Slow query due to implicit conversion

Last updated by | Vitor Tomaz | Feb 24, 2023 at 3:28 AM PST

Contents

Issue

The implicit conversion occurs when, without user intervention, SQL Server needs to convert the data from one data type to another during query execution. It may cause a suboptimal execution plan and may consume between 30% and 50% of the CPU time. A typical example is when an NVARCHAR parameter is passed into a Where clause and the compared column has a data type of VARCHAR or CHAR.

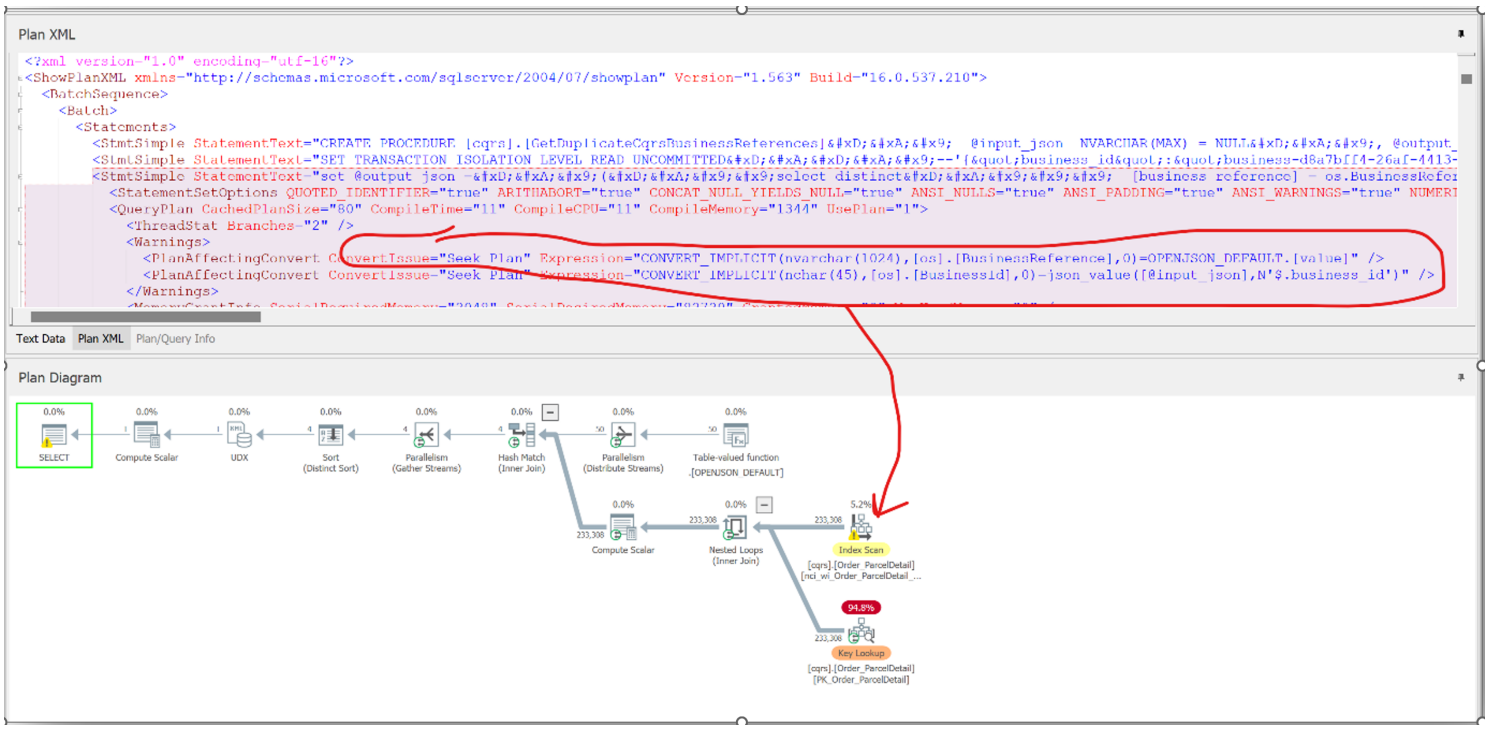
An implicit conversion may result in a table or clustered index scan because all column values in the table need to be evaluated by the data type conversion. If the table column has a different data type than the parameter used in query's parameters, the column values may need to be converted to the target data type before the rows are returned. This is preventing a more efficient index operation, e.g. an index seek. With SQL server performing a table or clustered index scan, the **avg_cpu_time_ms** and the **avg_logical_reads** can increase dramatically when performing the conversion.

The data type to be converted and the priority dependes on the data type precedence as described in Data type precedence (Transact-SQL) - SQL Server | Microsoft Docs. For example, NVARCHAR has a higher priority than VARCHAR hence the NVARCHAR parameter vs. VARCHAR column results in an implicit conversion of the table column.

