FIAP - 2018

SQL: Caso real de problemas de desempenho

ARQUITETURA DE BANCO DE DADOS E PERSISTÊNCIA

FIAP - 2018

SQL: Caso real de problemas de desempenho

ARQUITETURA DE BANCO DE DADOS E PERSISTÊNCIA

Nome	**RM**
Alexandre Dias Simões	41030
Anderson Costa de Jesus	42533
Jonathan Batista da Silva	43966
Marcos Paulo Oliveira Souza	41052

Sumário

1. Resumo

Nosso artigo é baseado em casos reais, com um cenário bem específico, mas atende todos os requisitos desde quando estamos iniciando um projeto, seja somente de banco de dados, ou envolvendo aplicações integradas, onde executa todas as funções importantes e corretas rapidamente.

O caso nos mostra que nem sempre o erro está ligando totalmente ao banco de dados, porém pode ir muito além do esperando, como iremos analisar, muitas vezes acreditamos somente em criação de index para otimizar o tempo retorno, e já está resolvido grande parte. Porém não é desta maneira que funciona, o problema pode estar muito mais longe do que imagina.

Utilizamos uma análise minuciosa de como o arquivo XML chega, e como é toda a tratativa de dados, conforme evidência podemos identificar as tabelas utilizadas, até chegar no conceito de que a aplicação também está causando lentidão no processo como um todo.

Logo conclui-se que não é somente realizar a criação de índice, ou até mesmo saber ao certo qual é o tempo correto para a transação no banco ser finalizada, por fim é sempre necessário integrar todas as camadas de uma aplicação e suas diversas responsabilidades, pois assim será possível um gerenciamento de tarefas, e juntos podemos analisar parte a parte.

2. Introdução

Com o crescimento digital, tem-se aumentado muito a quantidade de dados e o quão rápido e/ou ágil uma informação deve chegar ao consumidor final (usuário), seja em ambiente corporativo ou não, quanto mais ágil e eficiente um sistema interage, ou seja, quanto mais performático, melhor para o usuário e os profissionais de tecnologia e desenvolvimento tem um grande desafio quando o assunto é performance e principalmente quando tem relação a dados. A cada vez que alguém utiliza um sistema, o mesmo está gerando dados que consequentemente geram informações e tudo isso é armazenado em um "Banco de dados".

Devido a quantidade de informações que estão sendo geradas e armazenadas, existe um profissional que é responsável por cuidar da qualidade e eficiência dos dados e do local estão armazenados e o nome deste profissional é DBA (*Database Administrator*). Para garantir que um dado esteja sendo armazenado da melhor forma possível, o DBA possuí diversas ferramentas disponíveis e a mais conhecida é o SGBD (Sistema de gerenciamento de banco de dados), ele é responsável por fornecer uma interface e opções para gerenciamento e tomada de decisões principalmente tratando-se de performance.

Este artigo tem o intuito de mostrar a importância que as boas práticas e técnicas de criação e manutenção de banco de dados tem dentro do meio das corporações, além das consequências e custos que o mal cuidado dos dados podem trazer e para isso, foram utilizadas ferramentas criadas pela empresa Microsoft que são elas o Sql Server (Banco de dados) e Sql Server Management Studio (SGBD) e com essas ferramentas podemos analisar o desempenho de consultas, alterar o banco e criar formas de torna-las mais rápidas, utilizando índices e melhorando a escrita (*Query*) da consulta. Para que tal trabalho fosse possível, foi utilizado um cenário real, em uma empresa onde há a necessidade de indicar ao cliente que foi gerado um boleto bancário para ele e, ao pagar, a aplicação deve comunicar ao BACEN (Banco central do Brasil) a nova situação do boleto emitido. Essas informações são enviadas através de um serviço em forma de fila e, todo conteúdo que chega através deste serviço é processada e armazenada pelo banco de dados, porém, devido à alta demanda de clientes, o sistema começou a apresentar problemas de desempenho o que gerou muitas reclamações dos usuários e dentro deste cenário será desenvolvido o tema deste artigo afim de contextualizar e evidenciar o leitor sobre alguns problemas de banco de dados, como analisar e solucionar os mesmos.

