**Are your indexes being thwarted by mismatched datatypes?**

Unexpected SQL Server Performance Killers #1

FROM: <https://blog.bertwagner.com/are-your-indexes-being-thwarted-by-mismatched-datatypes-d3985375e528>

*In this series I explore scenarios that hurt SQL Server performance and show you how to avoid them. Pulled from my collection of “things I didn’t know I was doing wrong for years.”*

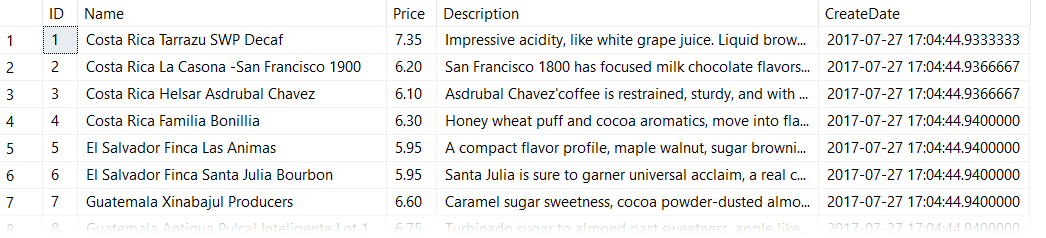
Have you ever encountered a query that runs slowly, even though you’ve created indexes for it?

There’s a few different reasons why this may happen. The one I see most frequently happens in the following scenario.

**I’ll have an espresso please**

Let’s say I have a [table dbo.CoffeeInventory](https://gist.github.com/bertwagner/e1225294ae73f2d74575d834e93597e8) of coffee beans and prices that I pull from [my favorite green coffee bean supplier](https://www.sweetmarias.com/) each week. It looks something like this:

-- Make sure Actual Execution Plan is on  
-- Let's see what our data looks like  
SELECT \* FROM dbo.CoffeeInventory



If you want to follow along, you can get this data set from [this GitHub Gist](https://gist.github.com/bertwagner/e1225294ae73f2d74575d834e93597e8)

I want to be able to efficiently query this table and filter on price, so next I create an index like so:

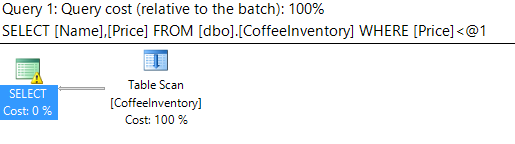
CREATE CLUSTERED INDEX CL\_Price ON dbo.CoffeeInventory (Price)

Now, I can write my query to find out what coffee prices are below my willingness to pay:

SELECT Name, Price FROM dbo.CoffeeInventory WHERE Price < 6.75

You would expect this query to be blazing fast and use a clustered index seek, right?

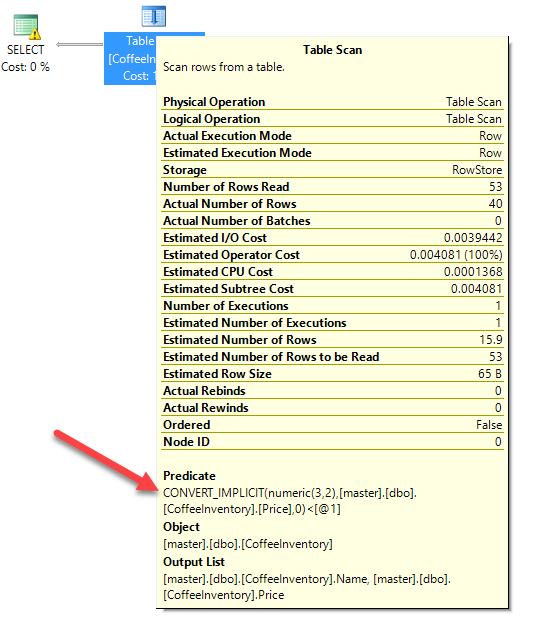
**WRONG!**



What the heck?

Why is SQL scanning the table when I added a clustered index on the column that I am filtering in my predicate? That’s not how it’s supposed to work!

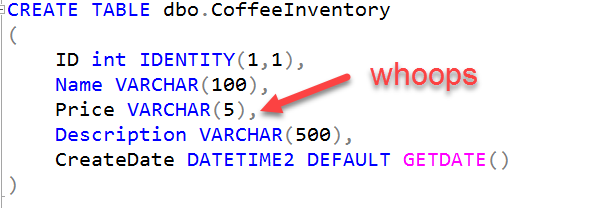
Well dear reader, if we look a little bit closer at the table scan operation, we’ll notice a little something called CONVERT\_IMPLICIT:



CONVERT\_IMPLICIT: ruiner of fast queries

What is CONVERT\_IMPLICIT doing? Well as it implies, it’s having to convert some data as it executes the query (as opposed to me having specified an explicit CAST() or CONVERT() function in my query).

