|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | **[Dica do Mês – Analisando o comportamento do comando DBCC Clean Table.](https://public-api.wordpress.com/bar/?stat=groovemails-events&bin=wpcom_email_click&redirect_to=https%3A%2F%2Fpedrogalvaojunior.wordpress.com%2F2019%2F05%2F28%2Fdica-do-mes-analisando-o-comportamento-do-comando-dbcc-clean-table%2F&sr=1&signature=0fcfeeabdd304ab078b8d30bea30ba38&user=150005308&_e=&_z=z)** por [Junior Galvão - MVP](https://public-api.wordpress.com/bar/?stat=groovemails-events&bin=wpcom_email_click&redirect_to=https%3A%2F%2Fpedrogalvaojunior.wordpress.com%2Fauthor%2Fpedrogalvaojunior%2F&sr=1&signature=85433cdf42bee2ceecda8c56dd1e624e&user=150005308&_e=&_z=z) |   Olá, bom dia. Tudo bem?  O frio já chegou na sua região? Aqui em São Roque, interior de hoje esta manhã esta sendo considerada até o presente momento a mais fria do ano (eu particularmente adoro o frio).  Fico extremamente contente e honrado com a sua visita ao meu blog, mesmo com todo este frio ter a sua presença aqui é muito importante, ainda mais neste post da sessão **Dica do Mês,** a qual foi criada á alguns anos com objetivo de compartilhar algo que possa ser considerada como uma dica ou melhores práticas para se trabalhar na área de banco de dados, mais especificamente falando relacionadas ao Microsoft SQL Server.  Posso dizer que o post de hoje é algo bastante simples, vamos conhecer um pouco mais sobre um dos mais tradicioais comandos pertencentes a categoria DBCC - Database Command Console existente no Microsoft SQL Server desde suas versões iniciais.  Estou me referindo ao comando DBCC CleanTable, considerado por muitos DBAs o "Veja, aquele produto de limpeza que utilizamos para tirar a gordura dos fogões e panelas", ele faz basicamente isso em nossas tabelas. No decorrer deste post vou tentar mostrar como Podemos fazer isso.  Sendo assim, sem mais delongas, vamos em frente, vou tentar mitigar a sua curiosidade e ao mesmo também satisfazer os meus objetivos. Seja bem-vindo ao post – **Dica do Mês - Analisando o comportamento do comando DBCC Clean Table.** **Introdução** O Microsoft SQL Server apresenta em sua galeria de comandos um conjunto muito particular e exclusivo denominado DBCC - Database Command Console em uma possível tradução para o Português "*Comandos de Console de Banco de Dados*", pois bem, este conjunto com categoria de comandos é composta por uma relação bem interessante e diversificada que nos permite fazer uso em diversos cenários, dentre eles por exemplos os comandos: **DBCC CheckDB** ou **DBCC ShrinkFile**, ambos muito conhecidos pelos DBAs e Administradores de Banco de Dados, caso você ainda não conheça, com certeza em algum momento terá a oportunidade.  Como já destaquei nesta relação exclusiva de comandos, podemos se deparar com um deles que nos permite aplicar sem qualquer tipo de alteração de configuração, ou mudanças drásticas na estrutura de nosso banco de dados, o que chamamos de remover a gordura de nossas tabelas, talvez você não consiga ou não esteja entendendo o que eu estou definindo como "gordura", na verdade me refiro por exemplo a aquelas colunas do tipo de dados VarChar, criadas inicialmente em nossas tabelas, mas que ao longo do tempo se tornam colunas praticamente consideradoras **espúrias** (algo sem sentido ou desnecessário), é ai que entra o nosso amigo DBCC CleanTable, ele tem um papel único e específico justamente para este tipo de cenário, que daqui a pouco eu vou demonstrar, mas