3. Materiais e métodos

3.1. Ferramentas

Neste artigo foi utilizado o seguinte ambiente:

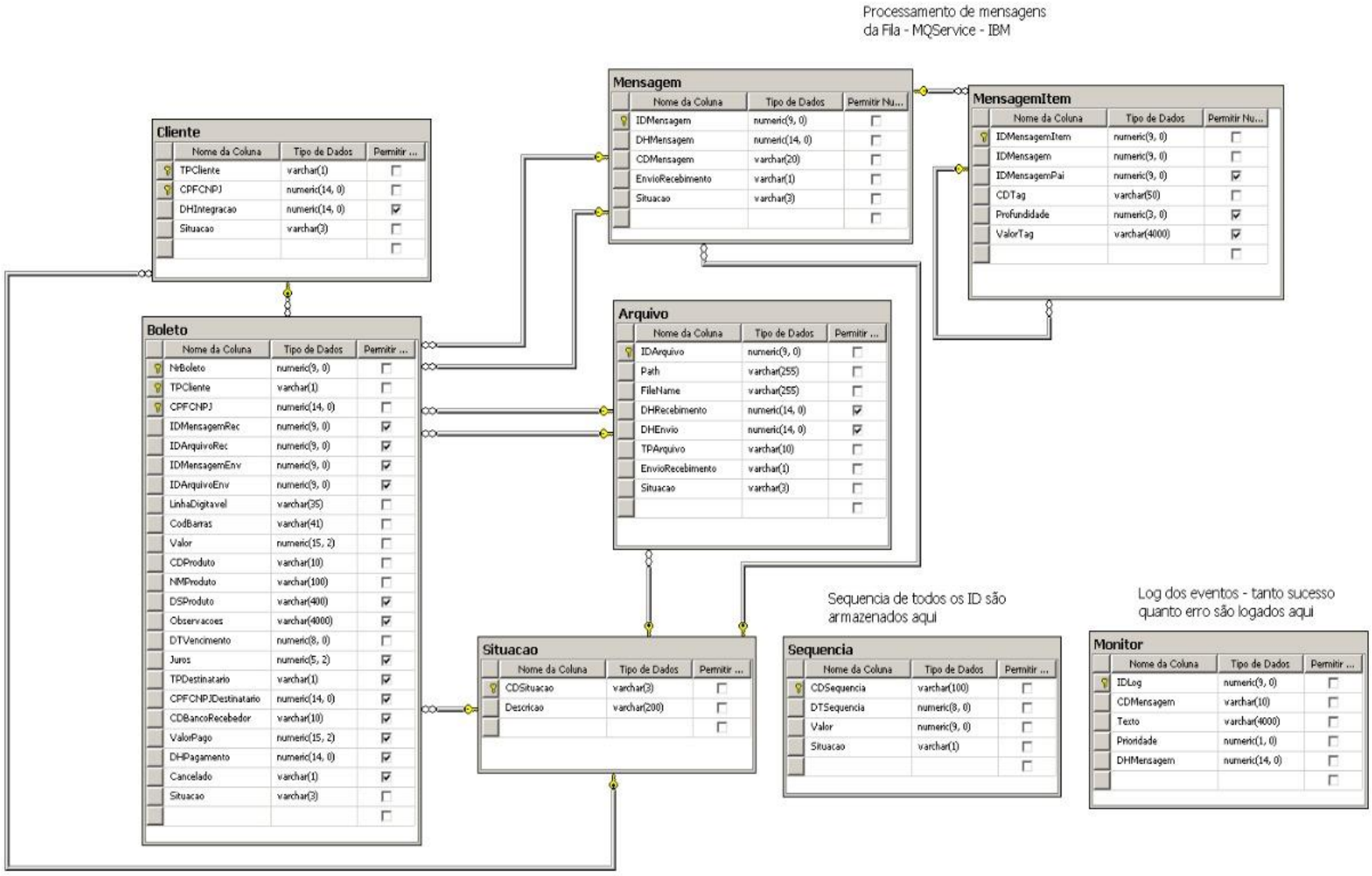
- Sistema utilizado neste artigo foi feito para simular o comportamento do sistema real;
- SQL Server 2016;
- Windows 7;
- Processador: Intel I7;
- RAM: 8GB

3.2. Cenário

O artigo a seguir, foi baseado em um cenário real, onde o BACEN envia mensagens através de uma fila (IBM MQ) e de arquivos xml trafegadas via "connect direct".

