SQL SERVER Performance Tuning Notes

http://msbiskills.com/

Avoid Using Function in WHERE Clause. Why?

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. When these same functions are used in the WHERE clause this forces SQL Server query optimizer to do a table scan or index scan to get the desired results instead of doing an index seek if there is an index that can be used.

The reason for this is that SQL Server has to call the function for each value of the column and match that with the matching criteria.

Some examples are given below 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

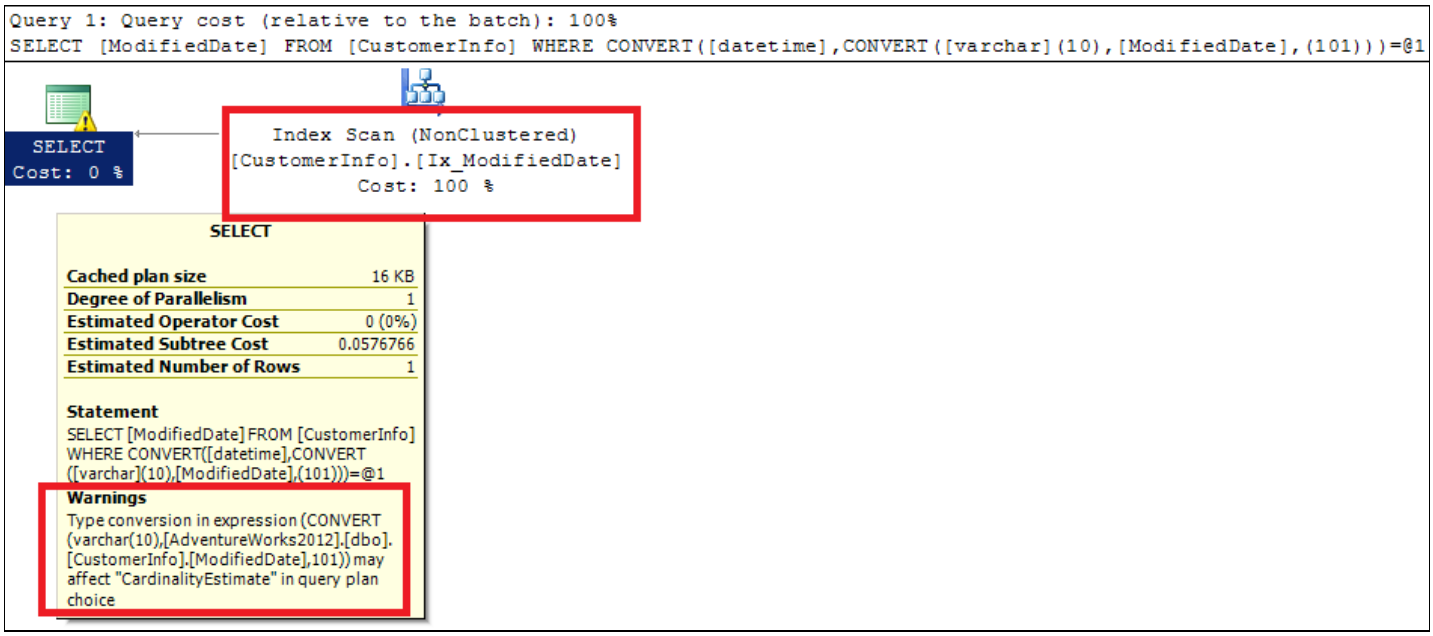
Now let's say we wanted to find out below all modified dates when ModifiedDate = '1947-08-29'. So to satisfy above query we wrote below query

```
SELECT ModifiedDate FROM CustomerInfo WHERE ModifiedDate = '1947-08-29'
```

But above query will not return anything since the Modified Date column is of type DateTime. So we have to modify our query. So we came up with a new query.

```
SELECT ModifiedDate FROM CustomerInfo WHERE
CAST(CONVERT(VARCHAR(10),ModifiedDate,101) AS DATETIME) = '1947-08-29'
```

Now let's check the execution plan of the above query. Now with the below execution plan we can clearly find out 2 bad things. One is that we are having NonClustered index scan and the other thing is that we have a warning due to column data type conversion. The culprit here is the functions we are using in the where clause.

