# DICA DO MÊS – COMO FICAR “QUASE LOUCO” UTILIZANDO ANÁLISE COMBINATÓRIA NO MICROSOFT SQL SERVER

Estamos no mês de março, carnaval já passou, o primeiro trimestre de 2019 está acabando, e para minha alegria te encontro mais uma vez no meu blog, caso esta seja a sua primeira visita ou acesso, fico mais feliz ainda, seja muito bem-vindo.

Este é mais um post da sessão **Dica do Mês**, sessão dedicada a compartilhar bimestralmente dicas, novidades, curiosidades e demais assuntos, conteúdos e informações relacionadas ao Microsoft SQL Server, Banco de Dados e Tecnologias de Banco de Dados.

No post de hoje, quero compartilhar com vocês a minha mais nova “loucura” criada em meus ambientes de estudos acadêmicos para ser utilizada no Microsoft SQL Server através do uso da Análise Combinatória, isso mesmo mais uma vez a matemática esta presente em nossas vidas e desta vez foi justamente para permitir a criação de um script que permite criar todas as sequência de combinações de letras e números afim de construir um gerador de placas de carros para todos os estados brasileiros.

Não parece realmente coisa de louco, minha esposa disse que sim, eu também acho (kkkkk).

Pois bem, ficou curioso para saber como eu criei mais esta “loucura”? Calma, daqui a pouco eu conto mais sobre isso para você.

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 – Como ficar “quase louco” utilizando análise combinatória no Microsoft SQL Server.**

### **INTRODUÇÃO**

Muito se fala que a área de banco de dados, e posteriormente os Sistemas Gerenciadores de Bancos de Dados (SGBDs) nasceram dos métodos, técnicas e também das teorias existentes na Matemática.

A cada dia eu tenho mais certeza que esta analogia é verdadeira, e neste post, eu pretendo justamente mostrar como mais uma vez esta fantástica e grandiosa área de estudos e conhecimentos pode nos ajudar a transformar algo que parece ter um nível de raciocínio tão complexo ou talvez impossível, em algo na verdade simples, fácil e de rápida compreensão.

Para ser mais direto, estou me referindo a análise combinatória, uma das mais variadas áreas de conhecimento e aprendizado existentes na Matemática e que este mero ser humano demorou um bom tempo para conseguir entender de verdade e aplicar de forma bem racional.

### **ANÁLISE COMBINATÓRIA**

Podemos determinar a **análise combinatória** como sendo um conjunto de possibilidades constituídos por elementos finitos, a mesma baseia-se em critérios que possibilitam a contagem. Realizamos o seu estudo na lógica matemática, analisando possibilidades e combinações.

Por exemplo: Descubra quantos números com 3 algarismos conseguimos formar com o conjunto numérico **{1, 2, 3},** olha a teoria de conjuntos aí gente….

* *Conjunto de elementos finito*: {1, 2, 3}.
* Conjunto de possibilidades de números com 3 algarismos: {123, 132, 213, 231, 312, 321}.

A análise combinatória estuda os seguintes conteúdos:

* Princípio fundamental da contagem;
* Fatorial;
* Permutação simples;
* Permutação com repetição;
* Arranjo simples; e
* Combinação simples.

Não vou abortar todos estes conteúdos de estudo utilizado pela análise combinatória neste post, mas sim o que mais entendo como importante e de extrema necessidade para o cenário que estaremos utilizando a posterior, sendo estes:

* **Permutação Simples; e**
* **Permutação com repetição.**

### **PERMUTAÇÃO SIMPLES**

Na permutação os elementos que compõem o agrupamento mudam de ordem, ou seja, de posição. Determinamos a quantidade possível de permutação dos elementos de um conjunto, com a seguinte expressão:

**Pn = n!**

**Pn = n . (n-1) . (n-2) . (n-3)…..1!**

*Exemplo: Em uma eleição para representante de sala de aula, 3 alunos candidataram-se:***Fernanda, Eduardo e Malú. Quais são os possíveis resultados dessa eleição?**

* **Fernanda (F);**
* **Eduardo (E); e**
* **Malú (M).**

Os possíveis resultados dessa eleição podem ser dados com uma permutação simples, acompanhe:

**n = 3 (Quantidade de candidatos concorrendo a representante)**

**Pn = n!**

**Pn = 3 . 2 . 1!**

**Pn = 6**

Para a eleição de representante, temos 6 possibilidades de resultado, em relação a posição dos candidatos, ou seja, 1º, 2º e 3º lugar.