O software retira as mensagens da fila/connect-direct, trata, valida e realiza o processo de normalização de dados que consiste em tratar os dados de forma que seja possível armazena-los em sua respectiva tabela, referenciando em cada campo o seu respectivo dado e criando os devidos relacionamentos, essas mensagens/arquivos são boletos bancários tendo grandes volumes de dados que é armazenado nesse banco e com essa grande massa, uma consulta simples pode resultar em um grande problema de desempenho.

Segue o Modelo de Entidade e Relacionamentos (MER):


Processamento de mensagens
da Fila - MQService - IBM
Mensagem
Nome da Coluna
Tipo de Dados
Permitir Nu...
IDMensagem
numeric(9, 0)
DHMensagem
numeric(14, 0)
CDMensagem
varchar(20)
EnvioRecebimento
varchar(1)
Situacao
varchar(3)
MensagemItem
Nome da Coluna
Tipo de Dados
Permitir Nu...
IDMensagemItem
numeric(9, 0)
IDMensagem
numeric(9, 0)
IDMensagemPai
numeric(9, 0)
CDTag
varchar(50)
Profundidade
numeric(3, 0)
ValorTag
varchar(4000)
Cliente
Nome da Coluna
Tipo de Dados
Permitir ...
TPCliente
varchar(1)
CPFCNPJ
numeric(14, 0)
DHIntegracao
numeric(14, 0)
Situacao
varchar(3)
Arquivo
Nome da Coluna
Tipo de Dados
Permitir ...
IDArquivo
numeric(9, 0)
Path
varchar(255)
FileName
varchar(255)
DHRecebimento
numeric(14, 0)
DHEnvio
numeric(14, 0)
TPArquivo
varchar(10)
EnvioRecebimento
varchar(1)
Situacao
varchar(3)
Boleto
Nome da Coluna
Tipo de Dados
Permitir ...
NrBoleto
numeric(9, 0)
TPCliente
varchar(1)
CPFCNPJ
numeric(14, 0)
IDMensagemRec
numeric(9, 0)
IDArquivoRec
numeric(9, 0)
IDMensagemEnv
numeric(9, 0)
IDArquivoEnv
numeric(9, 0)
LinhaDigitavel
varchar(35)
CodBarras
varchar(41)
Valor
numeric(15, 2)
CDProduto
varchar(10)
NMProduto
varchar(100)
DSProduto
varchar(400)
Observacoes
varchar(4000)
DTVencimento
numeric(8, 0)
Juros
numeric(5, 2)
TPDestinatario
varchar(1)
CPFCNPJDestinatario
numeric(14, 0)
CDBancoRecebedor
varchar(10)
ValorPago
numeric(15, 2)
DHPagamento
numeric(14, 0)
Cancelado
varchar(1)
Situacao
varchar(3)
Situacao
Nome da Coluna
Tipo de Dados
Permitir ...
CDSituacao
varchar(3)
Descricao
varchar(200)
Sequencia de todos os ID são
armazenados aqui
Sequencia
Nome da Coluna
Tipo de Dados
Permitir ...
CDSequencia
varchar(100)
DTSequencia
numeric(8, 0)
Valor
numeric(9, 0)
Situacao
varchar(1)
Log dos eventos - tanto sucesso
quanto erro são logados aqui
Monitor
Nome da Coluna
Tipo de Dados
Permitir ...
IDLog
numeric(9, 0)
CDMensagem
varchar(10)
Texto
varchar(4000)
Prioridade
numeric(1, 0)
DHMensagem
numeric(14, 0)

3.3. SQL Structured Query Language

A forma de recuperar os dados armazenados, são através de consultas chamadas de query, onde existem diversas formas de escrever respeitando a estrutura estabelecida pela empresa fabricante do servidor de banco de dados. Todos os exemplos utilizados neste artigo, foram construídos com base na estrutura do SQL Server fornecido pela Microsoft.

