and compile time:
   CPU time = 15 ms, elapsed time = 23 ms.
-----------------------

 SQL Server Execution Times:
   CPU time = 0 ms,  elapsed time = 0 ms.
Second Batch

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

(41 rows affected)
Table 'Workfile'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'PersonPhone'. Scan count 41, logical reads 104, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'EmailAddress'. Scan count 41, logical reads 104, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'Store'. Scan count 1, logical reads 103, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'BusinessEntityAddress'. Scan count 633, logical reads 1464, physical reads 1, read-ahead reads 48, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'Address'. Scan count 1, logical reads 250, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'Person'. Scan count 0, logical reads 51, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'SalesPerson'. Scan count 1, logical reads 2, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'Employee'. Scan count 1, logical reads 9, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'StateProvince'. Scan count 2, logical reads 8, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
```

```
72  Table 'CountryRegion'. Scan count 2, logical reads 8, physical reads 0, read-ahead
    ↪  reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
73  Table 'AddressType'. Scan count 1, logical reads 2, physical reads 0, read-ahead
    ↪  reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
74  Table 'SalesTerritory'. Scan count 1, logical reads 2, physical reads 0, read-ahead
    ↪  reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
75  Table 'PhoneNumberType'. Scan count 1, logical reads 2, physical reads 0,
    ↪  read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead
    ↪  reads 0.
76
77   SQL Server Execution Times:
78     CPU time = 16 ms,  elapsed time = 45 ms.
```

2.1 sparse

Now, let us save the above contents of the messages tab to a file `statistics/query1.txt` and add these commands to the LaTeX document:

- `\input{preamble/sparser.tex}` after loading all packages
- `\sparse{statistics/query1.txt}` anywhere

After typesetting, four new files will appear in the `statistics`-folder:

- `query1_io_b1s6.txt`
- `query1_io_b1s7.txt`
- `query1_io_b2s3.txt`
- `query1_time.txt`

The file with the `time`-suffix holds all time and also count statistics, files with an `io`-infix contain io statistics, where `b`x`s`y indicates the x-th batch and the y-th statement.

2.2 sparsertime

To get an overview, let us start with the last file by inserting the following code:

```
1  \begin{table}
2  \sparsertime{statistics/query1_time.txt}[1]
3  \end{table}
```

The result is shown in table 2. Columns one to three hold the parse and compile times of the two batches. Columns four to seven contain information about the effected rows and execution times for each statement. The optional second parameter tells SParser to skip all statements with zero affected rows and execution times. In this example, that is why only three statements are shown, corresponding to the `SELECT` statements from our query.

Batch	Parse and compile time		Stmt	Rows	Execution Times	
	CPU [ms]	elapsed [ms]			CPU [ms]	elapsed [ms]
1	188	404	6	17	0	12
			7	712	0	298
2	15	23	3	41	16	45
Total	203	427		770	16	355

Table 2: Time and count statistics. Statements are skipped, when the number of affected rows and execution times are zero.

2.3 sparserio

Next up are the io statistics. To display the data from the sixth statement in the first batch (see table 3), use the `\sparserio` command as shown. The first optional parameter truncates the table names in the first column to the given number of characters, to avoid overflowing table cells. Table names are also allowed to wrap anywhere, when too long.

```
\begin{table}
\sparserio[14]{statistics/query1_io_b1s6.txt}
\end{table}
```

Often, for example when doing performance optimization, one would like to compare two similar statements. That is why `\sparserio` has three more optional arguments:

Table	Scans	Reads			LOB-Reads		
		logical	physical	r.-ahead	logical	physical	r.-ahead
Address	0	34	3	0	0	0	0
BusinessEn...	17	34	2	0	0	0	0
CountryRegion	0	34	3	0	0	0	0
EmailAddress	17	34	2	0	0	0	0
Employee	0	34	2	0	0	0	0
Person	0	51	3	0	0	0	0
PersonPhone	17	34	2	0	0	0	0
PhoneNumbe...	1	35	1	0	0	0	0
SalesPerson	1	2	1	0	0	0	0
SalesTerritory	1	35	1	0	0	0	0
StateProvince	0	34	3	0	0	0	0
Total	54	361	23	0	0	0	0

Table 3: Scan and read counts for statement six in batch one.