Veja a seguir os possíveis resultados dessa eleição:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Resultado 1** | **Resultado 2** | **Resultado 3** | **Resultado 4** | **Resultado 5** | **Resultado 6** |
| **FE*M*** | **F*M*E** | **EF*M*** | **E*M*F** | ***M*EF** | ***M*FE** |

### **PERMUTAÇÃO COM REPETIÇÃO**

Nessa permutação alguns elementos que compõem o evento experimental são repetidos, quando isso ocorrer devemos aplicar a seguinte fórmula:

***Pn*(*n*1,*n*2,*n*3…*nk*)=*n*!*n*1!⋅*n*2!⋅*n*3!…*nk*!**

* ***Pn*(*n*1,*n*2,*n*3…*nk*) = permutação com repetição**
* ***n*! = total de elemetos do evento**
* ***n*1!⋅*n*2!⋅*n*3!…*nk*! = Elementos repetidos do evento**

*Exemplo: Quantos anagramas são possíveis formar com a palavra CASA. A palavra CASA possui:*

**4 letras (n) e duas vogais que se repetem (n1).**

* **n! = 4!**
* **n1! = 2!**

***Pn*(*n*1)=*n*!*n*1!**

***Pn*(*n*1)=4!2!**

***Pn*(*n*1)=4⋅3⋅2⋅1!2⋅1!**

***Pn*(*n*1)=242=12**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Anagramas da palavra CASA sem repetição | | | | | |
| **CASA** | **ACSA** | **ASCA** | **ASAC** | **SCAA** | **CSAA** |
| **AASC** | **AACS** | **CAAS** | **SAAC** | **SACA** | **ACAS** |

Bom, agora que conhecemos um pouco destes conceitos, você pode estar se perguntando:

**“O que o Microsoft SQL Server tem haver com isso?”.**

Então, tudo, pois ele faz justamente uso destes elementos e dos demais quando queremos realizar as combinações das mais variadas possíveis que envolvem letras, letras e números, ou somente números.

E aí, até aqui tudo tranquilo? Espero que sim, pois daqui em diante começaremos a preparar nosso ambiente e aplicaremos a análise combinatória e as permutações para colocarmos em funcionamento a minha “loucura”.

Para você ter a ideia do nível de loucura que estaremos trabalhando, ao realizar o uso das vinte e seis letras do alfabeto em nosso idioma da língua portuguesa teremos basicamente a seguinte permutação com repetição:

* **n! = 3!**
* **n1! = 26!**

Ou seja, de forma mais simples, vamos permutar: 26 letras \* 26 letras \* 26 letras, o que não apresentara um total de: ***17.756 (Dezessete Mil, Setecentos e Cinquenta e Seis) combinações de letras distintas.***

Mas não terminamos isso nossa caminhada, depois de realizar estas combinações de letras **(17.756)**, vamos fazer uso dos arranjos (**este conteúdo eu não abordei**), que nos permitirá criar em tempo real todos os agrupamentos entre letras e números (de 0 até 9999), estabelecendo a seguinte fórmula:

***Arranjos = LetrasCombinadas(17756) \* Numeracao(0…9999) = 177.542.244***

os anagramas que vimos apouco, como por exemplo: **AAA-0001,** o qual vai nos permitir obter um total de: **177.542.244 (Cento e Setenta e Sete Milhões, Quinhentos e Quarente e Dois Mil, Duzentas e Quarenta e Quatro) agrupamentos ou arranjos únicos e distintos que teremos a disposição para serem armazenados.**

Não é algo de louco mesmo?

