SQL SERVER Performance Tuning Notes

http://msbiskills.com/

CAST IS SARgable, How to write better Where Clause. Let's Check?

SQL Server has many functions that can be used in your WHERE clause and in the SELECT clause. Now these functions can be user defined or inbuilt. These functions normally provide functionality which would be very difficult to get without these functions.

Now when these functions are used improperly in the WHERE clause these functions can cause major performance issues. Now very few SQL functions are SARgable and most of them are NOT SARgable.

Now what is SARgable?

It is Search Argument able. It's the ability of the query optimizer to use indexes. Well we can tweak where clause of our queries little bit so that optimizer can use indexes.

Let's see an example that shows using a function in the WHERE clause can affect performance. Add actual execution plan and statistics IO ON for better understanding.

Let's say we have a table called CustomerInfo with around 20K records. This Customerinfo table has 3 columns named (CustomerID, CustomerName, and ModifiedDate). This table also has 2 indexes which are given below-

index_name	index_description	index_keys
Ix_ModifiedDate	nonclustered located on PRIMARY	ModifiedDate
PK__Customer__A4AE64B86C543C6C	clustered, unique, primary key located on PRIMARY	CustomerID

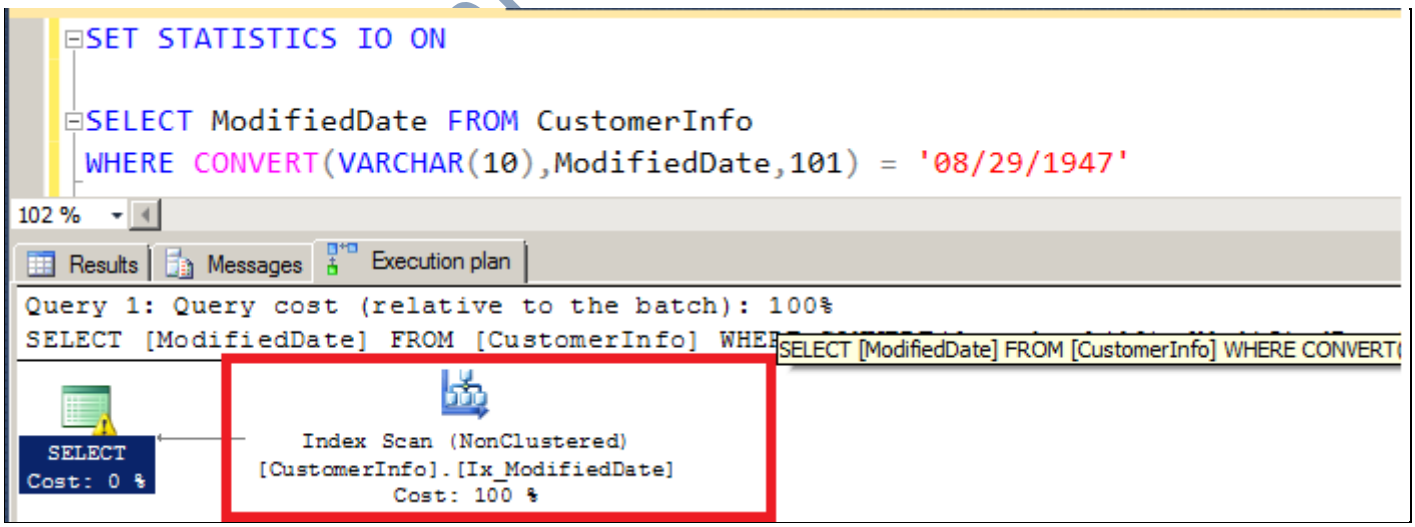
Now let's say we wanted to find out below all modified dates when ModifiedDate = '08/29/1947'. So we wrote the below query-

```
SET STATISTICS IO ON

SELECT ModifiedDate FROM CustomerInfo
WHERE CONVERT(VARCHAR(10),ModifiedDate,101) = '08/29/1947'
```

Now the above query is perfectly fine, it's give us data that we wanted but the problem here is that we are not using any indexes effectively here. It is reading all the pages from the leaf level of the tree. Let's check out the execution plan.

