

Non-SARGable predicates

Last updated by | Vitor Tomaz | Feb 24, 2023 at 3:32 AM PST


Contents

- [What is SARGable](#)
- [What makes a predicate Non-SARGable](#)
- [How to mitigate](#)
- [More information](#)

What is SARGable

SARG stands for **S**earch **ARG**ument. It's an important term since it tells us if a predicate can be used for an index seek.

When a predicate is non-SARGable it means that you will see operators like Index Scans.

According with this [article](#) 

"A sargable predicate is one of the form (or which can be put into the form) "comparison-operator value". SARGS are expressed as a boolean expression of such predicates in disjunctive normal form."

What makes a predicate Non-SARGable

For a predicate to be SARGable it requires for the column and value to be directly compared.

A few examples for better understanding below.

Let's first create a table for our example, inserting some data.

```
create table sargtest(id int identity(1,1) primary key clustered,  
col1 nvarchar(255),  
col2 varchar(255))  
  
go  
insert into sargtest values (CONVERT(nvarchar(255), NEWID()),CONVERT(varchar(255), NEWID()))  
go 2000
```

Now creating two indexes:

```
create index idx01 on sargtest(col1)  
create index idx02 on sargtest(col2)
```

The predicate below is SARGable - Note the Index Seek Operation:

```
SET SHOWPLAN_TEXT on
go
select * from sargtest where col1 = '04540F08-A62B-4F45-8E4A-BA67FE7A63B1'
```

```
|--Nested Loops(Inner Join, OUTER REFERENCES:([archetype_rimarqu].[dbo].[sargtest].[id]))
|--Index Seek(OBJECT:([archetype_rimarqu].[dbo].[sargtest].[idx01]), SEEK:([archetype_rimarqu].[dbo].[s
|--Clustered Index Seek(OBJECT:([archetype_rimarqu].[dbo].[sargtest].[PK__sargtest__3213E83F785D3773])),
```

Now the same query but with non-SARGable argument:

```
SET SHOWPLAN_TEXT on
go
select * from sargtest where cast(col1 as varchar) = '04540F08-A62B-4F45-8E4A-BA67FE7A63B1'
```


The access to the data is done now through an Index Scan:

```
|--Nested Loops(Inner Join, OUTER REFERENCES:([archetype_rimarqu].[dbo].[sargtest].[id]))
|--Index Scan(OBJECT:([archetype_rimarqu].[dbo].[sargtest].[idx01]), WHERE:(CONVERT(varchar(30),[arche
|--Clustered Index Seek(OBJECT:([archetype_rimarqu].[dbo].[sargtest].[PK__sargtest__3213E83F785D3773])),
```

As we can see the application of functions on top of columns can make predicates Non-SARGable. This will include any function, like UPPER, ISNULL, CAST, LOWER, etc.

```
SET SHOWPLAN_TEXT on
go
select * from sargtest where LOWER(col1) = '04540F08-A62B-4F45-8E4A-BA67FE7A63B1'
```

```
|--Nested Loops(Inner Join, OUTER REFERENCES:([archetype_rimarqu].[dbo].[sargtest].[id]))
|--Index Scan(OBJECT:([archetype_rimarqu].[dbo].[sargtest].[idx01]), WHERE:(lower([archetype_rimarqu].
|--Clustered Index Seek(OBJECT:([archetype_rimarqu].[dbo].[sargtest].[PK__sargtest__3213E83F785D3773])),
```

Adding User functions, LIKE operator and others are also [included](#) 

How to mitigate

The issue is mainly from development perspective - customer issue. As support engineers we can't change customer queries.

The solution will always be related with a code or table structure change.

Anyway some ways to fix non-SARGable predicates for some example scenarios:

CAST or convert on top of the column

```
select * from table where cast(column as varchar) = 'aaaaa'
```

Perform the conversion on the value side

```
select * from table where column = cast('aaaaa' as nvarchar)
```

Or use a [persisted computed column](#) ☐. After this change the query so it uses the computed column on the predicate.

Adding two columns

```
select * from table where column1 + column2 = 55
```

Use a [persisted computed column](#) ☐. After this change the query so it uses the computed column on the predicate.

Using datetime functions

```
select * from table where YEAR(date) = '2022'
```

Change the query to:

```
select * from table where date between '20220101' and '20221231'
```

or use a [persisted computed column](#) ☐. After this change the query so it uses the computed column on the predicate.

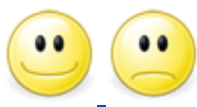
More information

[Search and destroy non sargable queries](#) ☐

[How to use sargable expressions in T-SQL queries; performance advantages and examples](#) ☐

[If You Can't Index It, It's Probably Not SARGable](#) ☐

How good have you found this content?



-