**[Are scalar functions really a bad thing?](http://www.sqlservercentral.com/blogs/confessions-of-a-microsoft-addict/2018/09/10/are-scalar-functions-really-a-bad-thing/)**

[Daniel Janik](http://www.sqlservercentral.com/blogs/confessions-of-a-microsoft-addict/)

Posted on 10 September 2018

FROM: <http://www.sqlservercentral.com/blogs/confessions-of-a-microsoft-addict/2018/09/10/are-scalar-functions-really-a-bad-thing/?utm_source=SSC&utm_medium=pubemail>

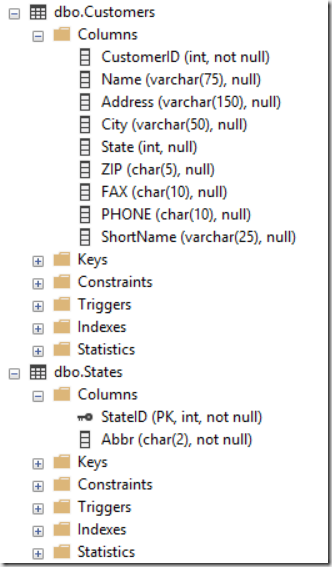
I’ve read a lot of things lately pointing to scalar functions as if they were the devil. In this blog I’m going to explore if that’s the case. Let’s have a look.

It’s true that in many situations a scalar function is often a performance bottleneck; but, is there a situation where they could be responsibly used?

What if you had a lookup table that almost never changed? Is it worth doing a join on the lookup to get the data you need?

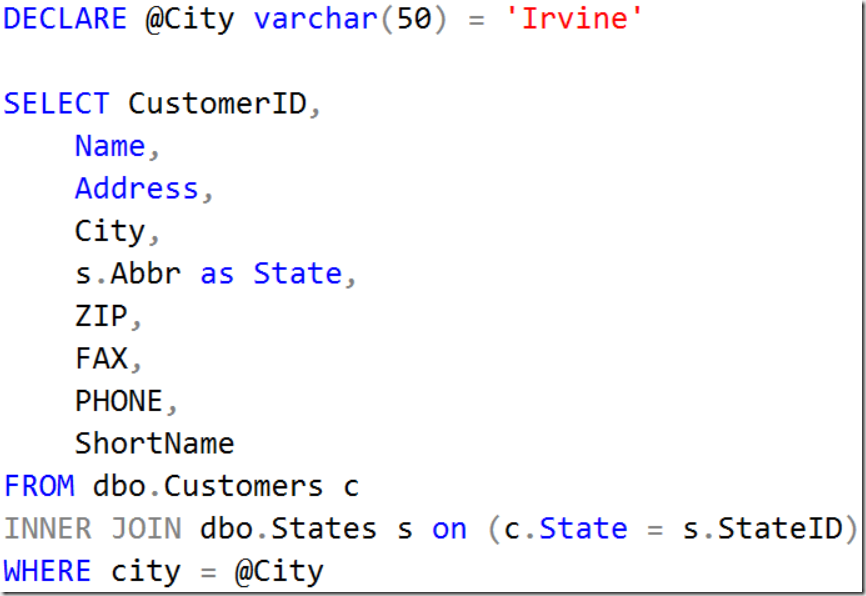
Let’s examine a simple join between a customer address and a state lookup table.