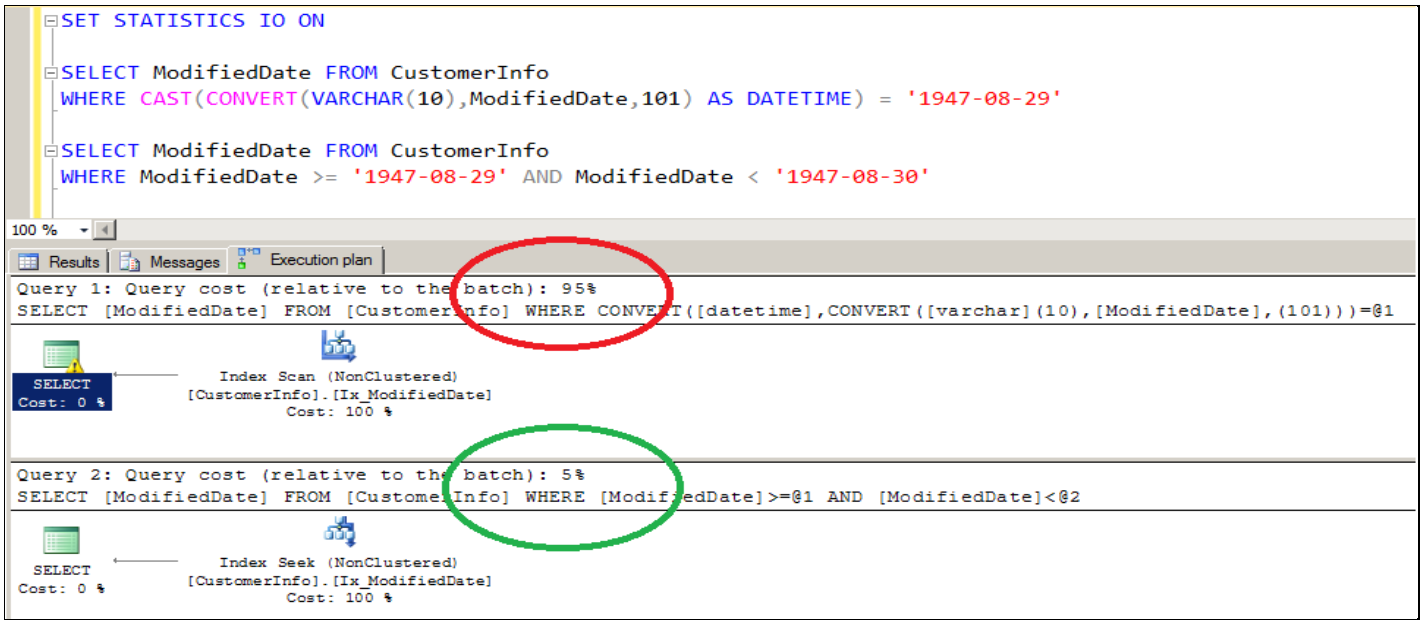


Now let's modify the above query and check out what happens. Here we are getting Index seek.

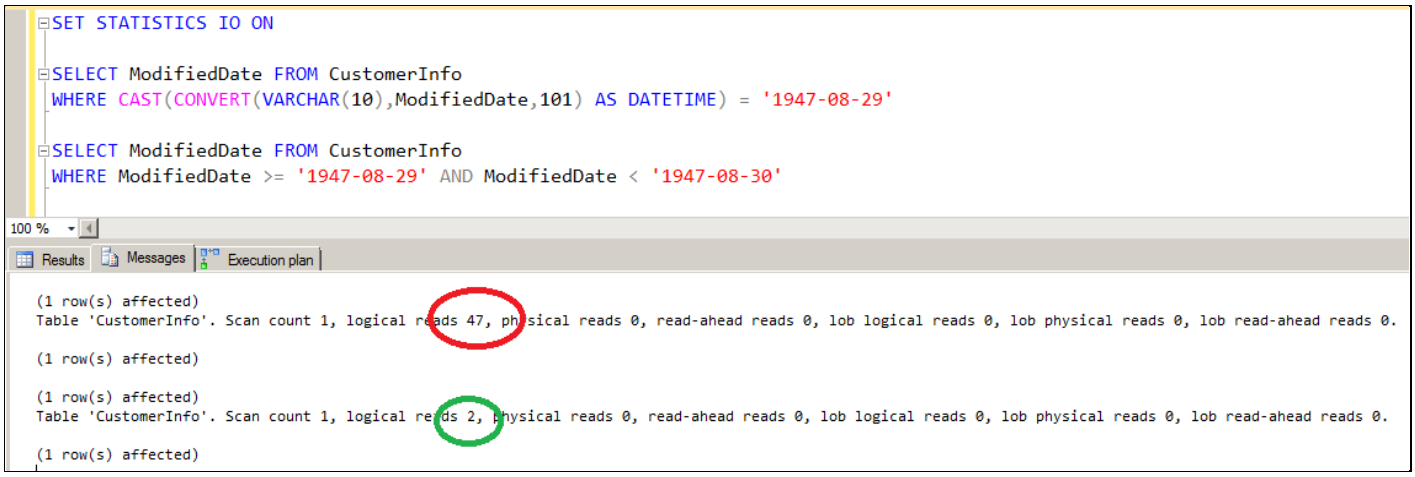
```
SELECT ModifiedDate FROM CustomerInfo
WHERE ModifiedDate >= '1947-08-29' AND ModifiedDate < '1947-08-30'
```

Now one of the best things in SSMS is that we have run 2 queries simultaneously and compare how much each query costs. First query is taking 95% of the cost and seconds query is taking

only 5% of the cost. Now the first query is reading all the records from the leaf node that's why we have Non Clustered Scan instead of seek.