### **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: **GeradorDePlacas**;
* Tabelas: **LetrasCombinadas**, **Numeracao**, **Placas** e **FaixasDePlacasPorEstado**;
* CTEs: **CTEMeuAlfabeto**; e
* Stored Procedure: **P\_PesquisarPlacas**.

## CRIANDO O AMBIENTE

Através do ***Bloco de Código 1*** apresentado abaixo, vamos realizar a criação dos respectivos elementos destacados anteriormente:

**— Bloco de Código 1 —**

**— Criando o Banco de Dados —**

Create Database GeradorDePlacas

Go

**— Acessando o Banco de Dados —**

Use GeradorDePlacas

Go

**— Desativando a contagem de linhas —**

Set NoCount On

Go

**— Criando a Tabela LetrasCombinadas para armazenar todas as combinações de Letras —**

Create Table LetrasCombinadas

(CodigoSequencialLetrasCombinadas SmallInt Primary Key Identity(1,1) Not Null,

SequencialDeLetrasCombinadas Char(3) Not Null)

Go

**— Criando a Tabela Numeracao para armazenar a faixa numérica de 1 até 9999 —**

Create Table Numeracao

(CodigoNumeracao SmallInt Primary Key Identity(1,1) Not Null)

Go

**— Criando a Tabela Placas para armazenar o CodigoSequencialLetrasCombinadas e o número da Placa —**

Create Table Placas

(CodigoSequencialPlacas Int Primary Key Identity(1,1) Not Null,

CodigoSequencialLetrasCombinadas SmallInt Not Null,

CodigoSequencialNumeroPlacas SmallInt Not Null)

Go

**— Criando a Tabela FaixasDeFaixasDePlacasPorEstado para armazenar as faixas de placas por Estado —**

Create Table FaixasDePlacasPorEstado

(CodigoSequencialFaixasDePlacasPoEstado TinyInt Primary Key Identity(1,1) Not Null,

CodigoSequencialFaixasDeLetrasNumerosInicial Char(7) Not Null,

CodigoSequencialFaixasDeLetrasNumerosFinal Char(7) Not Null,

FaixasDePlacasPorEstadoNomeDoEstado Varchar(30) Not Null)

Go

**— Inserindo a distribuição de Faixas de Placas Por Estado —**

Insert Into FaixasDePlacasPorEstado Values (‘AAA0001′,’BEZ9999′,’Paraná’)

Insert Into FaixasDePlacasPorEstado Values (‘BFA0001′,’GKI9999′,’São Paulo’)

Insert Into FaixasDePlacasPorEstado Values (‘GKJ0001′,’HOK9999′,’Minas Gerais’)

Insert Into FaixasDePlacasPorEstado Values (‘HOL0001′,’HQE9999′,’Maranhão’)

Insert Into FaixasDePlacasPorEstado Values (‘HQF0001′,’HTW9999′,’Mato Grosso do Sul’)

Insert Into FaixasDePlacasPorEstado Values (‘HTX0001′,’HZA9999′,’Ceará’)

Insert Into FaixasDePlacasPorEstado Values (‘HZB0001′,’IAP9999′,’Sergipe’)

Insert Into FaixasDePlacasPorEstado Values (‘IAQ0001′,’JDO9999′,’Rio Grande do Sul’)

Insert Into FaixasDePlacasPorEstado Values (‘JDP0001′,’JKR9999′,’Distrito Federal’)

Insert Into FaixasDePlacasPorEstado Values (‘JKS0001′,’JSZ9999′,’Bahia’)

Insert Into FaixasDePlacasPorEstado Values (‘JTA0001′,’JWE9999′,’Pará’)

Insert Into FaixasDePlacasPorEstado Values (‘JWF0001′,’JXY9999′,’Amazonas’)

Insert Into FaixasDePlacasPorEstado Values (‘JXZ0001′,’KAU9999′,’Mato Grosso’)

Insert Into FaixasDePlacasPorEstado Values (‘KAV0001′,’KFC9999′,’Goiás’)

Insert Into FaixasDePlacasPorEstado Values (‘KFD0001′,’KME9999′,’Pernambuco’)