antes vamos conhecer um pouquinho sobre este comando. **DBCC CleanTable** Adicionado ao Microsoft SQL Server a partir da edição 2008, o comando DBCC CleanTable possui como papel principal a capacidade de recuperar e liberar o espaço ocupado por colunas existentes em uma tabela consideradas colunas com comprimento variável quando utilizam os seguintes tipos de dados:   * **Varchar**; * **Nvarchar**; * **Varchar(max)**; * **Nvarchar(max)**; * **Varbinary**; * **Varbinary(max)**; * **Text**; * **Ntext**; * **Image**; * **Sql\_variant;** e * **XML**.   Ele recupera espaço anteriormente ocupado por um destes tipos de dados, depois que uma coluna de comprimento variável é descartada, mas não recupera espaço depois que uma coluna de comprimento fixo é descartada.  Mesmo não tem a capacidade de recuperar o espaço ocupado em disco logo após este limpeza na estrutura da tabela, o comando DBCC CleanTable, pode ajudar a melhorar ou até mesmo zerar taxas de fragmentação que possam estar sendo apresentadas justamente em uma tabela devido a utilização destas colunas com comprimentos variáveis.  **Considerações**   * As colunas descartadas forem armazenadas em linha, DBCC CLEANTABLE recuperará espaço da unidade de alocação IN\_ROW\_DATA da tabela. * Quando as colunas forem armazenadas fora de linha, o espaço será recuperado da unidade de alocação LOB\_DATA ou ROW\_OVERFLOW\_DATA, dependendo do tipo de dados da coluna descartada.Se o espaço recuperado de uma página ROW\_OVERFLOW\_DATA ou LOB\_DATA resultar em uma página vazia, DBCC CLEANTABLE removerá a página. DBCC CLEANTABLE executa como uma ou mais transações. * O espaço recuperado de uma página ROW\_OVERFLOW\_DATA ou LOB\_DATA resultar em uma página vazia, DBCC CLEANTABLE removerá a página. DBCC CLEANTABLE executa como uma -ou mais transações. * Caso não especificado um tamanho de lote, o comando processará a tabela inteira em uma transação e a tabela será bloqueada exclusivamente durante a operação. Para algumas tabelas grandes, o comprimento da única transação e o espaço do log requeridos podem ser muito grandes. Se um tamanho de lote for especificado, o comando executará em uma série de transações, cada qual incluindo o número especificado de linhas. * O comando DBCC CLEANTABLE não pode ser executado como uma transação dentro de outra transação. Essa operação é totalmente registrada. Não há suporte para DBCC CLEANTABLE para uso em tabelas do sistema, tabelas temporárias ou a parte do índice columnstore xVelocity de memória otimizada de uma tabela.   **Evite utilizar**  O comando DBCC CLEANTABLE não deve ser executado como uma tarefa de manutenção de rotina. Ao invés disso, utilize o DBCC CLEANTABLE depois de fazer mudanças significativas em colunas de comprimento variável em uma tabela ou exibição indexada e necessita recuperar o espaço sem-uso (considerada como área não alocada) prontamente.  **Alternativas para o uso do DBCC CleanTable**  Uma das possíveis alternativas quando desejamos recuperar o espaço ocupado por colunas de comprimemto variável aplica-se a reconstrução de índices em tabelas ou visões indexadas, mas este recurso pode ser considerado custoso no que se relaciona ao tempo de processamento ou até mesmo alocação de recursos durante sua execução.  