We have the following tables

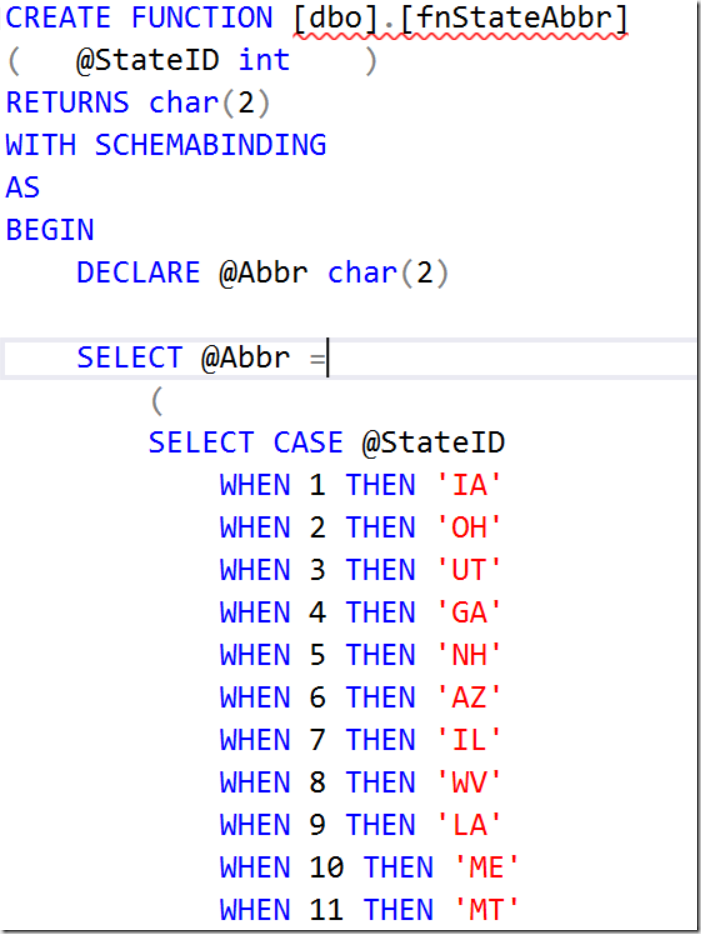
[](https://danieljanikcom.files.wordpress.com/2018/09/image.png)

To join these you’d usually use a very simple inner join to get the state code, which could also be the state name which is longer than the 4 bytes for int. You may also include other metrics on the States table like geo data or capitol. For now let’s pretend like Abbr is much larger than State (int) so it would make sense to dump it to a lookup table.

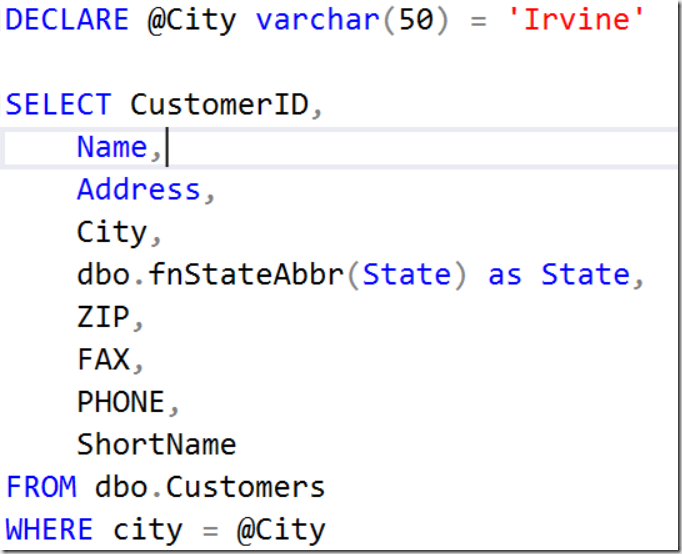
Here’s what the join query may look like

[](https://danieljanikcom.files.wordpress.com/2018/09/image1.png)

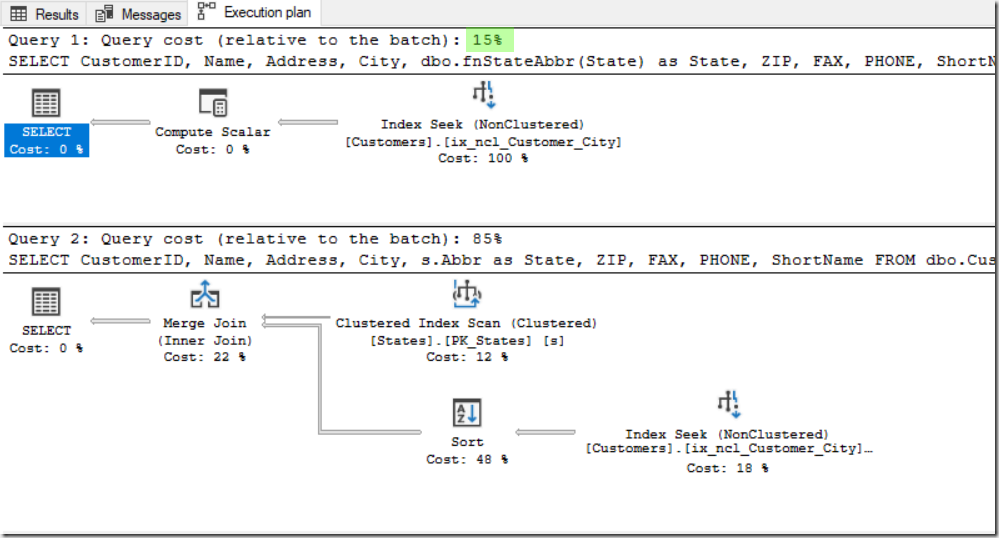
To test the value of scalar functions we created the the following code (note that it cuts off but you get the point)