Insert Into FaixasDePlacasPorEstado Values (‘KMF0001′,’LVE9999′,’Rio de Janeiro’)

Insert Into FaixasDePlacasPorEstado Values (‘LVF0001′,’LWQ9999′,’Piauí’)

Insert Into FaixasDePlacasPorEstado Values (‘LWR0001′,’MMM9999′,’Santa Catarina’)

Insert Into FaixasDePlacasPorEstado Values (‘MMN0001′,’MOW9999′,’Paraíba’)

Insert Into FaixasDePlacasPorEstado Values (‘MOX0001′,’MTZ9999′,’Espírito Santo’)

Insert Into FaixasDePlacasPorEstado Values (‘MUA0001′,’MVK9999′,’Alagoas’)

Insert Into FaixasDePlacasPorEstado Values (‘MVL0001′,’MXG9999′,’Tocantins’)

Insert Into FaixasDePlacasPorEstado Values (‘MXH0001′,’MZM9999′,’Rio Grande do Norte’)

Insert Into FaixasDePlacasPorEstado Values (‘MZN0001′,’NAG9999′,’Acre’)

Insert Into FaixasDePlacasPorEstado Values (‘NAH0001′,’NBA9999′,’Roraima’)

Insert Into FaixasDePlacasPorEstado Values (‘NBB0001′,’NEH9999′,’Rondônia’)

Insert Into FaixasDePlacasPorEstado Values (‘NEI0001′,’NFB9999′,’Amapá’)

Insert Into FaixasDePlacasPorEstado Values (‘NFC0001′,’NGZ9999′,’Goiás’)

Insert Into FaixasDePlacasPorEstado Values (‘NHA0001′,’NHT9999′,’Maranhão’)

Insert Into FaixasDePlacasPorEstado Values (‘NHU0001′,’NIX9999′,’Piauí’)

Insert Into FaixasDePlacasPorEstado Values (‘NIY0001′,’NJW9999′,’Mato Grosso’)

Insert Into FaixasDePlacasPorEstado Values (‘NJX0001′,’NLU9999′,’Goiás’)

Insert Into FaixasDePlacasPorEstado Values (‘NLV0001′,’NMO9999′,’Alagoas’)

Insert Into FaixasDePlacasPorEstado Values (‘NMP0001′,’NNI9999′,’Maranhão’)

Insert Into FaixasDePlacasPorEstado Values (‘NNJ0001′,’NOH9999′,’Rio Grande do Norte’)

Insert Into FaixasDePlacasPorEstado Values (‘NOI0001′,’NPB9999′,’Amazonas’)

Insert Into FaixasDePlacasPorEstado Values (‘NPC0001′,’NPQ9999′,’Mato Grosso’)

Insert Into FaixasDePlacasPorEstado Values (‘NPR0001′,’NQK9999′,’Paraíba’)

Insert Into FaixasDePlacasPorEstado Values (‘NQL0001′,’NRE9999′,’Ceará’)

Insert Into FaixasDePlacasPorEstado Values (‘NRF0001′,’NSD9999′,’Mato Grosso do Sul’)

Insert Into FaixasDePlacasPorEstado Values (‘NSE0001′,’NTC9999′,’Pará’)

Insert Into FaixasDePlacasPorEstado Values (‘NTD0001′,’NTW9999′,’Bahia’)

Insert Into FaixasDePlacasPorEstado Values (‘NTX0001′,’NUG9999′,’Mato Grosso’)

Insert Into FaixasDePlacasPorEstado Values (‘NUH0001′,’NUL9999′,’Roraima’)

Insert Into FaixasDePlacasPorEstado Values (‘NUM0001′,’NVF9999′,’Ceará’)

Insert Into FaixasDePlacasPorEstado Values (‘NVG0001′,’NVN9999′,’Sergipe’)

Insert Into FaixasDePlacasPorEstado Values (‘NVO0001′,’NWR9999′,’Goiás’)

Insert Into FaixasDePlacasPorEstado Values (‘NWS0001′,’NXQ9999′,’Maranhão’)

