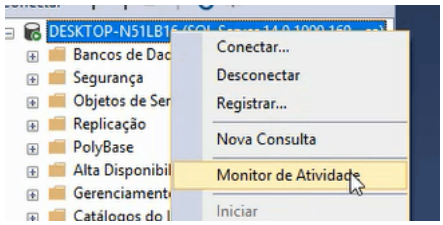
10

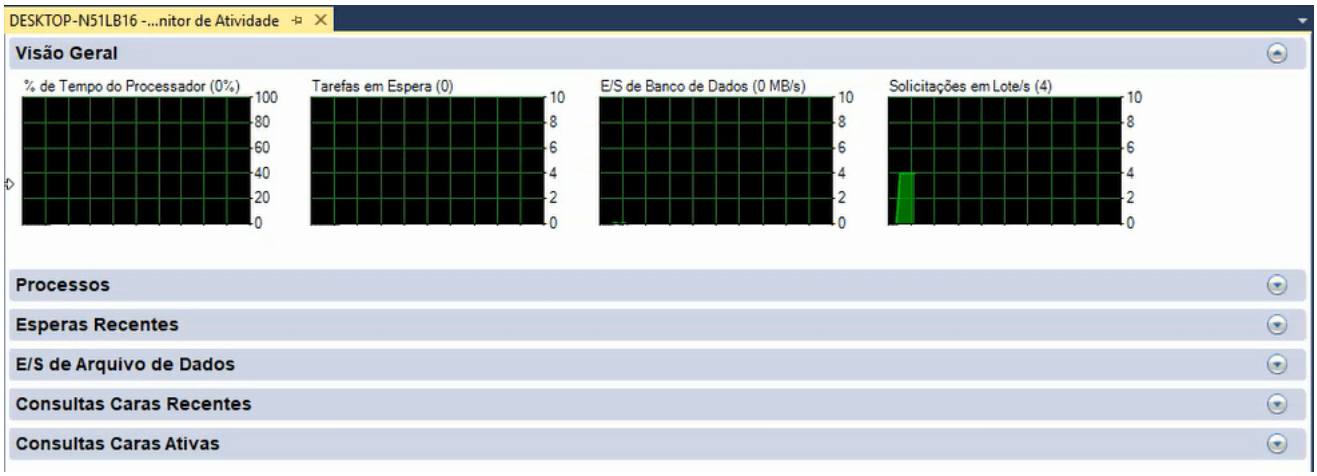
Mãos na massa

Chegou a hora de você pôr em prática o que foi visto na aula. Para isso, execute os passos listados abaixo.

1) Sobre o nome do servidor, clique com botão da direita do mouse e selecione a opção **Monitor de Atividade**:



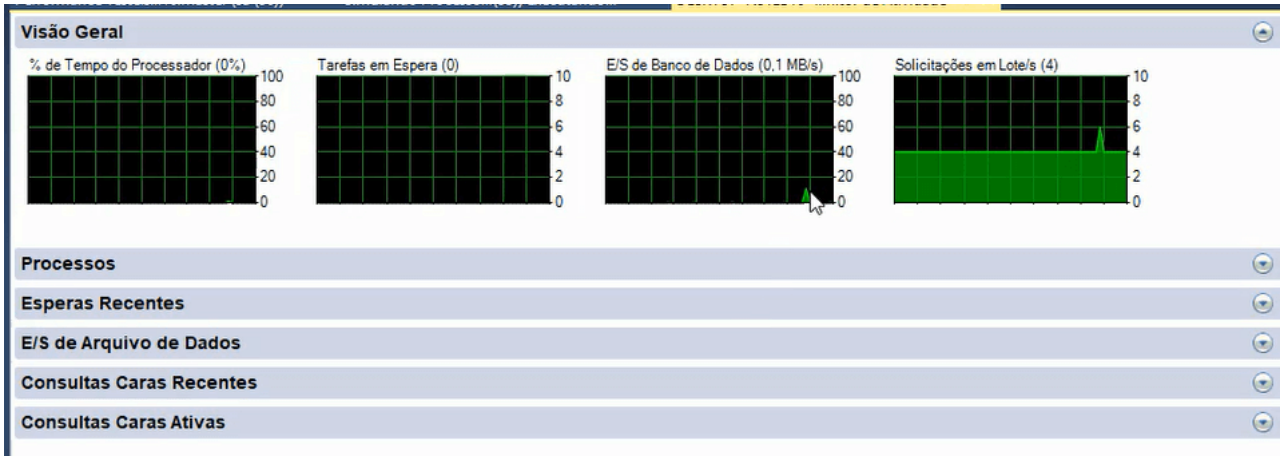
2) A janela abaixo mostra as atividades no servidor do SQL Server:



3) Abra o arquivo **Simulando Processos.sql** (caso você ainda não o tenha baixado, baixe-o aqui (https://caelum-online-public.s3.amazonaws.com/839-administracao-do-sql-server-2017/04/Simulando%20Processos.sql)) e execute os comandos abaixo:

```
USE SUCOS_VENDAS;

IF OBJECT_ID(' dbo.Nums2', 'U') IS NOT NULL DROP TABLE dbo.Nums2;
CREATE TABLE dbo.Nums2( n float );

DECLARE @max AS INT, @rc AS INT;
SET @max = 3600;
SET @rc = 1;
WHILE @rc < = @max
BEGIN
    DECLARE @X AS FLOAt
    SELECT @X = SUM(QUANTIDADE) FROM [ITENS NOTAS FISCAIS]
    INSERT INTO dbo.Nums2 (n) values (@X);
    SET @rc = @rc + 1;
    WAITFOR DELAY '00:00:02';
END
```

4) Enquanto estes comandos estão sendo executados, observe o monitor. Pouca atividade foi incluída no monitoramento:

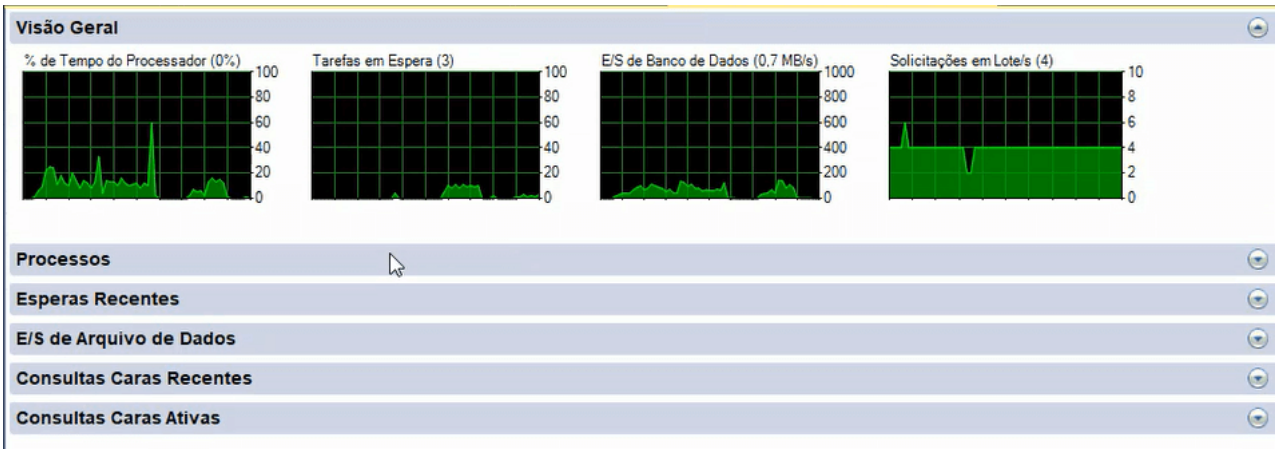


5) Agora, execute outros comandos do arquivo **Performance Teste.sql** (caso você ainda não o tenha baixado, baixe-o aqui (https://caelum-online-public.s3.amazonaws.com/839-administracao-do-sql-server-2017/04/Performance%20Teste.sql)):