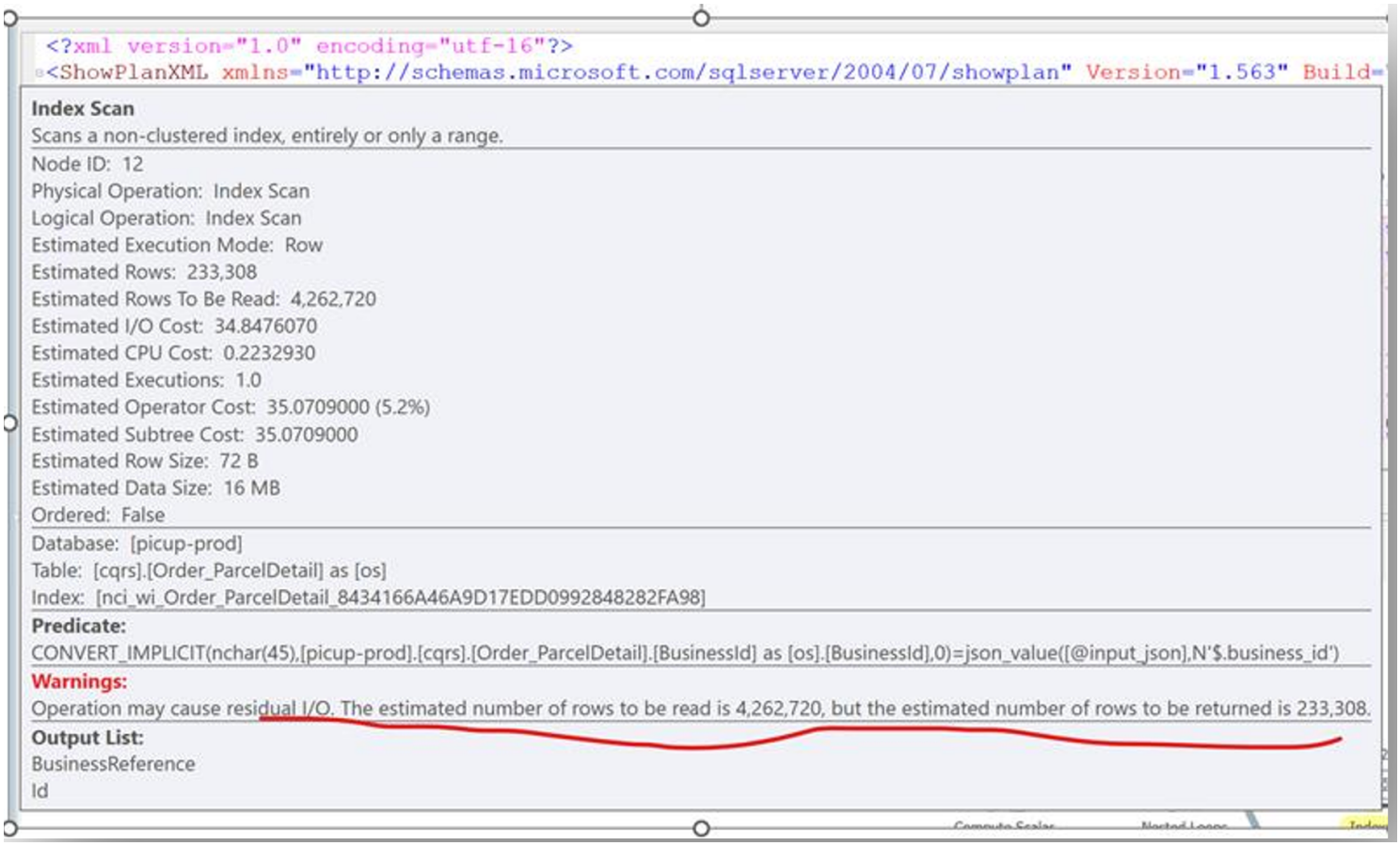
Implicit conversions also create a wrong cardinality estimation, thus contributing to a suboptimal execution plan.

Investigation / Analysis

Reading the execution plan (XML), it's possible to find this conversions by checking the "convert_implicit" expression.



And in the execution plan diagram it's possible to see the warning signal in the "Select" operation



Running the below query, is possible to get all the execution plan for all queries with implicit conversions, however is always important to understand and read the execution plan before recommend any change in customer query.

```
SELECT TOP(50) DB_NAME(t.[dbid]) AS [Database Name], t.text AS [Query Text],qs.total_worker_time AS [Total Wor
qs.total_worker_time/qs.execution_count AS [Avg Worker Time], qs.max_worker_time AS [Max Worker Time],
qs.total_elapsed_time/qs.execution_count AS [Avg Elapsed Time], qs.max_elapsed_time AS [Max Elapsed Time],
qs.total_logical_reads/qs.execution_count AS [Avg Logical Reads],qs.max_logical_reads AS [Max Logical Reads],
qs.execution_count AS [Execution Count], qs.creation_time AS [Creation Time],qp.query_plan AS [Query Plan]
FROM sys.dm_exec_query_stats AS qs WITH (NOLOCK)
CROSS APPLY sys.dm_exec_sql_text(sql_handle) AS t
CROSS APPLY sys.dm_exec_query_plan(plan_handle) AS qp
WHERE CAST(query_plan AS NVARCHAR(MAX)) LIKE ('%CONVERT_IMPLICIT%') AND t.[dbid] = DB_ID()
ORDER BY qs.total_worker_time DESC OPTION (RECOMPILE);
```

Mitigation

The potential solution / option is to make sure the application uses the same datatype in every tables and every queries, preventing the SQL server to perform this convert in query execution time.

How good have you found this content?