```
\begin{table}
\sparserio{statistics/query1_io_b2s3.txt}[statistics/query1_io_b1s7.txt][1][2]
\end{table}
```

All odd rows of table 4 belong to the first file, all even rows to the second file. For tables, that appear in only one of the two files, a row of zeros is added. The third parameter lets you skip tables, when all numbers are identical in both files (here Workfile and Worktable). The value of the fourth parameter controlls the intensity of the color highlighting: if the number of scans or reads in the second file is lower/higher than in the first one, the table cell has a green/red background color, where the opacity depends on the relative difference.

3 Command reference

`\sparse{<file>}[<debug-out>]`

Parses a text file with time and io statistics from SSMS into multiple csv files in the same folder. Parameters:

- `<file>` (string, mandatory): text file with the messages tab contents from SSMS
- `<debug-out>` (0 or 1, optional): see section 4.5 for details

`\sparserio[<chars>]{<file1>}[<file2>][<skip>][<color>]`

Creates a table holding scan and read counts from statistics io. Parameters:

- `<chars>` (integer, optional): truncates table names in the first column to the given number of characters, where 0 means no truncation. If `<chars>` is larger than four and the table name has more than `<chars>` characters, the table name is truncated to `<chars>`-4 characters and an ellipsis ("...") is appended. For reference, see the function `\sparser_breakanywhere:Nn`.
- `<file1>` (string, mandatory): text file with the io data produced by `\sparse`
- `<file2>` (string, optional): another text file with the io data produced by `\sparse`. Providing this parameter switches the presentation to a table where the data from both files is interweaved.
- `<skip>` (0 or 1, optional): if active, the data from both files is compared for every table. When both rows are identical, the table is skipped, so only tables appear, where an actual change has occured. The totals in the last row remain unaffected, so be aware, that these numbers not necessarily match the sums of the data shown.
- `<color>` (non-negative float, optional): intensity of the color highlighting, where 0 deactivates coloring. When the value of the second file is larger (smaller) than the value of the first one, the background color is set to `spred!`p (`spdarkgreen!`p), where p is the percentage the color is mixed with white according to the `xcolor`

package. If you want to customize the colors used, see section 4.3. The actual value of p is computed by the `\sparser_colorscale:nnnnn` function as follows:

$$p(x_1, x_2, s_1, s_2, c) = \left[100 \cdot \left(\frac{|x_1 - x_2|}{\max(s_1, s_2)} \right)^{1/c} \right],$$

where

- x_1, x_2 are the scan or read counts of a specific table for `file1` and `file2`, respectively.
- s_1, s_2 are the sums of all scans and counts over all tables for `file1` and `file2`, respectively.
- c is the `<color>` parameter given to `\sparserio`.

`\sparsertime{file}[skip]`

Creates a table with an overview of all batches and statements including time statistics and the number of rows affected. Parameters:

- `<file>` (string, mandatory): text file with the time data produced by `\sparse`
- `<skip>` (0 or 1, optional): if active, statements with zero execution times and no rows affected will be skipped.

4 Loose ends and customization

4.1 Languages

Sparser is capable of multilangual support albeit only english and german in- and output is supported at the moment. If you wish to contribute to SParser or customize your version, use this guide to get started:

Input: the `\sparse` command calls the `\sparser_readdata:nn` function, which scans the input file line by line. After stripping all numbers from a line and taking the last 105 characters, the result is compared to a given set of key phrases. There are five kinds of phrases (compile time header, execution time header, io statistics, rows affected, compile or execution time data) and upon matching one of those, the function `sparser_fsm:nn` is called. To add a new translation, just copy and paste the respective line and replace the key phrase in the first set of braces.