Pois bem, agora que já conhecemos um pouco sobre este comando, suas considerações, o quando usar e não usar, vamos avançar um pouco este post, dando início a nosso cenário de estudos afim de analisarmos de uma forma bem simples e artificial como o DBCC CleanTable pode nos ser útil. **NOSSO AMBIENTE** Como de costume vamos utilizar um ambiente isolado dos demais bancos de dados que você possa conter, desta maneira nosso cenário será constituído dos seguintes elementos:   * Banco de Dados: **TesteDBCCCleanTable**; * Arquivo Texto: [**QueimadasCleanTable (100 MBs - Dados - Queimadas 2002)**](https://1drv.ms/x/s!AjrQ6-4-Es6JuhHy4Vz3Se4wmgUV); e * Tabela: **QueimadasCleanTable**.   **Observações**   1. Estaremos fazendo uso de um arquivo texto, o qual é parte importante para nossa análise. O mesmo possui os dados que serão importados para o Microsoft SQL Server, contendo com conjunto real de valores coletados através do portal do [INPE](http://www.inpe.br/queimadas/bdqueimadas/) - **Instituto Nacional de Pesquisas Especiais**, através de seu Banco de Dados de análise de queimadas ocorridas no Brasil ao longo dos últimos 70 anos, sendo considerada uma das mais importantes fontes de dados abertos do Brasil, disponível para qualquer tipo de análise. Particularmente falando, sou um grande admirador do trabalho realizado pelo INPE, o qual eu comecei em 2017 a estudar para um dos meus projetos no mestrado e diretamente relacionados com a minha nova área de estudos: Data Warehouse, Data Mining e BI. 2. A estrutura apresentada no arquivo texto, não tem por finalidade ou regra ser considerada uma estrutura padronizada, como também, algo que atenda as regras da Normalização, longe disso, todo conjunto de dados, tipos de dados, nomes das colunas e sua composição física e lógica foi definida e criada para atender as regras de negócio muito específicas de um trabalho de mestrado acadêmico, o qual não faz parte deste estudo ou análise. 3. O objetivo deste post não se realiza a apresentar, demonstrar ou orientar como realizar o procedimento de importação de dados para o Microsoft SQL Server, sendo assim, caso você tenha dúvidas ou dificuldades para realizar este procedimento, acesse: [**Import and Export Data with the SQL Server Import and Export Wizard**](https://docs.microsoft.com/en-us/sql/integration-services/import-export-data/import-and-export-data-with-the-sql-server-import-and-export-wizard?view=sql-server-2017)**.** 4. Logo após a execução do processo de importação dos dados, você notará que nossa tabela e sua estrutura foi criada sem respeitar uma análise de uso de tipos de dados, criação de chaves primárias ou outras considerações que podemos definir como melhores práticas de modelagem de banco de dados. Não foi fique preocupado, pois estaremos realizando toda esta reestruturação logo na sequência.   Avançando mais um pouco, vamos criar nosso banco de dados e logo na sequência começarmos nossa análise, para tal procedimentos, utilizaremos o *Bloco de Código 1* abaixo:  **— Bloco de Código 1 - Criando nosso cenário —**  **— Criando o Banco de Dados —**  Create Database TesteDBCCCleanTable  Go  **— Acessando o Banco de Dados —**  Use TesteDBCCCleanTable  Go  Presumo que neste momento você já tenha feito download do arquivo QueimadasTableCleanTable.txt, como também, já tenha realizado a importação dos dados e criação da tabela,.  Ótimo, espero que todo processo de importação de dados tenha ocorrido corretamente, agora com a estrutura criada e acessível, teremos a possibilidade de começar a realizar nossa análise, nosso próximo passo será reestrutura a tabela QueimadasCleanTable, definindo sua chave primária, alterando tipos de dados em determinadas colunas, removendo outras, enfim colocando um pouco de ordem na casa.  