The reason it needs to do this is because I defined my Price column as a VARCHAR(5):



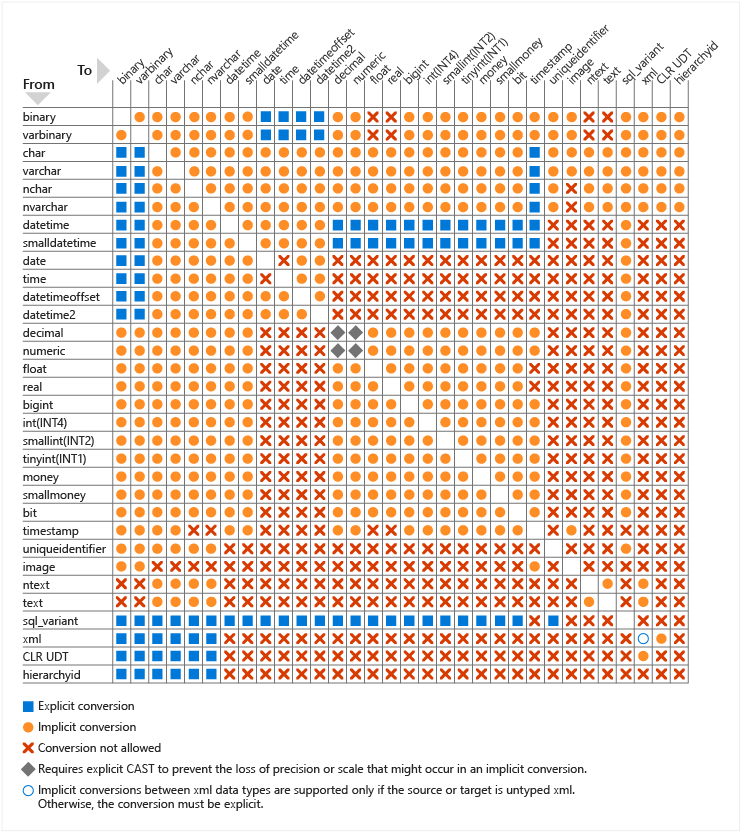
Who put numeric data into a string datatype? Someone who hasn’t had their coffee yet today.

In my query however, I’m doing a comparison against a number WHERE Price < 6.75. SQL Server is saying it doesn’t know how to compare a string to a number, so it has to convert the VARCHAR string to a NUMERIC(3,2).

**This is painful.**

Why? Because SQL is performing that implicit conversion to the numeric datatype for *every single row* in my table. Hence, it can’t seek using the index because it ends up having to scan the whole table to convert every record to a number first.

And this doesn’t only happen with numbers and string conversion. [Microsoft has posted an entire chart](https://docs.microsoft.com/en-us/sql/t-sql/data-types/data-type-conversion-database-engine) detailing what types of data type comparisons will force an implicit conversion:



<https://docs.microsoft.com/en-us/sql/t-sql/data-types/data-type-conversion-database-engine>

That’s a lot of orange circles/implicit conversions!

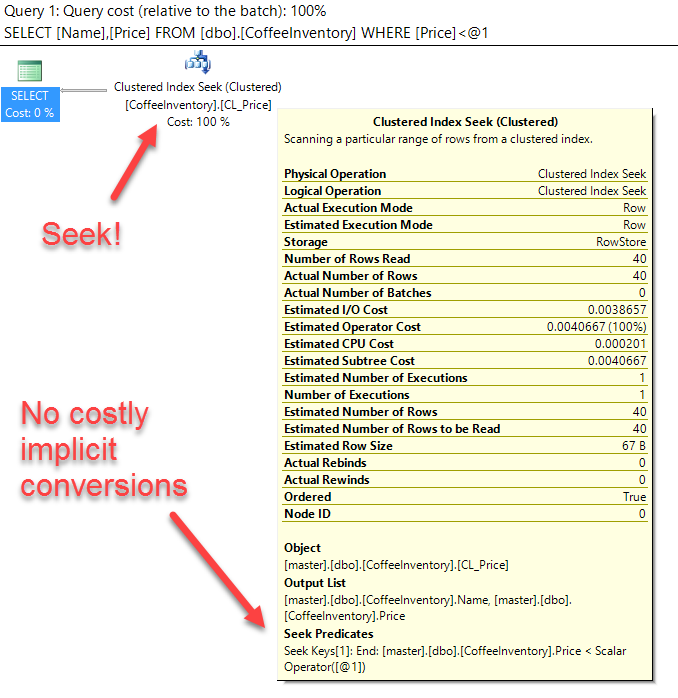
**How can I query my coffee faster?**

Well in this scenario, we have two options.

1. Fix the datatype of our table to align with the data actually being stored in this (data stewards love this).
2. Not cause SQL Server to convert every row in the column.

Number 1 above is self-explanatory, and the better option if you can do it. However, if you aren’t able to modify the column type, you are better off writing your query like this:

SELECT Name, Price FROM dbo.CoffeeInventory WHERE Price < '6.75'



Since we do a comparison of equivalent datatypes, SQL Server doesn’t need to do any conversions and our index gets used. Woo-hoo!

**What about the rest of my server?**

Remember that chart above? There are a lot of different data comparisons that can force a painful column side implicit conversion by SQL Server.

Fortunately, [Jonathan Kehayias](https://www.sqlskills.com/blogs/jonathan/finding-implicit-column-conversions-in-the-plan-cache/) has written a great query that helps you find column side implicit conversions by querying the plan cache. Running his query is a great way to identify most of the implicit conversions happening in your queries so you can go back and fix them — and then rejoice in your improved query performance!