[](https://danieljanikcom.files.wordpress.com/2018/09/image2.png)

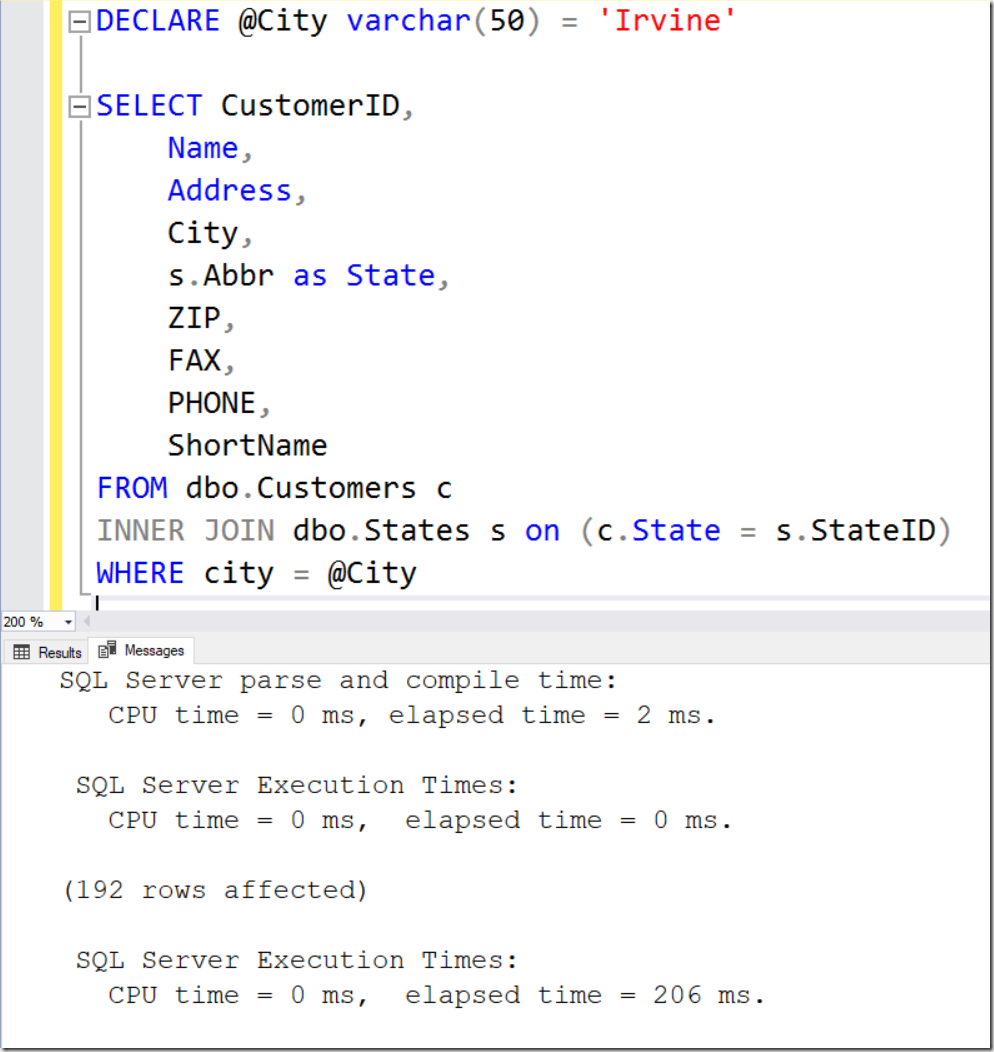
Now that we have a function to test with, we want to create a query that returns the same results as our join query. Let’s have a look.

