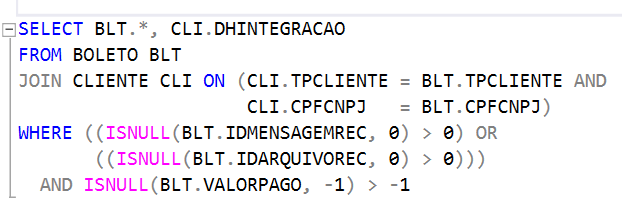
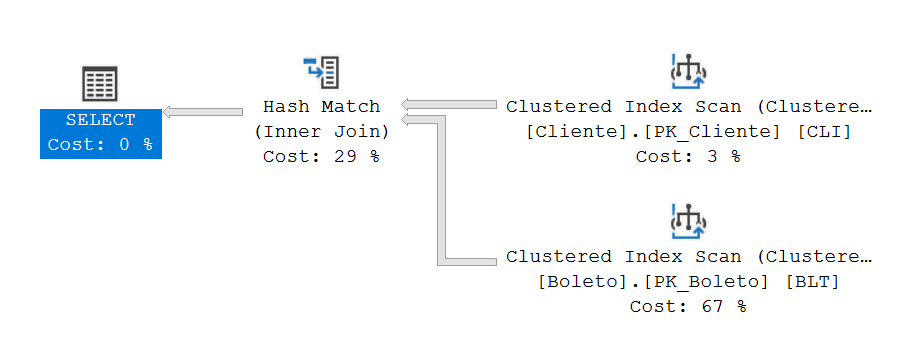
## Cenário 1 – Query 1

### Query 1



### Plano de execução

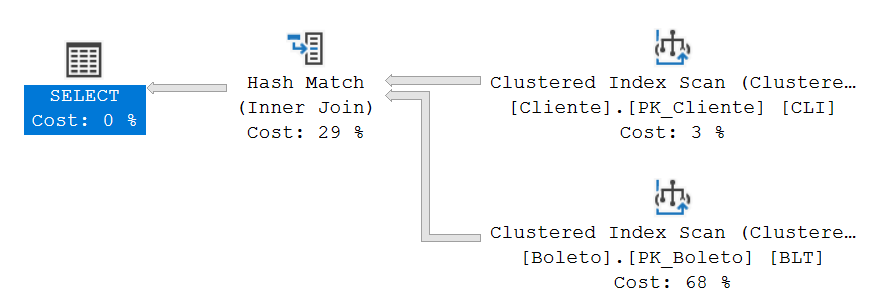


40.000 linhas retornadas em 30 segundos

## Cenário 2 – Query 1

Criação de índice (IDX\_BOLETO\_1) para os campos IDMENSAGEMREC, IDARQUIVOREC e VALORPAGO da tabela Boleto.

### Plano de execução

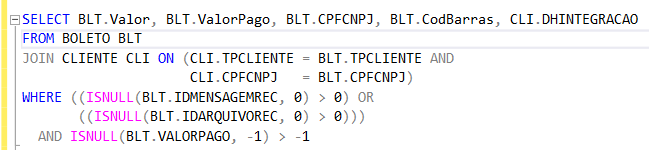


40.000 linhas retornadas em 23 segundos.

Apesar de uma pequena melhora na execução, o SQL Server não faz uso do índice que fora recém-criado.

## Cenário 3 – Query 1

### Limitando a projeção da tabela boleto para obter apenas 4 campos



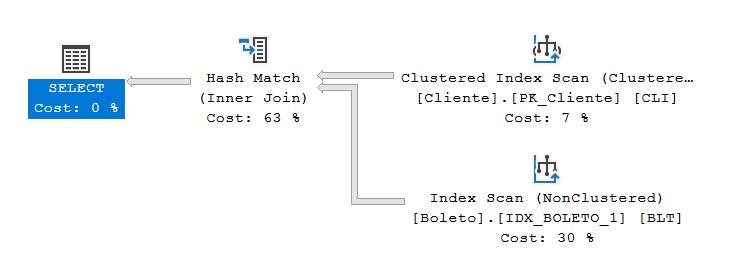
40.000 linhas retornadas em 6 segundos.

### Plano de execução

Apesar do visível ganho de desempenho, o índice continuou sendo ignorado pelo SQL Server.