O programa mencionado anteriormente, deve disponibilizar os dados dos boletos para consulta dos usuários e, para tal atividade e análise do artigo, será utilizada a seguinte consulta abaixo:

Query 1:

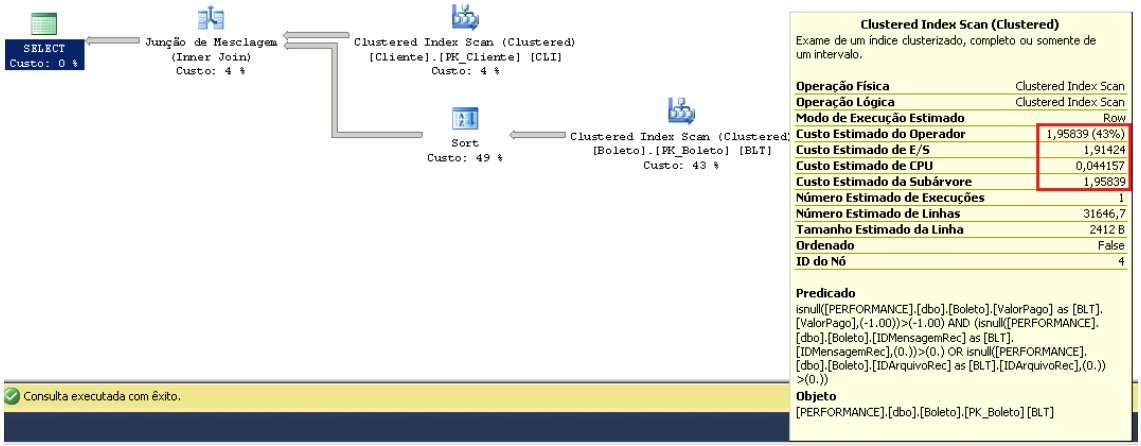
```
--BOLETOS RECEBIDOS QUE JÁ FORAM PAGOS
SELECT BLT.*, CLI.DHINTEGRACAO
FROM BOLETO BLT
JOIN CLIENTE CLI ON (CLI.TPCLIENTE = BLT.TPCLIENTE AND
                     CLI.CPFCNPJ   = BLT.CPFCNPJ)
WHERE ((ISNULL(BLT.IDMENSAGEMREC, 0) > 0) OR
       ((ISNULL(BLT.IDARQUIVOREC, 0) > 0)))
  AND ISNULL(BLT.VALORPAGO, -1) > -1
```

3.4. Performance de consultas

Toda consulta feita em banco de dados possui um “custo” de plano de execução e, este plano descreve como a sequência de operações físicas e lógicas, que o servidor terá que realizar para que os dados sejam retornados. O mecanismo de consultas escolhe o melhor plano de execução baseado em estatísticas, quantidade de dados dentre outros fatores e assim o SQL Server escolhe aquele que possui o menor custo. O Sql Server Management Studio possui recursos onde podemos visualizar graficamente o plano de execução e as estimativas de tempo no qual podemos tirar algumas conclusões sobre o quão eficiente as consultas estão sendo realizadas.

A imagem abaixo, representa o plano de execução para query mencionada acima:

Plano de execução:



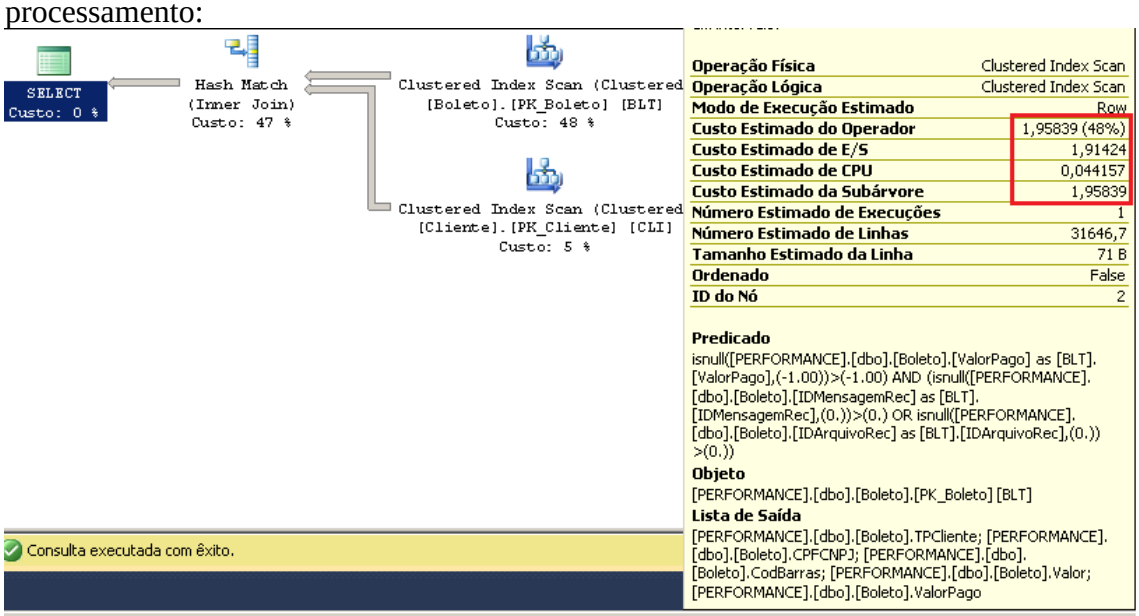
Observa-se que o plano acima teve um alto custo de processamento.