[](https://danieljanikcom.files.wordpress.com/2018/09/image3.png)

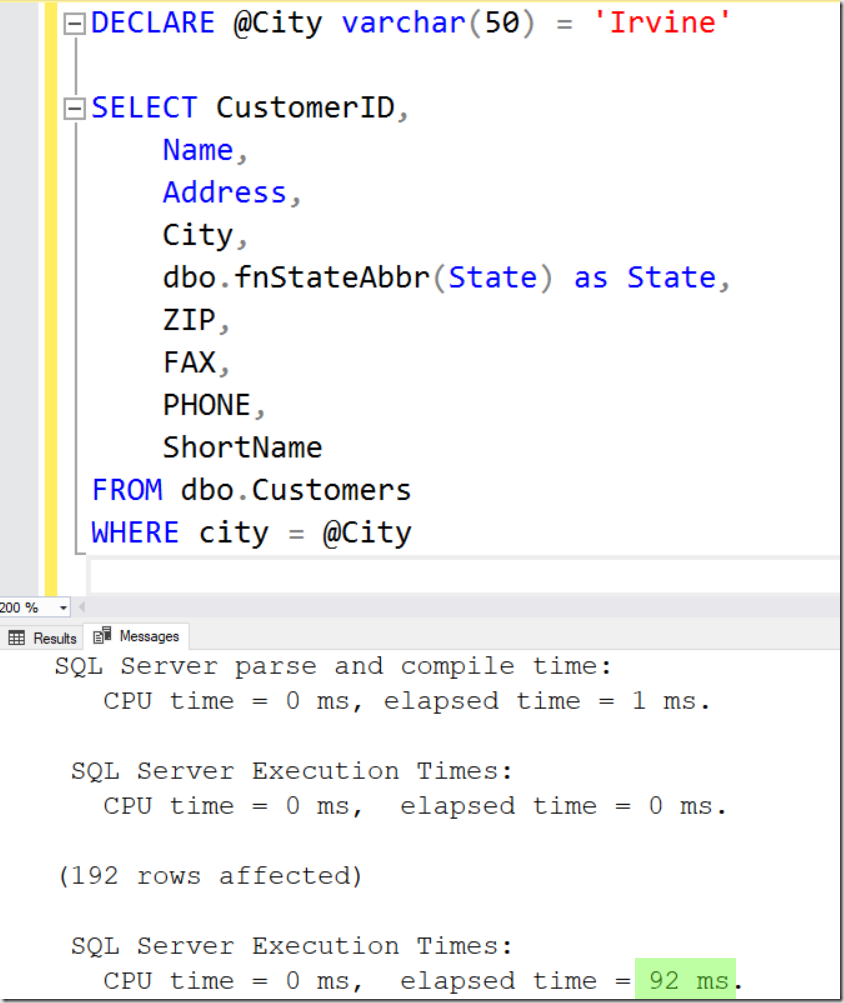
How do these compare with an actual plan?

[](https://danieljanikcom.files.wordpress.com/2018/09/image4.png)

Wow! The query with the function has much less cost! Incredible results, right? Not so fast! Sometimes cost can be misleading. To perform a true test we’ll need to do a bit more. Let’s turn off the plan and turn on STATISTICS TIME.