Desta forma, vamos utilizar o *Bloco de Código 2* a seguir:  **— Bloco de Código 2 - Reestruturando a Tabela QueimadasCleanTable —**  **-- Remover a Anulabilidade da coluna CodigoQueimada --**  Alter Table QueimadasCleanTable  Alter Column CodigoQueimada Int Not Null  Go  **-- Adicionar a coluna chave primária na Tabela QueimadasCleanTable --**  Alter Table QueimadasCleanTable  Add Constraint [PK\_QueimadasCleanTable\_Codigo]  Primary Key (CodigoQueimada)  Go  **-- Alterando o Tamanho e Tipo de Dados da Coluna Pais --**  Alter Table QueimadasCleanTable  Alter Column Pais Char(6) Not Null  Go  **-- Alterando os tipos de dados e tamanho da coluna Satelite --**  Alter Table QueimadasCleanTable  Alter Column Satelite Varchar(10) Not Null  Go  **-- Alterando os tipos de dados e tamanho das colunas --**  Alter Table QueimadasCleanTable  Alter Column Municipio Varchar(40) Not Null  Go  Alter Table QueimadasCleanTable  Alter Column Estado Varchar(20) Not Null  Go  Alter Table QueimadasCleanTable  Alter Column Bioma Varchar(15) Not Null  Go  **-- Alterando o formato do dado armazenado na coluna DataHora --**  Update QueimadasCleanTable  Set DataHora=Convert(DateTime, DataHora, 102)  Go  **-- Alterando a Coluna DataHora --**  Alter Table QueimadasCleanTable  Alter Column DataHora DateTime Not Null  Go  **-- Alterando a Coluna Longitude --**  Alter Table QueimadasCleanTable  Alter Column Longitude Numeric(10,5) Not Null  Go  **-- Alterando a Coluna Latitude --**  Alter Table QueimadasCleanTable  Alter Column Latitude Numeric(10,5) Not Null  Go  Ufa, após este longo caminho percorrido, nossa tabela QueimadasCleanTable, deve estar apresentando uma estrutura similar a *Figura 1*:    *Figura 1 - Banco de Dados TesteDBCCCleanTable e Tabela QueimadasCleanTable criados.*  Sensacional, agora a brincadeira vai começar, devemos ter basicamente 752.252 (Setecentas e cinquenta e duas mil, duzentas e cinquenta e duas) linhas de registros lógicos inseridas nesta tabela, uma massa de dados interessante e bem diversificada para nosso estudo, afim de confirmarmos nossas massa de dados, vamos executar o *Bloco de Código 3* a seguir, para retornarmos em tela uma pequena porção de dados:  **— Bloco de Código 3 - Validando uma porção de dados da Tabela QueimadasCleanTable —**  Select Top 1000 DataHora,  Satelite,  Pais,  Estado,  Municipio  From QueimadasCleanTable  Go  Após a execução do *Bloco de Código 3*, você deverá ter obtido em tela, um resultado similar ao apresentado abaixo pelo *Figura 2*:  *Figura 2 - Dados coletados e apresentados em tela após a execução do Bloco de código 3.*  Por enquanto nenhuma novidade, nada em especial foi apresentado, não é mesmo? Eu acredito que sim. Agora que toda estrutura da tabela foi refeita, nossos dados foram validados, o que pode estar faltando para fazermos uso do comando DBCC CleanTable?  A resposta é simples, falta identificar quais são as colunas de comprimento variável existentes em nossa tabela que poderemos utilizar para entender o comportamento deste comando, sendo assim, nosso próximo passo será identificar quais seriam as colunas e seus respectivos tipos de dados que formam a estrutura da tabela QueimadasCleanTable, através da execução do *Bloco de Código 4* apresentado abaixo:  **— Bloco de Código 4 - Identificando as colunas de comprimento variável —**  Select [st.name](http://st.name/) As 'TableName',  [sc.name](http://sc.name/) As 'ColumnName',  sc.column\_id As 'ColumnID',  [sty.name](http://sty.name/) As 'DataType',  sc.max\_length As 'MaxLength'  from **sys.tables** st Inner Join **sys.columns** sc  on st.object\_id = sc.object\_id  Inner Join **sys.systypes** sty  on sc.system\_type\_id = sty.xtype  Where [st.name](http://st.name/) = 'QueimadasCleanTable'  And [sty.name](http://sty.name/) = 'VarChar'  Order By st.Name Asc, sc.column\_id Asc  Go  Note que estamos fazendo uso das conhecidas e tradicionais tabelas de sistema:   * **sys.tables;** * **sys.columns; e** * **sys.systypes.