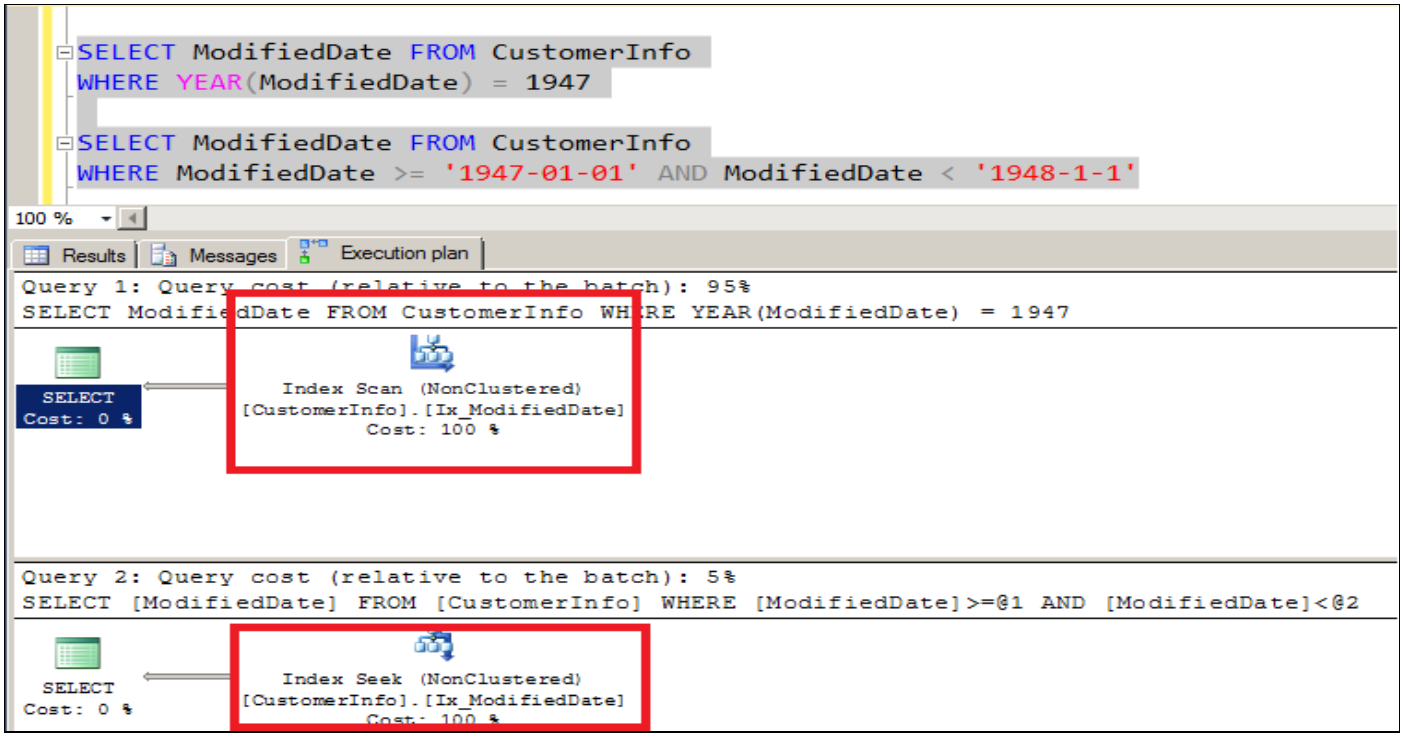


Now check out the difference between logical reads (1 Logical Read = 1 8KB Page). The first query is having 47 logical reads and in second case the query is having only 2 logical read because we are seeking index.



In the second query we are using range based seeks. Hence always try not to use functions with the columns in the where clause. Always try to rewrite the query so that we can get good performance.

Now let's try some more examples. In below example we are finding modified date for a particular year.



In the above example in the first query is using Year function on Modified date to pick up the year and then we are matching that year with value 1947. That's why we are getting a NonClustered Scan as SQL Server has to read all the values from leaf node. In the second query we are using where clause effectively that's why we are getting a NonClustered Index seek, basically it is a range based seek which is much more efficient as we have read less number of data pages.