```
{
        % compile time header
        {SQL Server parse and compile time:} {\sparser_fsm:nn {comp}{0}}
        {SQL Server-Analyse- und Kompilierzeit:} {\sparser_fsm:nn {comp}{0}}
        % execution time header
        {SQL Server Execution Times:} {\sparser_fsm:nn {exec}{0}}
        {SQL Server-Ausführungszeiten:} {\sparser_fsm:nn {exec}{0}}
```

```
        % io statistics
        {Scan count , logical reads , physical reads , read-ahead reads , lob
        ↪  logical reads , lob physical reads , lob read-ahead reads.}
        ↪  {\sparser_fsm:nn {io}{1}}
        {e , Read-Ahead-Lesevorgänge , logische LOB-Lesevorgänge , physische
        ↪  LOB-Lesevorgänge , Read-Ahead-LOB-Lesevorgänge.} {\sparser_fsm:nn
        ↪  {io}{1}}
        % rows affected
        {( rows affected)} {\sparser_fsm:nn {rows}{1}}
        {( Zeilen betroffen)} {\sparser_fsm:nn {rows}{1}}
        {( Zeile betroffen)} {\sparser_fsm:nn {rows}{1}}
        % compile or execution time data
        {, CPU-Zeit = ms, verstrichene Zeit = ms.} {\sparser_fsm:nn {}{1}}
        {CPU time = ms, elapsed time = ms.} {\sparser_fsm:nn {}{1}}
}
```

Output: SParser uses the `\sparser_dict:n` function to output table headers and the "Total" in the last row by checking the value of `\languagename` (which is automatically set to the desired language when loading the `babel` package). Just add your `{language}{translation}`-pair to the select case-heap.

4.2 Number format

All numbers are outputted using the `\num` command available in the `siunitx`-package. For example, use

```
\usepackage[group-minimum-digits = 3, group-separator={,}]{siunitx}
```

to increase readability for large numbers. If the command is not defined before the Sparser tool is loaded, `\num` will be set equal to `\relax`.

4.3 Colors

The color highlighting in the `\sparserio` command and the debug out use the following color names:

- sporange
- spdarkgreen
- spgray
- spred

You can overwrite these colors at any time using `\definecolor`, otherwise default values are used.

4.4 Dependencies

SParser loads the following packages: `xparse`, `xstring`, `xcolor`, `datatool`, `booktabs`, `tabularx`, `colortbl` and `multirow`.

4.5 Debugging

If the `\sparse` command ever shows unexpected behavior and you would like to investigate further, use the second optional parameter to activate a debug output:

```
1  \sparse{statistics/query1.txt}[1]
```

For every line in the input file, information about the state of the parser is given:

- `b`: batch counter
- `s`: statement counter
- `a`: active flag. 1 signals that data is accepted to be buffered or written, 0 means no data is handled.
- `state`: reading a key phrase can cause the parser to switch into another state depending on the previous state. The parser remains in the last state until a new valid key phrase is recognized. Possible values are: `init`, `comp`, `exec`, `io`, `rows`
- `data`: input is recognized and valid, `ignored`: data is recognized but somehow invalid (for example when the parser enters an unexpected state), `no match`: input not recognized
- `written`: data has been written to file, `buffered`: data stored in internal buffer (for example when additional data is required for a complete row).

Here is the output for the example from before:

```
parsing statistics/query1.txt...
opening time output stream statistics/query1_time.txt...
b:1 s:0 a:0 state:comp SQL Server parse and compile tim
b:1 s:0 a:1 state:comp data: 1,188,404 buffered
b:1 s:0 a:0 state:comp no match: \par
b:1 s:1 a:0 state:exec SQL Server Execution Times:
b:1 s:1 a:1 state:exec data: 1,0,0,0 written
b:1 s:1 a:0 state:exec no match: \par
b:1 s:2 a:0 state:exec SQL Server Execution Times:
b:1 s:2 a:1 state:exec data: 2,0,0,0 written
b:1 s:2 a:0 state:exec no match: \par
b:1 s:3 a:0 state:exec SQL Server Execution Times:
b:1 s:3 a:1 state:exec data: 3,0,0,0 written
b:1 s:3 a:0 state:exec no match: ----------------------
b:1 s:3 a:0 state:exec no match: \par
b:1 s:4 a:0 state:exec SQL Server Execution Times:
```