4. Resultado

Analisando melhor a query, notamos que existe um **BLT.*** que traz todos os campos da tabela Boleto, aumentando o custo de forma excessiva. Alteramos a query para fazer uma projeção melhor dos campos que realmente importam:

```
SELECT BLT.Valor, BLT.ValorPago, BLT.CPFCNPJ, BLT.CodBarras, CLI.DHINTEGRACAO
FROM BOLETO BLT
JOIN CLIENTE CLI ON (CLI.TPCLIENTE = BLT.TPCLIENTE AND
                     CLI.CPFCNPJ   = BLT.CPFCNPJ)
WHERE ((ISNULL(BLT.IDMENSAGEMREC, 0) > 0) OR
       ((ISNULL(BLT.IDARQUIVOREC, 0) > 0)))
  AND ISNULL(BLT.VALORPAGO, -1) > -1
```

Com isso, nosso plano de execução mudou, porém, ainda com custos altos de processamento:



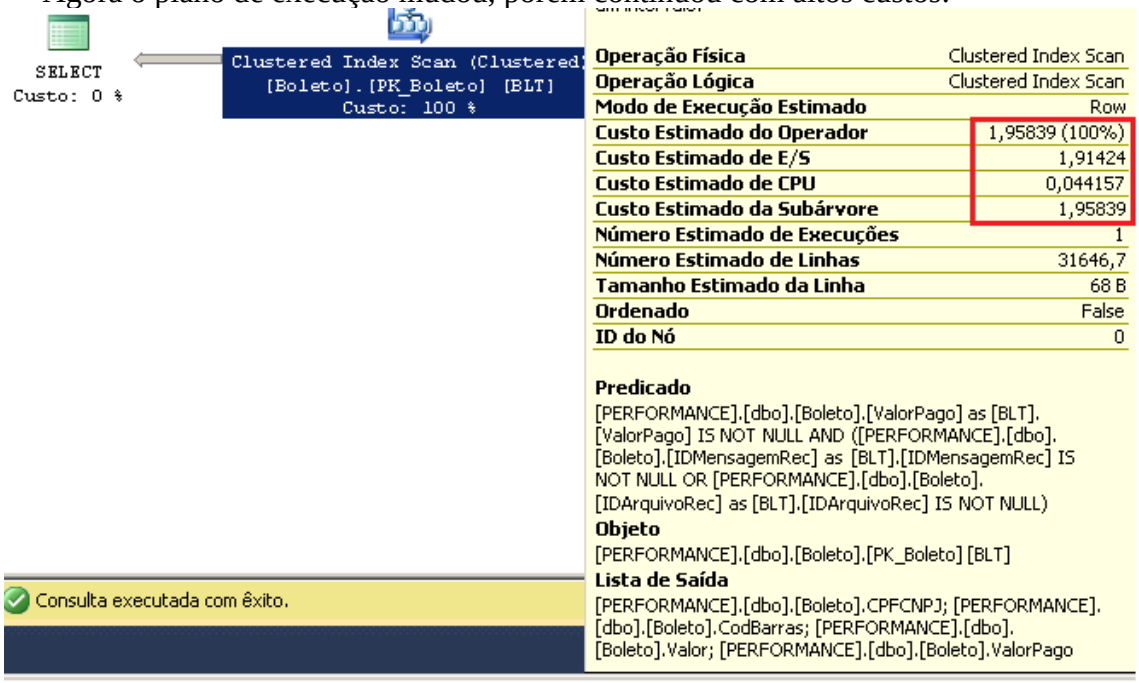
Ainda visando melhorias na query, vemos as condições “where” e percebemos que a consulta está usando funções diretamente nos campos da tabela, o que atrapalha muito o processamento do SQL Server. Portanto, mudamos a consulta para uma outra forma, permitindo um melhor processamento:

```
SELECT BLT.Valor, BLT.ValorPago, BLT.CPFCNPJ, BLT.CodBarras, CLI.DHINTEGRACAO
FROM BOLETO BLT
JOIN CLIENTE CLI ON (CLI.TPCLIENTE = BLT.TPCLIENTE AND
                     CLI.CPFCNPJ   = BLT.CPFCNPJ)
WHERE ((BLT.IDMENSAGEMREC IS NOT NULL) OR
       (BLT.IDARQUIVOREC  IS NOT NULL))
  AND BLT.VALORPAGO IS NOT NULL
```