**   A execução do *Bloco de Código 4* é simples e rápida, a Figura 3 abaixo, deve ilustrar o resultado obtido após sua execução:  *Figura 3 - Relação de colunas que utilizam o tipo de dados Varchar() com tamanho variáveis.*  Estamos quase lá, já sabemos da existência **de 7(sete) colunas** que neste momento fazem parte da estrutura da nossa tabela QueimadasCleanTable que nos possibilitam serem utilizadas.  Como eu destaquei anteriormente o comando DBCC CleanTable tem como papel principal recuperar o espaço ocupado por estas colunas, quando as mesmas venham a ser removidas ou sofram alterações em seus tamanhos.  Para que possamos entender de forma clara e didática como o DBCC CleanTable trabalha, temos a necessidade de identificar os espaços ocupados neste momento por nossa tabela, e obrigatoriamente as taxas de alocação e fragmentação de dados, vamos então executar o *Bloco de Código 5*, o qual vai nos ajudar a identificar o espaçamento ocupado por nossa tabela:  **— Bloco de Código 5 - Identificando o espaço e áreas de alocação ocupadas pela Tabela QueimadasCleanTable —**  **-- Identificando os espaços ocupados --**  sp\_spaceused 'QueimadasCleanTable'  Go  **-- Identificando as taxas de alocação, fragmentação e distribuição de registros --**  Select object\_name(ddips.object\_id) As 'Tabela',  [si.name](http://si.name/) As 'Índice',  convert(decimal(5,2),isnull(ddips.avg\_fragmentation\_in\_percent,0)) As '% Média de Fragmentação',  convert(decimal(5,2),isnull(ddips.avg\_page\_space\_used\_in\_percent,0)) As '% Média de Espaço utilizado',  ddips.page\_count As 'Páginas',  ddips.compressed\_page\_count As 'Páginas compactadas',  ddips.record\_count As 'Registros',  ddips.ghost\_record\_count As 'Registros Fantasmas'  From sys.dm\_db\_index\_physical\_stats(db\_id(), object\_id('QueimadasCleanTable'),null, null, 'detailed') ddips Inner Join sys.indexes si  on si.object\_id = ddips.object\_id  Go  Acredito que você deve ter observado que o Bloco de Código 5 foi dividido em duas partes, a primeira fazendo uso da System Stored Procedure: **SP\_SpaceUsed**, e a segunda, através da DMF - Dynamic Management Function - **sys.dm\_db\_index\_physical\_stats**.  Como uma forma de ajudar a identificar e entender os dados coletados após a execução deste bloco de código, apresenta abaixo a Tabela 1 com os dados coletados através SP\_SpaceUsed e Tabela 2 com os dados coletados através **sys.dm\_db\_index\_physical\_stats**.   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Tabela 1 - SP\_SpaceUsed - Espaços Ocupados** | | | | | | | name | rows | reserved | data | index\_size | unused | | QueimadasCleanTable | 752252 | 81736 KB | 81272 KB | 312 KB | 152 KB |   Ao realizarmos uma breve análise, podemos observar através dos resultados apresentados na Tabela 1, que nosso tabela QueimadasCleanTable, neste momento esta ocupando uma área em disco de quase **82Mbs (Megabytes)**, sendo **81.2Mbs** para dados e **312Kbs (Kilobytes)** para índices, com uma área não alocada de 152Kbs.   