```
b:1 s:4 a:1 state:exec data: 4,0,0,0 written
b:1 s:4 a:0 state:exec no match: FirstBatch
b:1 s:4 a:0 state:exec no match: \par
b:1 s:5 a:0 state:exec SQL Server Execution Times:
b:1 s:5 a:1 state:exec data: 5,0,0,0 written
b:1 s:5 a:0 state:exec no match: \par
b:1 s:6 a:1 state:rows data: 17 buffered
opening io output stream statistics/query1_io_b1s6.txt...
b:1 s:6 a:1 state:io.. data: PhoneNumberT,1,35,1,0,0,0,0, written
b:1 s:6 a:1 state:io.. data: PersonPhone,17,34,2,0,0,0,0, written
b:1 s:6 a:1 state:io.. data: EmailAddress,17,34,2,0,0,0,0, written
b:1 s:6 a:1 state:io.. data: SalesTerrito,1,35,1,0,0,0,0, written
b:1 s:6 a:1 state:io.. data: CountryRegio,0,34,3,0,0,0,0, written
b:1 s:6 a:1 state:io.. data: StateProvinc,0,34,3,0,0,0,0, written
b:1 s:6 a:1 state:io.. data: Employee,0,34,2,0,0,0,0, written
b:1 s:6 a:1 state:io.. data: Address,0,34,3,0,0,0,0, written
b:1 s:6 a:1 state:io.. data: Person,0,51,3,0,0,0,0, written
b:1 s:6 a:1 state:io.. data: BusinessEnti,17,34,2,0,0,0,0, written
b:1 s:6 a:1 state:io.. data: SalesPerson,1,2,1,0,0,0,0, written
b:1 s:6 a:0 state:io.. no match: \par
b:1 s:6 a:0 state:exec SQL Server Execution Times:
b:1 s:6 a:1 state:exec data: 6,17,0,12 written
b:1 s:6 a:0 state:exec no match: \par
b:1 s:7 a:1 state:rows data: 712 buffered
...closing io output stream statistics/query1_io_b1s6.txt
opening io output stream statistics/query1_io_b1s7.txt...
b:1 s:7 a:1 state:io.. data: Workfile,0,0,0,0,0,0,0, written
b:1 s:7 a:1 state:io.. data: Worktable,0,0,0,0,0,0,0, written
b:1 s:7 a:1 state:io.. data: Address,1,216,1,211,0,0,0, written
b:1 s:7 a:1 state:io.. data: BusinessEnti,6,61,1,48,0,0,0, written
b:1 s:7 a:1 state:io.. data: AddressType,1,2,1,0,0,0,0, written
b:1 s:7 a:1 state:io.. data: Store,1,103,1,101,0,0,0, written
b:1 s:7 a:1 state:io.. data: StateProvinc,1,4,0,0,0,0,0, written
b:1 s:7 a:1 state:io.. data: CountryRegio,1,4,1,16,0,0,0, written
b:1 s:7 a:0 state:io.. no match: \par
b:1 s:7 a:0 state:exec SQL Server Execution Times:
b:1 s:7 a:1 state:exec data: 7,712,0,298 written
b:2 s:0 a:0 state:comp SQL Server parse and compile tim
b:2 s:0 a:1 state:comp data: 2,15,23 buffered
b:2 s:0 a:0 state:comp no match: -----------------------
b:2 s:0 a:0 state:comp no match: \par
b:2 s:1 a:0 state:exec SQL Server Execution Times:
b:2 s:1 a:1 state:exec data: 1,0,0,0 written
b:2 s:1 a:0 state:exec no match: SecondBatch
b:2 s:1 a:0 state:exec no match: \par
b:2 s:2 a:0 state:exec SQL Server Execution Times:
b:2 s:2 a:1 state:exec data: 2,0,0,0 written
b:2 s:2 a:0 state:exec no match: \par
b:2 s:3 a:1 state:rows data: 41 buffered
```