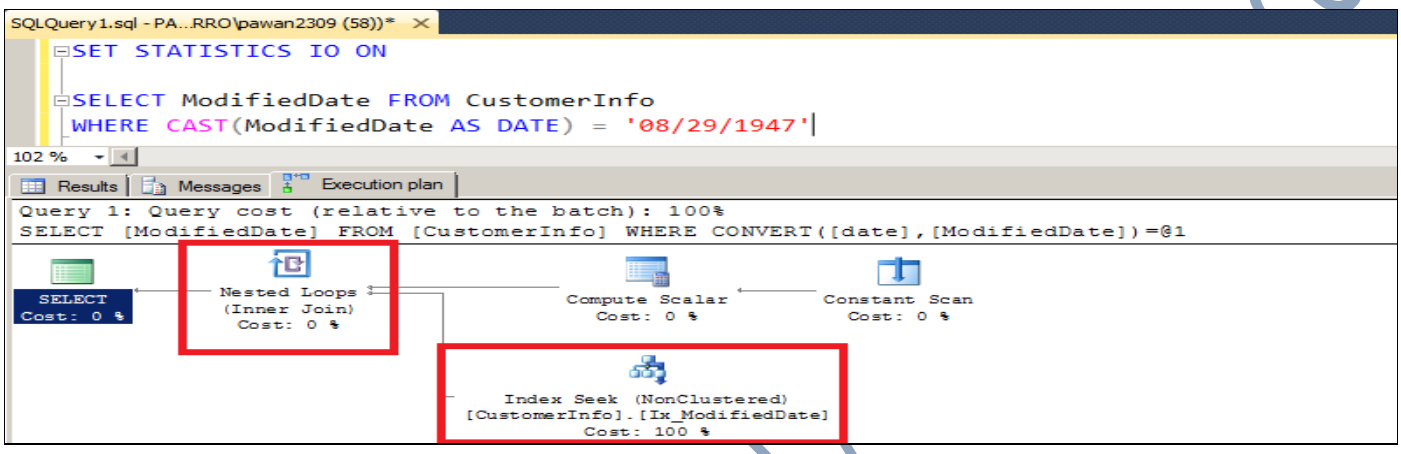
In the below execution plan we can clearly see that we are doing nonclustered scan of complete leaf pages even we have index on this columns.



So this is not good. Now let's rewrite this query in a different fashion and check its execution plan.

```
SET STATISTICS IO ON

SELECT ModifiedDate FROM CustomerInfo
WHERE CAST(ModifiedDate AS DATE) = '08/29/1947'
```



Now as per the execution plan we are doing index seek which is a better approach in this case. But here we are doing some other physical operations like Nested Loop, Compute Scalar and Constant scan. Although these operators are not taking any cost but the question we have right now is can we remove these extra physical operators.

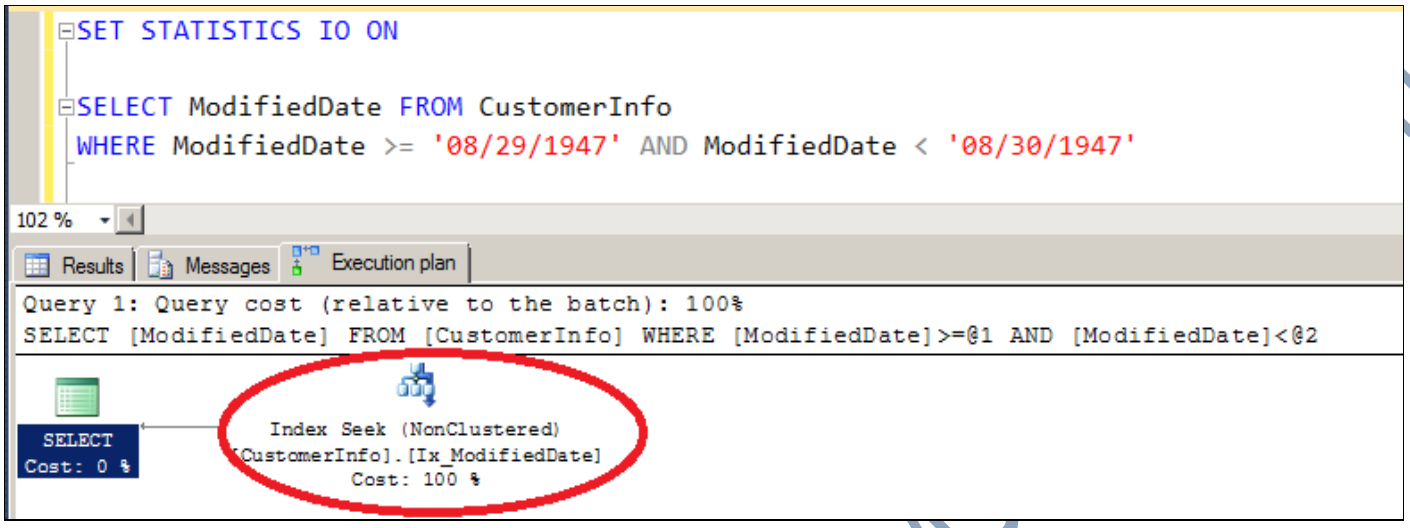
Let's rewrite this query in some other fashion and check out its execution plan.

```
SET STATISTICS IO ON

SELECT ModifiedDate FROM CustomerInfo
WHERE ModifiedDate >= '08/29/1947' AND ModifiedDate < '08/30/1947'
```

In the below execution plan we can see that we have only one physical operation that is index seek. This type of seek is called range based seek which is good in this case. Well you choose

query 2 or query 3 depending upon your requirement but certainly you would like to change query 1.



Summary

When we are checking things for performance always try to find out these small things. These can help you queries performance greatly. We are reducing I/O by reading less data pages. So by not using functions in the WHERE clause we can provide big performance gains. We just have to tweak where clause. That should be easy.

That's all folks; I hope you've enjoyed learning about Avoid functions in the where clause – Part II, and I'll see you soon with more "Performance Tuning" articles.

Thanks!

Pawan Kumar Khowal

MSBISKills.com