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | |  | | | | | | | | | Tabela | Índice | % Média de Fragmentação | % Média de Espaço utilizado | Páginas | Páginas compactadas | Registros | Registros Fantasmas | | QueimadasCleanTable | PK\_CodigoQueimada\_Queimadas2018\_CleanTable | 0.01 | 99.34 | 10159 | 0 | 752252 | 0 | | QueimadasCleanTable | PK\_CodigoQueimada\_Queimadas2018\_CleanTable | 0.00 | 44.07 | 37 | 0 | 10159 | 0 | | QueimadasCleanTable | PK\_CodigoQueimada\_Queimadas2018\_CleanTable | 0.00 | 5.92 | 1 | 0 | 37 | 0 |   Já os dados apresentados pela Tabela 2, mostram uma pequena taxa de fragmentação de 0,01 % para nosso índice chave primária: **PK\_CodigoQueimada\_Queimadas2018\_CleanTable**, em sua área de alocação de dados, composta por **10.159** páginas de dados.  Estamos próximos da hora da verdade, com todo esta conjunto de dados coletados, poderemos comprovar como o DBCC CleanTable pode nos ajudar, através do Bloco de Codigo 6, realizaremos a exclusão de 4 (**Municipio, Bioma, AreaIndu e FRP)** das 7 colunas listadas anteriormente, logo na sequência vamos repetir a execução do Bloco de Código 5 para comprovar que as áreas e espaços ocupados continuam apresentando os mesmos valores, e nosso ultimo passo será executar o DBCC Clean Table.  **— Bloco de Código 6 - Removendo as colunas Municipio, Bioma, AreaIndu e FRP —**  Alter Table QueimadasCleanTable  Drop Column Municipio, Bioma, AreaIndu, FRP  Go  As colunas foram removidas corretamente, agora vamos repetidar a execução do *Bloco de Código 5* na sequência.  **-- Executar novamente o Bloco de Código 5 --**  **-- Identificando os espaços ocupados --**  sp\_spaceused 'QueimadasCleanTable'  Go  **-- Identificando as taxas de alocação, fragmentação e distribuição de registros --**  Select object\_name(ddips.object\_id) As 'Tabela',  [si.name](http://si.name/) As 'Índice',  convert(decimal(5,2),isnull(ddips.avg\_fragmentation\_in\_percent,0)) As '% Média de Fragmentação',  convert(decimal(5,2),isnull(ddips.avg\_page\_space\_used\_in\_percent,0)) As '% Média de Espaço utilizado',  ddips.page\_count As 'Páginas',  ddips.compressed\_page\_count As 'Páginas compactadas',  ddips.record\_count As 'Registros',  ddips.ghost\_record\_count As 'Registros Fantasmas'  From sys.dm\_db\_index\_physical\_stats(db\_id(), object\_id('QueimadasCleanTable'),null, null, 'detailed') ddips Inner Join sys.indexes si  on si.object\_id = ddips.object\_id  Go  A Figura 4, vai ilustrar e comprovar que os valores apresentados após a nova execução do Bloco de Código 5, são os mesmos obtidos em sua primeira execução:  *Figura 4 - Valores obtidos após a segunda execução do Bloco de Código 5.*  E agora chegou o grande momento, vamos executar o *Bloco de Código 7*, o qual terá a responsabilidade de executar o comando DBCC CleanTable, logo na sequência vamos executar novamente o *Bloco de Código 5*, e ai sim teremos uma surpresa:  **— Bloco de Código 7 - Executando o comando DBCC CleanTable —**  Dbcc CleanTable(TesteDBCCCleanTable,'dbo.QueimadasCleanTable')  Go  Por padrão como boa parte dos comandos DBCCs, o CleanTable, vai retornar na guia de mensagens a seguinte frase:  *"DBCC execution completed. If DBCC printed error messages, contact your system administrator."*  Agora repita novamente a execução do *Bloco de Código 5*, e observe que teremos um novo conjunto de valores apresentados.  **-- Executar novamente o Bloco de Código 5 --**  **-- Identificando os espaços ocupados --**  sp\_spaceused 'QueimadasCleanTable'  Go  **-- Identificando as taxas de alocação, fragmentação e distribuição de registros --**  Select object\_name(ddips.object\_id) As 'Tabela',  [si.name](http://si.name/) As 'Índice',  convert(decimal(5,2),isnull(ddips.avg\_fragmentation\_in\_percent,0)) As '% Média de Fragmentação',  convert(decimal(5,2),isnull(ddips.avg\_page\_space\_used\_in\_percent,0)) As '% Média de Espaço utilizado',  ddips.page\_count As 'Páginas',  ddips.compressed\_page\_count As 'Páginas compactadas',  ddips.record\_count As 'Registros',  ddips.ghost\_record\_count As 'Registros Fantasmas'  From sys.dm\_db\_index\_physical\_stats(db\_id(), object\_id('QueimadasCleanTable'),null, null, 'detailed') ddips Inner Join sys.indexes si  on si.object\_id = ddips.object\_id  Go  As tabelas 3 e 4 apresentadas na sequência, vamos nos ajudar a identificar estes novos valores apresentados após a execução do *Bloco de Código 7:*   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Tabela 3 - SP\_SpaceUsed - Espaços Ocupados** | | | | | | | name | rows | reserved | data | index\_size | unused | | QueimadasCleanTable | 752252 | 81736 KB | 81272 KB | 312 KB | 152 KB |   Ao analisarmos os valores apresentados na Tabela 3, inicialmente podemos ficar surpresos por não ocorreram mudanças, na verdade não vai ocorrer mesmo, pois como destacado no início deste post o DBCC CleanTable não tem a função de liberar o espaço físico e lógico ocupado pela tabela e suas estruturas.   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | Tabela 4 - Sys.dm\_db\_index\_physical\_stats - Taxas de Fragmentação, Distribuição de Páginas de Dados e Registros | | | | | | | | | Tabela | Índice | % Média de Fragmentação | % Média de Espaço utilizado | Páginas | Páginas compactadas | Registros | Registros Fantasmas | | QueimadasCleanTable | PK\_CodigoQueimada\_Queimadas2018\_CleanTable | 0.01 | 80.99 | 10159 | 0 | 752252 | 0 | | QueimadasCleanTable | PK\_CodigoQueimada\_Queimadas2018\_CleanTable | 0.00 | 44.07 | 37 | 0 | 10159 | 0 | | QueimadasCleanTable | PK\_CodigoQueimada\_Queimadas2018\_CleanTable | 0.00 | 5.92 | 1 | 0 | 37 | 0 |   Por outro lado, a Tabela 4 nos apresenta uma pequena mudança na coluna % Média de Espaço Utilizado que agora é de 80,99 % e antes era de 99,34%, ou seja, ao realizarmos a execução das colunas: **Municipio, Bioma, AreaIndu e FRP** o DBCC CleanTable realizou uma pequena recuperação de espaço que estava sendo ocupados por estas colunas em suas respectivas linhas de registro lógicos.  Isso não é algo fora do comum, conseguir reaproveitar as áreas que estavam sendo ocupadas anteriormente sem precisar realizar qualquer tipo de reconstrução ou mudanças de configuração. Eu acredito que sim.  Seguindo a tradição dos posts desta sessão, antes de encerrarmos, gostaria de contar com a sua participação neste post, respondendo a enquete abaixo:  [Veja a enquete](http://polldaddy.com/poll/10329980/)  **Quero propor um desafio**  Elabore um cenário similar ao apresentado aqui, e utilize os comandos Delete e Truncate Table em conjunto com o DBCC CleanTable, faça uma análise comparativa, tenho a certeza que este desafio vai lhe ajudar a entender de forma simples e objetiva as diferenças entre o Delete e o Truncate, sendo esta, uma das dúvidas mais recorrentes que podemos encontrar na internet.  Com isso chegamos ao final de mais um post da sessão Dica do Mês, espero que você tenha gostado, eu como de costume gostei muito. |