```
...closing io output stream statistics/query1_io_b1s7.txt
opening io output stream statistics/query1_io_b2s3.txt...
b:2 s:3 a:1 state:io.. data: Workfile,0,0,0,0,0,0,0, written
b:2 s:3 a:1 state:io.. data: Worktable,0,0,0,0,0,0,0, written
b:2 s:3 a:1 state:io.. data: PersonPhone,41,104,0,0,0,0,0, written
b:2 s:3 a:1 state:io.. data: EmailAddress,41,104,0,0,0,0,0, written
b:2 s:3 a:1 state:io.. data: Store,1,103,0,0,0,0,0, written
b:2 s:3 a:1 state:io.. data: BusinessEnti,633,1464,1,48,0,0,0, written
b:2 s:3 a:1 state:io.. data: Address,1,250,0,0,0,0,0, written
b:2 s:3 a:1 state:io.. data: Person,0,51,0,0,0,0,0, written
b:2 s:3 a:1 state:io.. data: SalesPerson,1,2,0,0,0,0,0, written
b:2 s:3 a:1 state:io.. data: Employee,1,9,0,0,0,0,0, written
b:2 s:3 a:1 state:io.. data: StateProvinc,2,8,0,0,0,0,0, written
b:2 s:3 a:1 state:io.. data: CountryRegio,2,8,0,0,0,0,0, written
b:2 s:3 a:1 state:io.. data: AddressType,1,2,0,0,0,0,0, written
b:2 s:3 a:1 state:io.. data: SalesTerrito,1,2,0,0,0,0,0, written
b:2 s:3 a:1 state:io.. data: PhoneNumberT,1,2,0,0,0,0,0, written
b:2 s:3 a:0 state:io.. no match: \par
b:2 s:3 a:0 state:exec SQL Server Execution Times:
b:2 s:3 a:1 state:exec data: 3,41,16,45 written
...closing io output stream statistics/query1_io_b2s3.txt
...closing time output stream statistics/query1_time.txt
...finished parsing statistics/query1.txt
```

Table	**Scans**	**Reads**			**LOB-Reads**		
		logical	physical	r.-ahead	logical	physical	r.-ahead
Address	1	250	0	0	0	0	0
	1	216	1	211	0	0	0
AddressType	1	2	0	0	0	0	0
	1	2	1	0	0	0	0
BusinessEntity	633	1,464	1	48	0	0	0
Address	6	61	1	48	0	0	0
CountryRegion	2	8	0	0	0	0	0
	1	4	1	16	0	0	0
EmailAddress	41	104	0	0	0	0	0
	0	0	0	0	0	0	0
Employee	1	9	0	0	0	0	0
	0	0	0	0	0	0	0
Person	0	51	0	0	0	0	0
	0	0	0	0	0	0	0
PersonPhone	41	104	0	0	0	0	0
	0	0	0	0	0	0	0
PhoneNumber	1	2	0	0	0	0	0
Type	0	0	0	0	0	0	0
SalesPerson	1	2	0	0	0	0	0
	0	0	0	0	0	0	0
SalesTerritory	1	2	0	0	0	0	0
	0	0	0	0	0	0	0
StateProvince	2	8	0	0	0	0	0
	1	4	0	0	0	0	0
Store	1	103	0	0	0	0	0
	1	103	1	101	0	0	0
Total	726	2,109	1	48	0	0	0
	11	390	5	376	0	0	0

Table 4: Comparison of io statistics from two different queries.