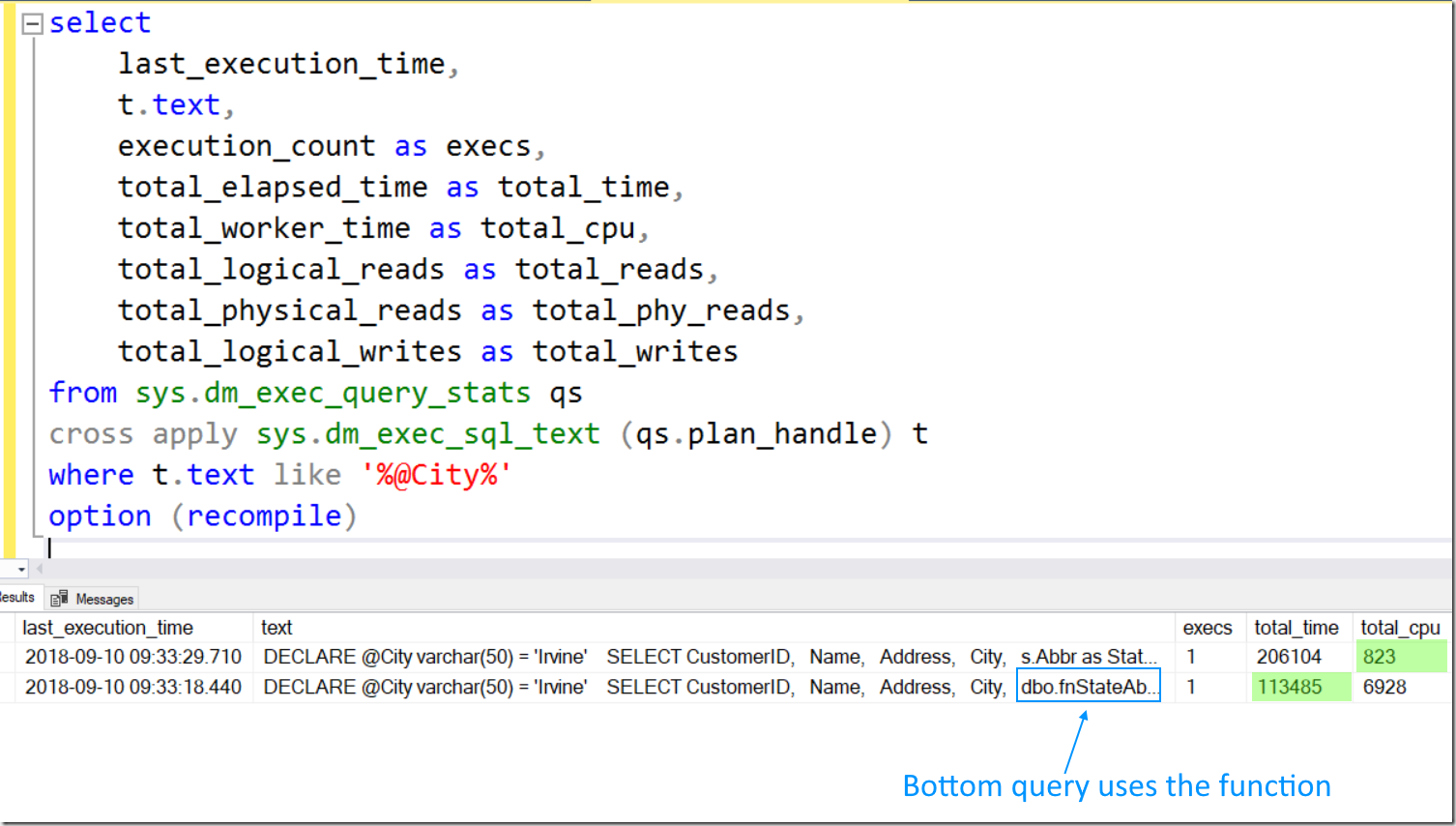
[](https://danieljanikcom.files.wordpress.com/2018/09/image5.png)

206 ms total elapsed time for the join query.

[](https://danieljanikcom.files.wordpress.com/2018/09/image6.png)

Only 92 ms for the funtion query. That’s good!

Let’s examine the query stats to see how these two compare.

[](https://danieljanikcom.files.wordpress.com/2018/09/image7.png)