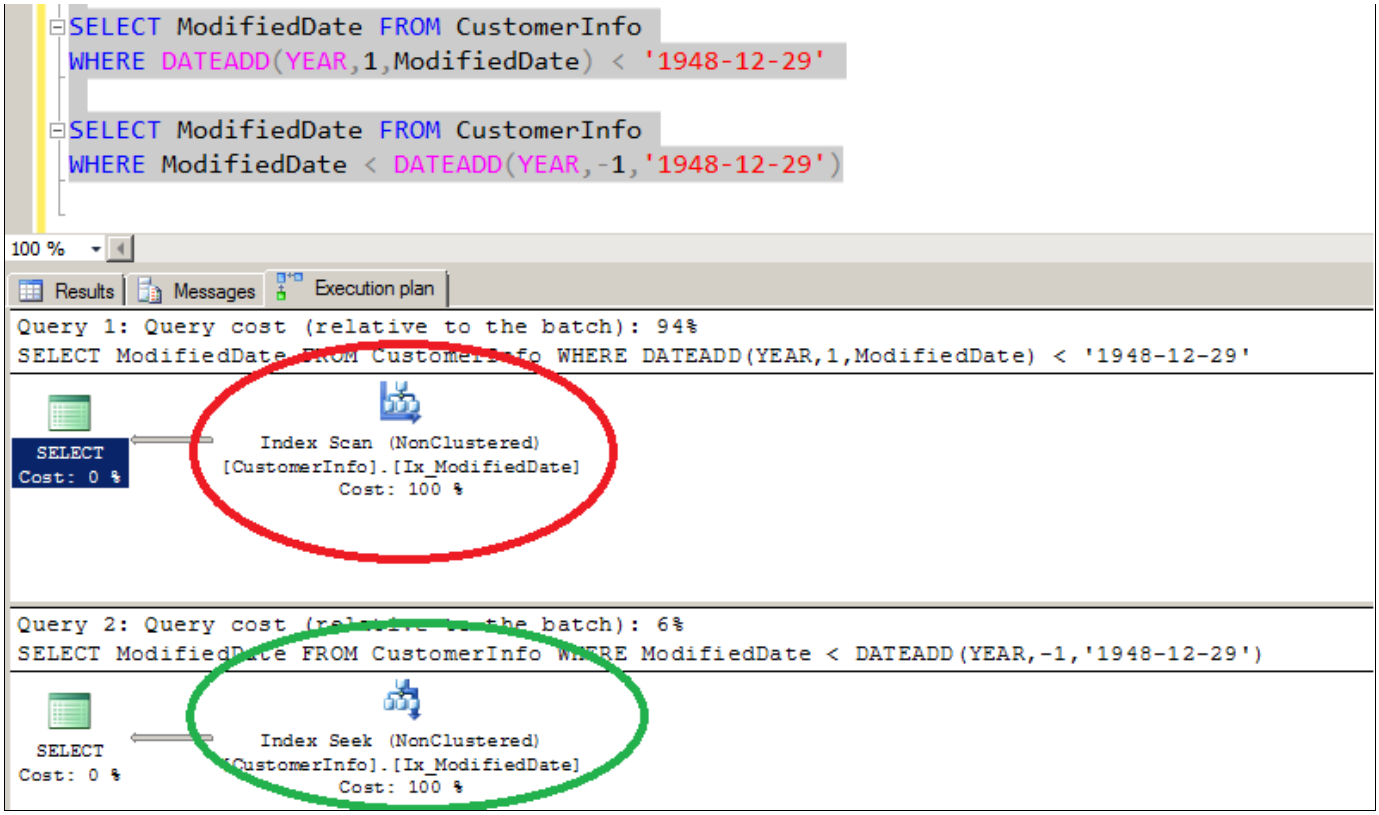
Let's have one more example. Explanation is not required in that case. It's an excellent way to use Where clause to get Seek. Let's say you want to fetch records less than a date – One Year.

E.g. Modified date < '1948-12-29' – 1 Year.
Now check out the queries below.

```
SELECT ModifiedDate FROM CustomerInfo
WHERE DATEADD(YEAR,1,ModifiedDate) < '1948-12-29'

SELECT ModifiedDate FROM CustomerInfo
WHERE ModifiedDate < DATEADD(YEAR,-1,'1948-12-29')
```

Now check the execution plans below. Clearly we can see that second query is taking only 6% of the total cost and the first query is taking 96% of the cost. That's why second query is a clear winner in this case.



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, and I'll see you soon with more "Performance Tuning" articles.

Thanks !

Pawan Kumar Khowal, MSBISKills.com