O plano de execução permanece inalterado, porém podemos começar a pensar em utilizar index. Mas antes vamos analisar a real necessidade de fazer o “JOIN” com a tabela de “Cliente” para trazer apenas a data de integração.
Juntamente com a equipe, observamos que esse campo “DHINTEGRACAO” (data de integração do cliente) não é utilizado em nenhum lugar, portanto retiramos o “JOIN” com a tabela “Cliente”:

```
SELECT BLT.Valor, BLT.ValorPago, BLT.CPFCNPJ, BLT.CodBarras
FROM BOLETO BLT
WHERE ((BLT.IDMENSAGEMREC IS NOT NULL) OR
       (BLT.IDARQUIVOREC  IS NOT NULL))
  AND BLT.VALORPAGO IS NOT NULL
```

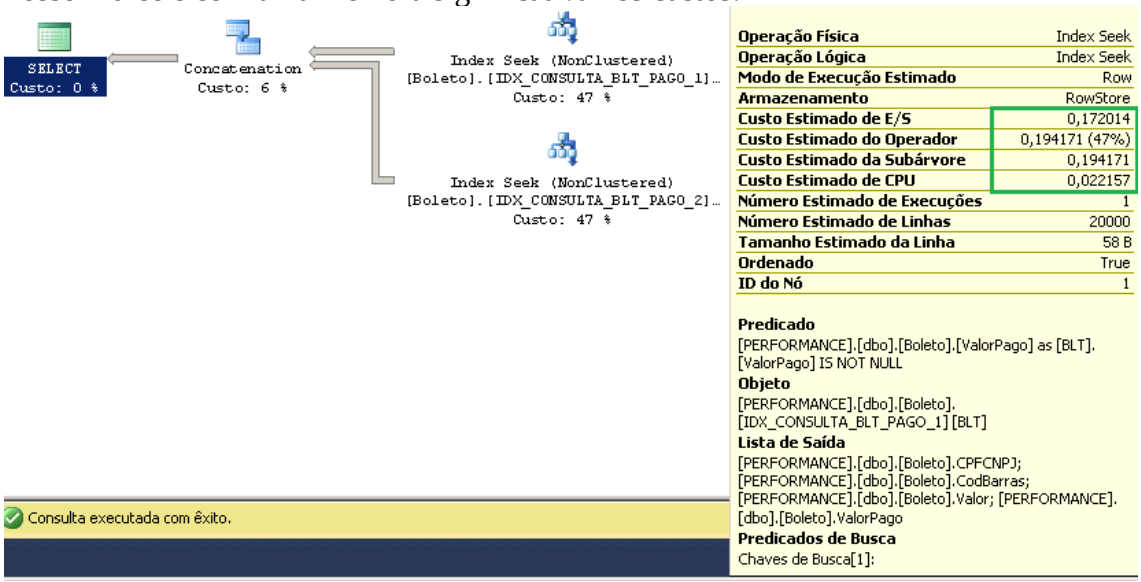
Agora o plano de execução mudou, porém continuou com altos custos:



Por fim, a alteração da query para que fizesse um “UNION ALL” ao invés de utilizar o operador “OR” pois este atrapalha muito o processamento. Em seguida criamos dois índices, um para “IDMENSAGEMREC e ValorPago” e outro para “IDARQUIVOREC e ValorPago”:

```
CREATE NONCLUSTERED INDEX IDX_CONSULTA_BLT_PAGO_1 ON Boleto (IDMENSAGEMREC,
ValorPago) INCLUDE (Valor, CPFCNPJ, CodBarras);

CREATE NONCLUSTERED INDEX IDX_CONSULTA_BLT_PAGO_2 ON Boleto (IDARQUIVOREC,
ValorPago) INCLUDE (Valor, CPFCNPJ, CodBarras);

SELECT BLT.Valor, BLT.ValorPago, BLT.CPFCNPJ, BLT.CodBarras
FROM BOLETO BLT
WHERE BLT.IDMENSAGEMREC IS NOT NULL
  AND BLT.VALORPAGO     IS NOT NULL

UNION ALL

SELECT BLT.Valor, BLT.ValorPago, BLT.CPFCNPJ, BLT.CodBarras
FROM BOLETO BLT
WHERE BLT.IDARQUIVOREC IS NOT NULL
  AND BLT.VALORPAGO    IS NOT NULL
```

Com essas alterações, tivemos um novo plano de execução, dessa vez utilizando nosso índice e com uma melhora significativa nos custos:



Conforme vimos nesses testes, a otimização de query depende de vários fatores. Nem sempre a criação de índice irá garantir melhor performance na execução. Vimos também que a simples projeção de uma query resulta numa execução otimizada, mesmo quando não há o índice na tabela.

4.1. Processamento dos boletos

Segundo o log do aplicativo de testes, percebemos que a integração de cliente foi feita em 685ms, porém o processamento das mensagens foi em 03:04.567 minutos. Temos um problema nesse ponto, ao analisarmos as queries percebemos que todas já estão otimizadas, usando índice e etc.

```
11:16:06.389 -> Inicio : Adicionando clientes
11:16:07.024 -> FIM: 10000 -> 00:00:00.625
------------------------------
11:16:07.039 -> Inicio : Adicionando mensagens
11:19:11.608 -> FIM: 70000 -> 00:03:04.567
------------------------------
11:19:11.635 -> Inicio : Adicionando arquivos
11:19:34.587 -> FIM: 10000 -> 00:00:22.950
------------------------------
```

Ao entendermos o que o software faz, identificamos que a cada loop o sistema solicita uma nova sequência, fazendo um update e um select em seguida em uma transação a parte.

Esse algoritmo funciona para pequenos registros, mas para um volume como esse é gerado um gargalo enorme. Após várias discussões chegamos à uma ideia: Sequência em lote, ou seja, o sistema já reserva um lote de sequencias e controla internamente, retirando a necessidade de ir no banco a cada loop.

Após as alterações no sistema para utilizar a sequência em lote, executamos o processamento novamente e a diferença de processamento foi enorme, conforme o log:

```
11:46:15.928 -> Inicio : Adicionando clientes
11:46:16.466 -> FIM: 10000 -> 00:00:00.536
------------------------------
11:46:16.491 -> Inicio : Adicionando mensagens
11:46:19.777 -> FIM: 70000 -> 00:00:03.285
------------------------------
11:46:19.802 -> Inicio : Adicionando arquivos
11:46:20.508 -> FIM: 10000 -> 00:00:00.701
------------------------------
```

5. Conclusão

São necessárias várias análises para uma real melhora das consultas, o que depende de n fatores como construção da query, projeção dos campos, campos que realmente são utilizados, utilização de funções nas condições (where) e etc., pois nem sempre apenas a criação do index afeta o resultado.

Outro ponto é pensar que todos problemas estão no banco de dados, em vários casos o software foi mal escrito e utiliza o banco de dados de forma errada.

Por fim, devemos sempre analisar os custos dos operadores, pois se formos nos basear no tempo de resposta da query podemos nos enganar porque o SQL Server utiliza “cache” para devolver as próximas consultas.

6. Referências

Como otimizar o desempenho da consulta do SQL Server. Pilecki, Maciej (Novembro, 2007). Disponível em: https://technet.microsoft.com/pt-br/library/2007.11.sqlquery.aspx

Processamento de instruções SQL. Pilecki, Maciej (2007). Disponível em: https://technet.microsoft.com/pt-br/library/ms190623(v=sql.105).aspx