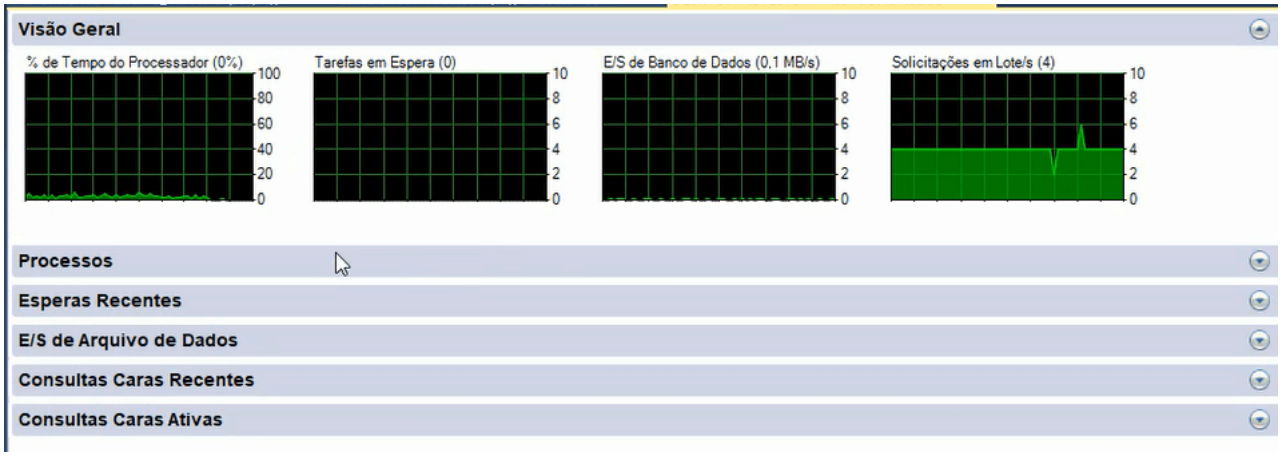
```
SET NOCOUNT ON;

USE SUCOS_VENDAS;

IF OBJECT_ID(' dbo.Nums', 'U') IS NOT NULL DROP TABLE dbo.Nums;
CREATE TABLE dbo.Nums( n INT NOT NULL PRIMARY KEY);
DECLARE @max AS INT, @rc AS INT;
SET @max = 30000000;
SET @rc = 1;
INSERT INTO Nums VALUES( 1);
WHILE @rc * 2 < = @max
BEGIN
    INSERT INTO dbo.Nums SELECT n + @rc FROM dbo.Nums;
    SET @rc = @rc * 2;
END

INSERT INTO dbo.Nums SELECT n + @rc FROM dbo.Nums WHERE n + @rc < = @max;

SELECT n, n +  1 AS n_plus_one FROM dbo.Nums WHERE n < = 30000000;
```

Estes são comandos mais "pesados", que irão influenciar no resultado do monitor.

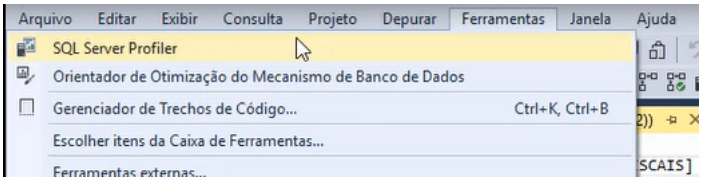
6) Durante a execução do segundo conjunto de comandos, você vê que o resultado do monitor é alterado:



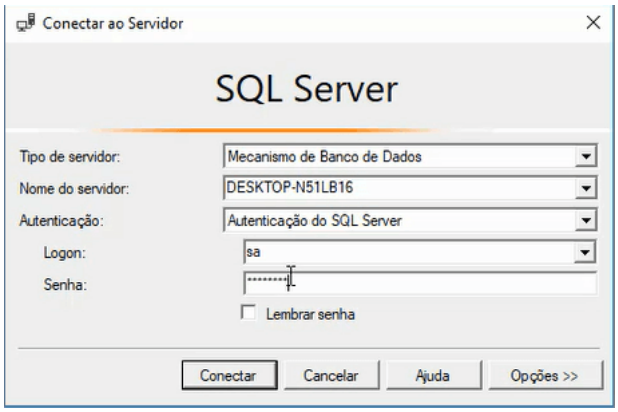
7) Após algum tempo, quando o segundo de comando volta ao normal, o monitor de atividades volta a ficar estável:



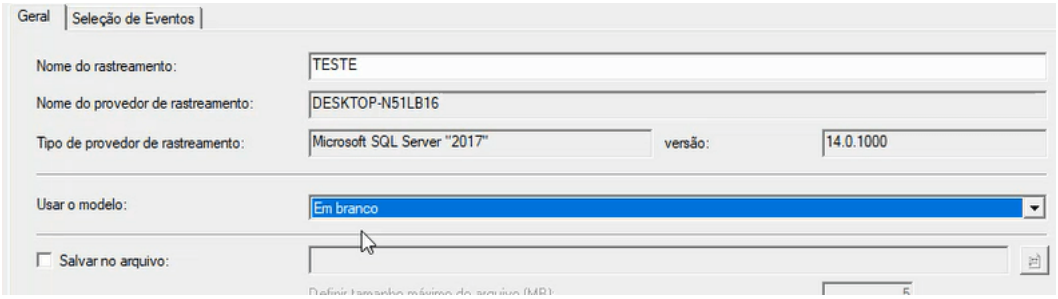
8) No menu do **Management Studio**, selecione **Ferramentas - SQL Server Profiler**:



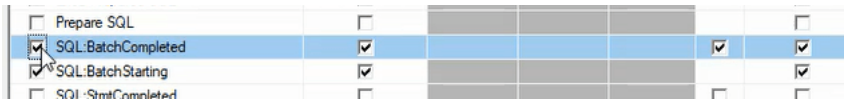
9) Efetue o login:



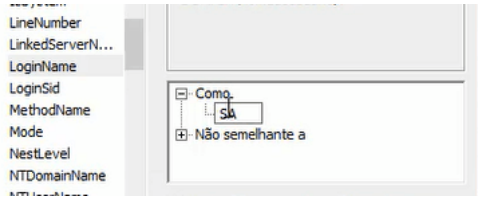
10) Coloque um nome para o *profile* e escolha o *template* **Em branco**:



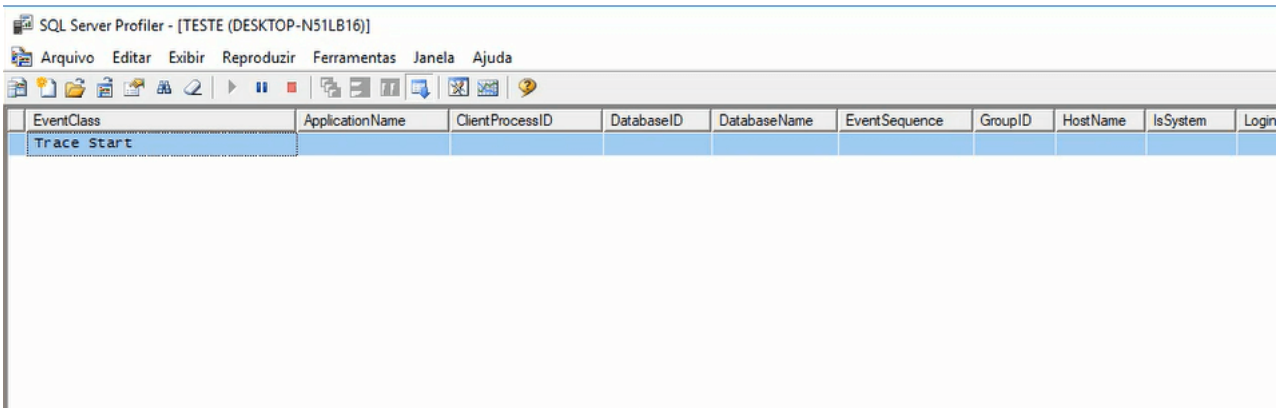
11) Na aba **Seleção de Eventos**, escolha somente as opções abaixo:



12) Clique no botão **Filtros de...**, selecione a seção **LoginName** e inclua o filtro abaixo:



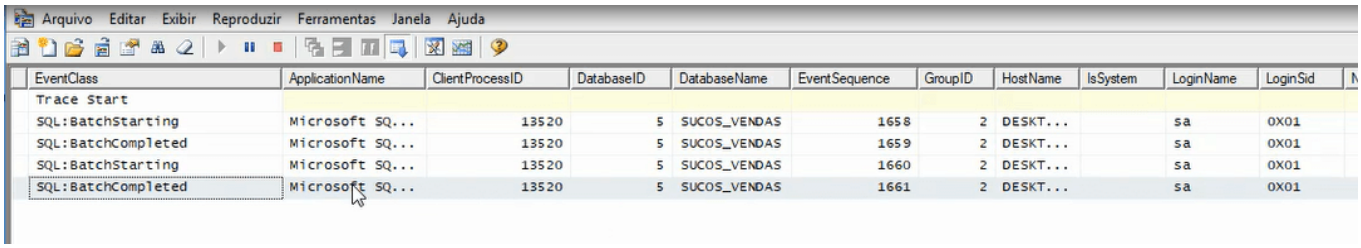
13) Feche esta caixa de diálogo e clique no botão **Executar**. O *trace* é inicializado:



14) Se você executar o comando:

```
SELECT * FROM [NOTAS FISCAIS] WHERE NUMERO = '100'
```

E voltar ao *trace*, você verá:

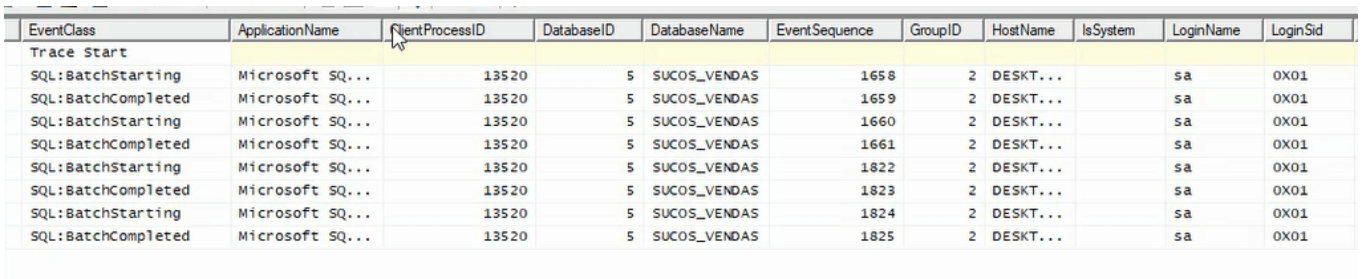


Ou seja, o rastreamento do comando.

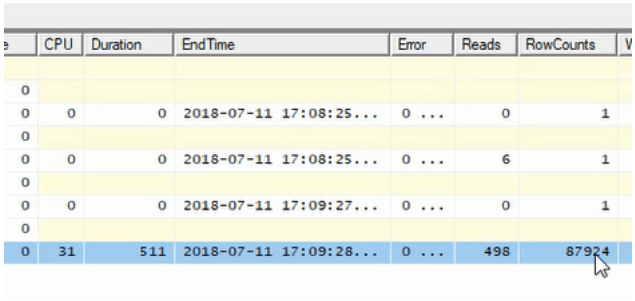
15) Repetindo a operação, agora com o comando:

```
SELECT * FROM [NOTAS FISCAIS]
```

Você coleta mais informação:



16) Uma informação importante é comparar o uso de CPU e o tempo de execução das consultas para comparar se um comando foi mais ou menos eficiente que outro:

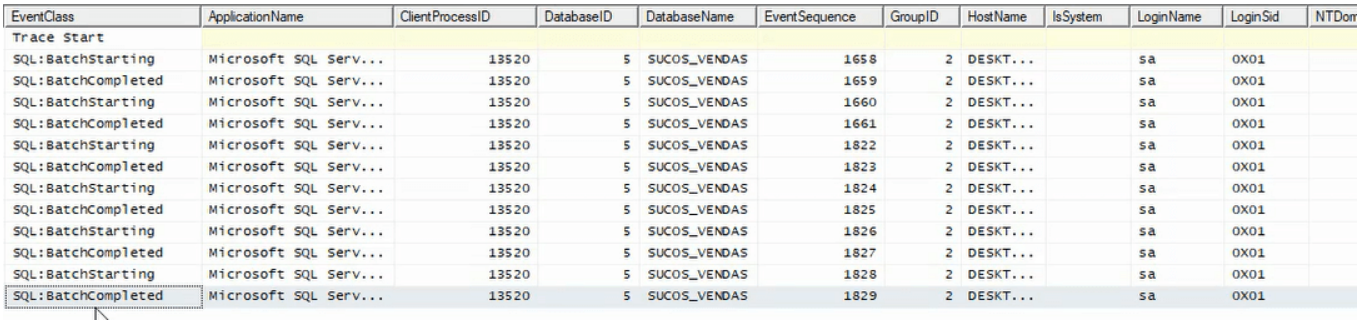


	CPU	Duration	EndTime	Error	Reads	RowCounts
0						
0	0	0	2018-07-11 17:08:25...	0 ...	0	1
0						
0	0	0	2018-07-11 17:08:25...	0 ...	6	1
0						
0	0	0	2018-07-11 17:09:27...	0 ...	0	1
0						
0	31	511	2018-07-11 17:09:28...	0 ...	498	87924

17) Execute o comando que representa um `JOIN`:

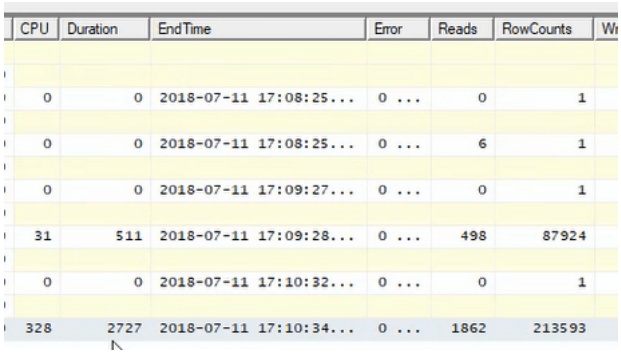
```
SELECT * FROM [TABELA DE CLIENTES] A INNER JOIN [NOTAS FISCAIS] B
ON A.CPF = B.CPF INNER JOIN [ITENS NOTAS FISCAIS] C ON B.NUMERO = C.NUMERO
```