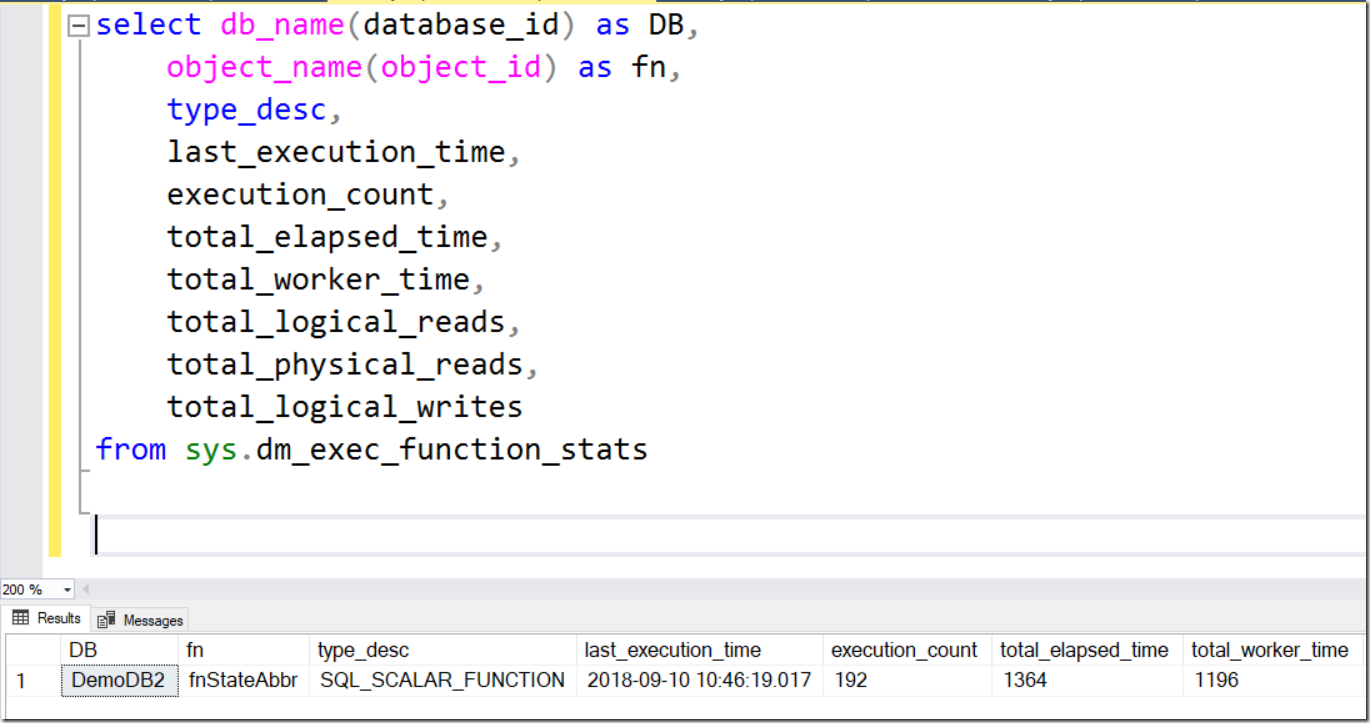
Here we find some unexpected results. The execution time was less for this single run; however, CPU was greater by a significant amount.

We should dig deeper. You may have not noticed in the query plans that the function isn’t represented in the plan at all. Note that this is SQL Server 2017 and the database is running in the latest compatibility level.