## Cenário 4 – Query 1

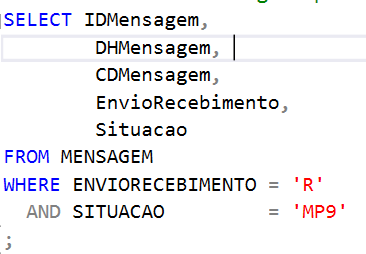
Adicionando os campos da projeção ao índice criado anteriormente, o plano de execução passa a usar o índice IDX\_BOLETO\_1 conforme a figura abaixo:



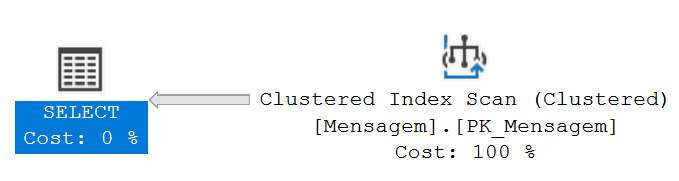
Aqui houve mais ganho na performance, passando a retornar 40.000 registros em 3 segundos.

## Cenário 4

### Query 2



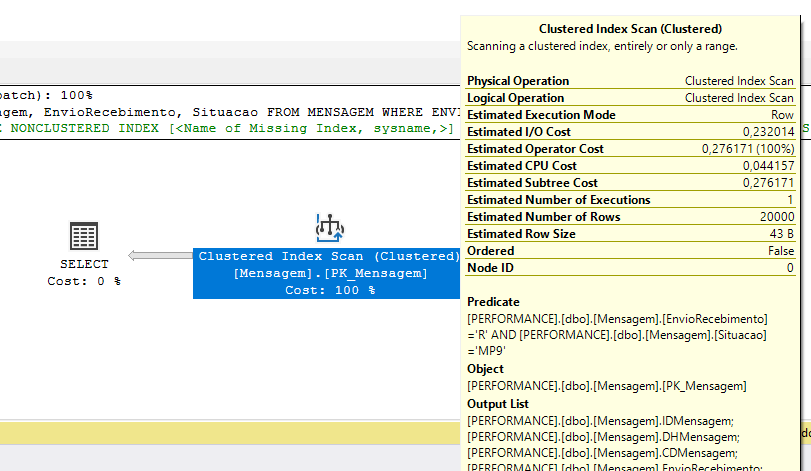
### Plano de execução



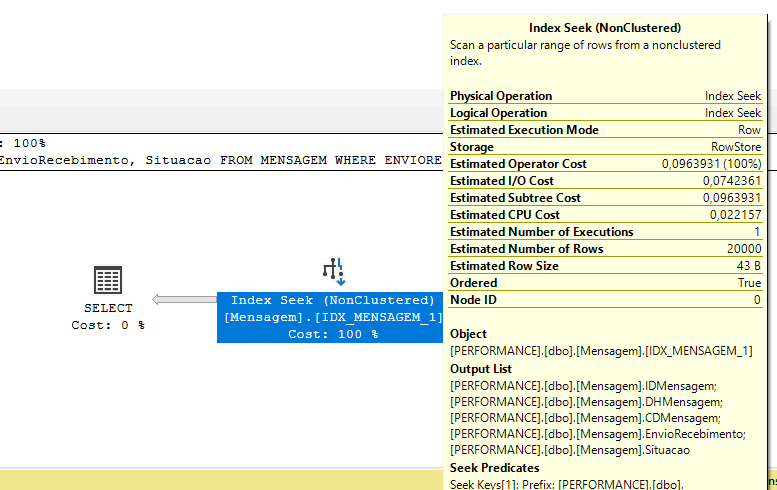
## Cenário 5 – Query 2

Criação do índice IDX\_MENSAGEM\_1 com os campos EnvioRecebimento e Situacao

Nota-se que o SQL Server continuou a não utilizar a query no plano de execução conforme a imagem abaixo:



Após adicionar os campos IDMensagem, DHMensagem e CDMensagem como Include Column no índice IDX\_MENSAGEM\_1, o mesmo passou a ser utilizado no plano de execução dando um ganho no custo de IO e Operador.



## Conclusão

Conforme vimos no exemplo acima, a otimização de query depende de vários fatores. Nem sempre a criação de índice irá garantir melhor performance na execução da query. Vimos também que a simples projeção de uma query, resulta numa execução otimizada mesmo quando não há o índice na tabela.