Mais dados são registrados no *trace*:



EventClass	ApplicationName	ClientProcessID	DatabaseID	DatabaseName	EventSequence	GroupID	HostName	IsSystem	LoginName	LoginSid	NTDom
Trace Start											
SQL:BatchStarting	Microsoft SQL Serv...	13520	5	SUCOS_VENDAS	1658	2	DESKT...		sa	0x01	
SQL:BatchCompleted	Microsoft SQL Serv...	13520	5	SUCOS_VENDAS	1659	2	DESKT...		sa	0x01	
SQL:BatchStarting	Microsoft SQL Serv...	13520	5	SUCOS_VENDAS	1660	2	DESKT...		sa	0x01	
SQL:BatchCompleted	Microsoft SQL Serv...	13520	5	SUCOS_VENDAS	1661	2	DESKT...		sa	0x01	
SQL:BatchStarting	Microsoft SQL Serv...	13520	5	SUCOS_VENDAS	1822	2	DESKT...		sa	0x01	
SQL:BatchCompleted	Microsoft SQL Serv...	13520	5	SUCOS_VENDAS	1823	2	DESKT...		sa	0x01	
SQL:BatchStarting	Microsoft SQL Serv...	13520	5	SUCOS_VENDAS	1824	2	DESKT...		sa	0x01	
SQL:BatchCompleted	Microsoft SQL Serv...	13520	5	SUCOS_VENDAS	1825	2	DESKT...		sa	0x01	
SQL:BatchStarting	Microsoft SQL Serv...	13520	5	SUCOS_VENDAS	1826	2	DESKT...		sa	0x01	
SQL:BatchCompleted	Microsoft SQL Serv...	13520	5	SUCOS_VENDAS	1827	2	DESKT...		sa	0x01	
SQL:BatchStarting	Microsoft SQL Serv...	13520	5	SUCOS_VENDAS	1828	2	DESKT...		sa	0x01	
SQL:BatchCompleted	Microsoft SQL Serv...	13520	5	SUCOS_VENDAS	1829	2	DESKT...		sa	0x01	

E você vê que esta nova consulta consumiu mais recursos:



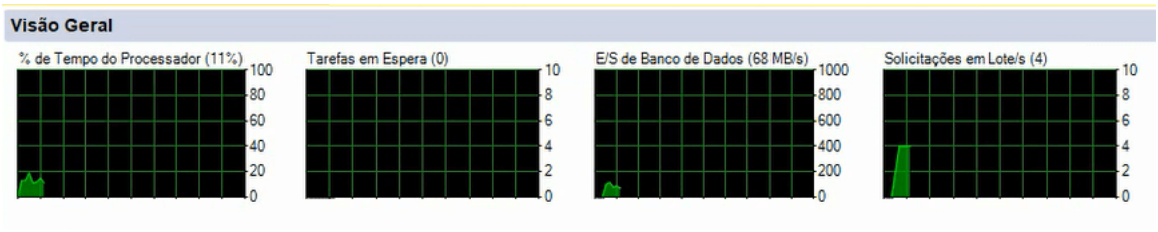
	CPU	Duration	EndTime	Error	Reads	RowCounts
0	0	0	2018-07-11 17:08:25...	0 ...	0	1
0	0	0	2018-07-11 17:08:25...	0 ...	6	1
0	0	0	2018-07-11 17:09:27...	0 ...	0	1
0	31	511	2018-07-11 17:09:28...	0 ...	498	87924
0	0	0	2018-07-11 17:10:32...	0 ...	0	1
0	328	2727	2018-07-11 17:10:34...	0 ...	1862	213593

18) Agora, execute os comandos abaixo, que irão consumir muitos recursos de máquina:

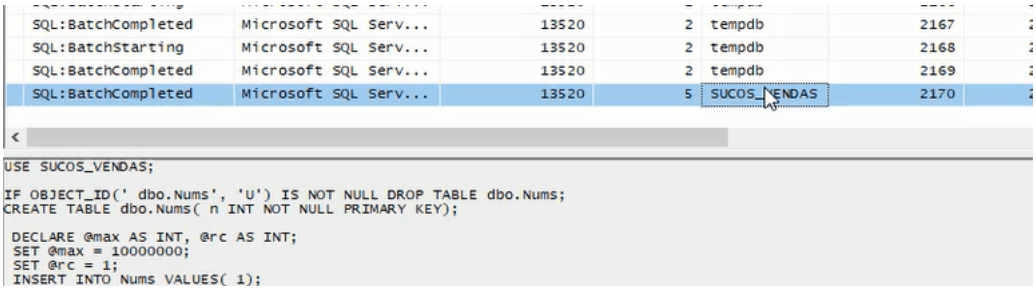
```
USE SUCOS_VENDAS;
IF OBJECT_ID(' dbo.Nums', 'U') IS NOT NULL DROP TABLE dbo.Nums;
CREATE TABLE dbo.Nums( n INT NOT NULL PRIMARY KEY);
DECLARE @max AS INT, @rc AS INT;
SET @max = 10000000;
SET @rc = 1;
INSERT INTO Nums VALUES( 1);
WHILE @rc * 2 < = @max
BEGIN
    INSERT INTO dbo.Nums SELECT n + @rc FROM dbo.Nums;
    SET @rc = @rc * 2;
END

INSERT INTO dbo.Nums SELECT n + @rc FROM dbo.Nums WHERE n + @rc < = @max;
```

```
SELECT n, n +  1 AS n_plus_one FROM dbo.Nums WHERE n < = 10000000;
```

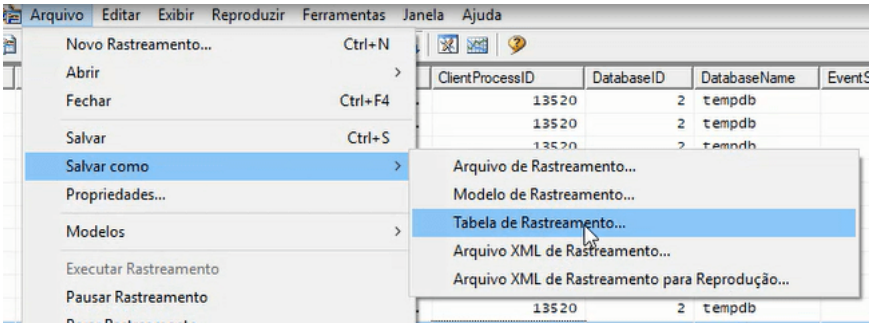
19) Você verá o *trace* executando o rastreamento deste comando, ao mesmo tempo que podemos acompanhá-lo no monitor de eventos:

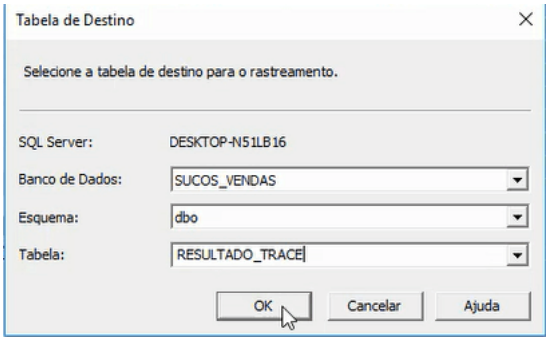


E no **SQL Profiler**, o monitoramento do processo:

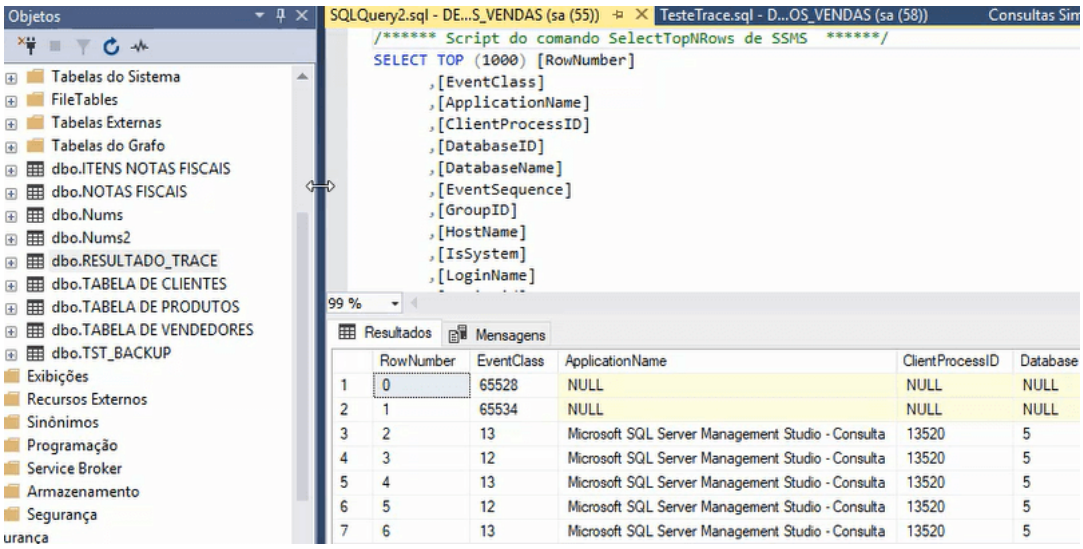


20) Você pode salvar o resultado do *trace* em uma tabela:





21) E visualizar o conteúdo do *trace* através de consultas usando T-SQL:



22) Abra o **Management Studio** e abra o arquivo **TestandoIndice.sql** (caso você ainda não o tenha baixado, baixe-o aqui (https://caelum-online-public.s3.amazonaws.com/839-administracao-do-sql-server-2017/04/TestandoIndice.sql)).

23) Selecione e execute os comandos abaixo:

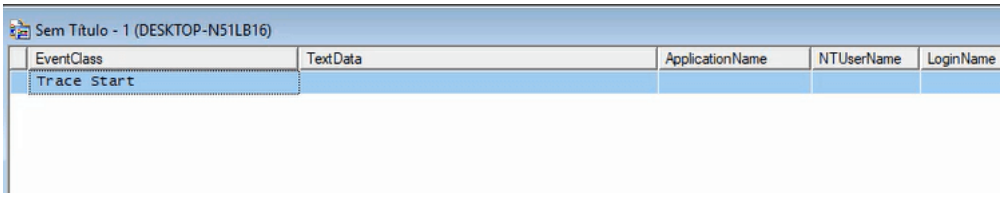
```
USE SUCOS_VENDAS;

CREATE TABLE dbo.Nums1( n varchar(10) NOT NULL);
CREATE TABLE dbo.Nums2( n varchar(10) NOT NULL);

DECLARE @max AS INT, @rc AS INT;
SET @max = 10000000;
SET @rc = 1;
INSERT INTO Nums1 VALUES( 1);
INSERT INTO Nums2 VALUES( 1);
WHILE @rc * 2 < = @max
BEGIN
    INSERT INTO dbo.Nums1 SELECT convert(varchar(10), n + @rc) FROM dbo.Nums1;
    INSERT INTO dbo.Nums2 SELECT convert(varchar(10), n + @rc) FROM dbo.Nums2;
    SET @rc = @rc * 2;
END

INSERT INTO dbo.Nums1 SELECT convert(varchar(10), n + @rc) FROM dbo.Nums1 WHERE n + @rc < = @max;
INSERT INTO dbo.Nums2 SELECT convert(varchar(10), n + @rc) FROM dbo.Nums2 WHERE n + @rc < = @max;
```

Aqui, você irá criar duas tabelas, chamadas `Nums1` e `Nums2`, onde você preenche com valores inteiros de 1 a 10 milhões.

24) Crie um índice para a tabela `Nums1` executando o comando:

```
CREATE NONCLUSTERED INDEX IX_Nums ON nums1 (n)
```

25) Acesse o **SQL Server Profiler** e proceda como feito na aula. Crie um *profile* para o usuário **sa** e que execute apenas o `SQL:BatchStarting` e `SQL:BatchCompleted`:



26) Execute os dois comandos abaixo, um de cada vez.

```
SELECT N FROM Nums1 where N = '10001'
```

E:

```
SELECT N FROM Nums2 where N = '10001'
```

Ambos irão retornar a linha cujo `N` é igual a 10001, porém um usando índice e outro não.

27) No **SQL Server Profiler**, você pode comparar os resultados do rastreamento:

NTUserName	LoginName	CPU	Reads	Writes	Duration	ClientProcessID	SPID	StartTime
								2018-07-1
	sa					13520	55	2018-07-1
	sa	0	0	0	0	13520	55	2018-07-1
	sa					13520	55	2018-07-1
	sa	2	25964	9	535	13520	55	2018-07-1
	sa					13520	55	2018-07-1
	sa	0	0	0	0	13520	55	2018-07-1
	sa					13520	55	2018-07-1
	sa	0	23	0	0	13520	55	2018-07-1