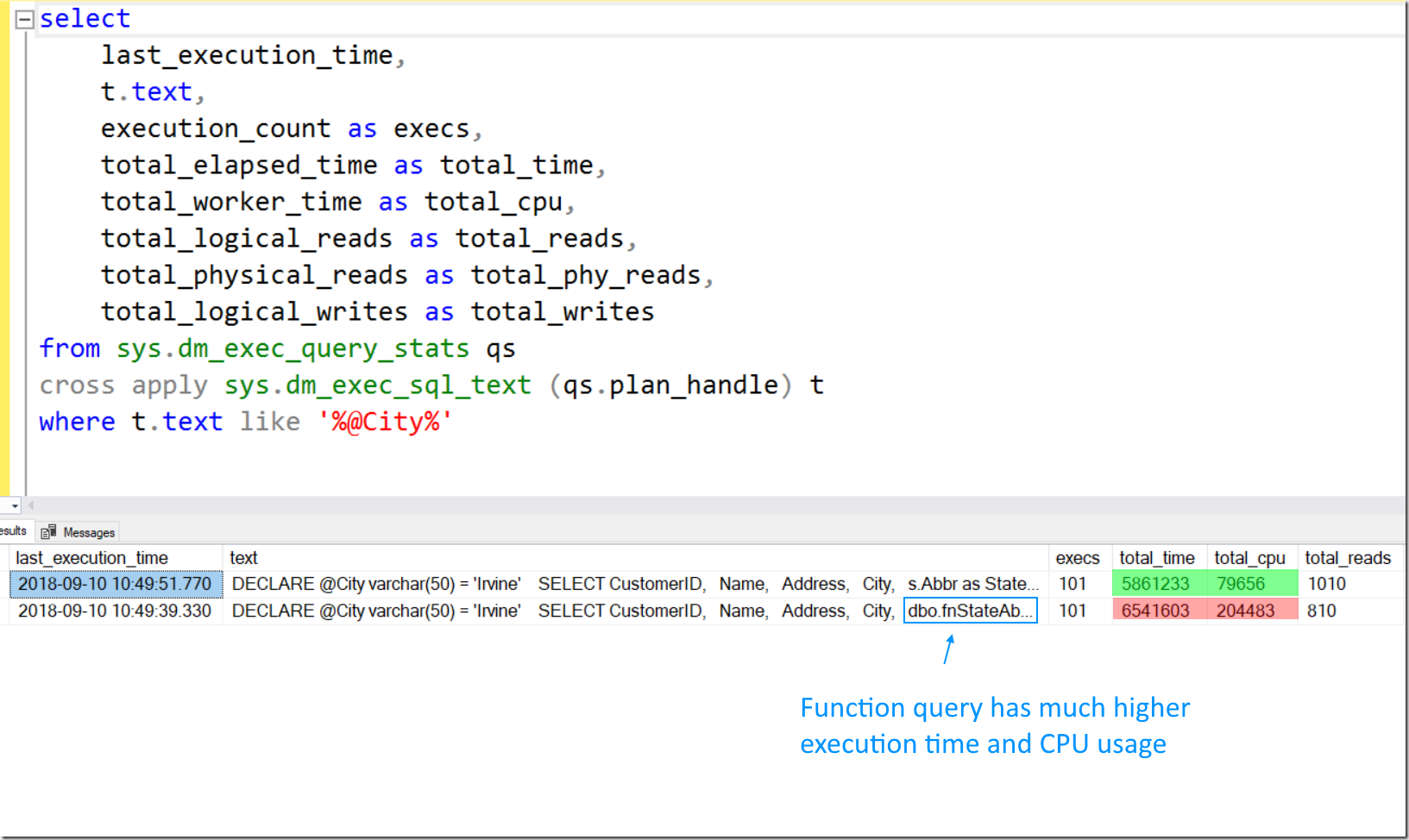
To find out more we’re going to look at the function stats DMV.

SELECT db\_name(database\_id) as DB,  
object\_name(object\_id) as fn,  
type\_desc,  
last\_execution\_time,  
execution\_count,  
total\_elapsed\_time,  
total\_worker\_time,  
total\_logical\_reads,  
total\_physical\_reads,  
total\_logical\_writes  
  
FROM sys.dm\_exec\_function\_stats

What we find is the function was executed 192 times. This is because the function must run once for each row in the result and it’s primarily this reason why scalar functions aren’t usually a great choice.

[](https://danieljanikcom.files.wordpress.com/2018/09/image8.png)

One final test is to run the queries 100 times. To do this, we’re going to use GO 100.

[](https://danieljanikcom.files.wordpress.com/2018/09/image9.png)

What have we learned? While the function may look great in the query plan it doesn’t stack up to actual runtime metrics. If there’s one thing a DBA should always say it’s “It depends” but in this case it doesn’t and you should instead add this quote to your toolbox, “You should never fully trust cost in query plans”.

http://www.sqlservercentral.com/Resources/Images/loading.gifLoading comments...

**Posted by When Scalar Functions Go Bad &#8211; Curated SQL on 11 September 2018**

[…] Daniel Janik head-fakes us a few times when looking at scalar user-defined function performance: […]

[Like](https://sqltechblog.com/2018/09/10/are-scalar-functions-really-a-bad-thing/?like_comment=2107&_wpnonce=73bda0e104)Like

**Posted by Ron Kyle on 18 September 2018**

You don’t answer the question posed in the title–are scalar function good, bad, or does it depend? I’m not sure how your conclusion of not trusting the query plan cost data answers this.

[Like](https://sqltechblog.com/2018/09/10/are-scalar-functions-really-a-bad-thing/?like_comment=2149&_wpnonce=4547163a33)Like