Insert Into FaixasDePlacasPorEstado Values (‘NXR0001′,’NXT9999′,’Acre’)

Insert Into FaixasDePlacasPorEstado Values (‘NXU0001′,’NXW9999′,’Pernambuco’)

Insert Into FaixasDePlacasPorEstado Values (‘NXX0001′,’NYG9999′,’Minas Gerais’)

Insert Into FaixasDePlacasPorEstado Values (‘NYH0001′,’NZZ9999′,’Bahia’)

Insert Into FaixasDePlacasPorEstado Values (‘OAA0001′,’OAO9999′,’Amazonas’)

Insert Into FaixasDePlacasPorEstado Values (‘OAP0001′,’OBS9999′,’Mato Grosso’)

Insert Into FaixasDePlacasPorEstado Values (‘OBT0001′,’OCA9999′,’Pará’)

Insert Into FaixasDePlacasPorEstado Values (‘OCB0001′,’OCU9999′,’Ceará’)

Insert Into FaixasDePlacasPorEstado Values (‘OCV0001′,’ODT9999′,’Espírito Santo’)

Insert Into FaixasDePlacasPorEstado Values (‘ODU0001′,’OEI9999′,’Piauí’)

Insert Into FaixasDePlacasPorEstado Values (‘OEJ0001′,’OES9999′,’Sergipe’)

Insert Into FaixasDePlacasPorEstado Values (‘OET0001′,’OFH9999′,’Paraíba’)

Insert Into FaixasDePlacasPorEstado Values (‘OFI0001′,’OFW9999′,’Pará’)

Insert Into FaixasDePlacasPorEstado Values (‘OFX0001′,’OGG9999′,’Paraíba’)

Insert Into FaixasDePlacasPorEstado Values (‘OGH0001′,’OHA9999′,’Goiás’)

Insert Into FaixasDePlacasPorEstado Values (‘OHB0001′,’OHK9999′,’Alagoas’)

Insert Into FaixasDePlacasPorEstado Values (‘OHL0001′,’OHW9999′,’Rondônia’)

Insert Into FaixasDePlacasPorEstado Values (‘OHX0001′,’OIQ9999′,’Ceará’)

Insert Into FaixasDePlacasPorEstado Values (‘OIR0001′,’OJK9999′,’Maranhão’)

Insert Into FaixasDePlacasPorEstado Values (‘OJR0001′,’OKC9999′,’Rio Grande do Norte’)

Insert Into FaixasDePlacasPorEstado Values (‘OKI0001′,’OLG9999′,’Bahia’)

Insert Into FaixasDePlacasPorEstado Values (‘OLH0001′,’OLN9999′,’Tocantins’)

Insert Into FaixasDePlacasPorEstado Values (‘OLO0001′,’OMH9999′,’Minas Gerais’)

Insert Into FaixasDePlacasPorEstado Values (‘OMI0001′,’OOF9999′,’Goiás’)

Insert Into FaixasDePlacasPorEstado Values (‘OOG0001′,’OOU9999′,’Mato Grosso do Sul’)

Insert Into FaixasDePlacasPorEstado Values (‘OOV0001′,’ORC9999′,’Minas Gerais’)

Insert Into FaixasDePlacasPorEstado Values (‘ORD0001′,’ORM9999′,’Alagoas’)

Insert Into FaixasDePlacasPorEstado Values (‘ORN0001′,’OSV9999′,’Ceará’)

Insert Into FaixasDePlacasPorEstado Values (‘OSW0001′,’OTZ9999′,’Pará’)

Insert Into FaixasDePlacasPorEstado Values (‘OUA0001′,’OUE9999′,’Piauí’)

Insert Into FaixasDePlacasPorEstado Values (‘OUF0001′,’OVD9999′,’Bahia’)

Insert Into FaixasDePlacasPorEstado Values (‘OVE0001′,’OWC9999′,’Espírito Santo’)

Insert Into FaixasDePlacasPorEstado Values (‘OWD0001′,’OYG9999′,’Santa Catarina’)

Insert Into FaixasDePlacasPorEstado Values (‘PEE0001′,’PFQ9999′,’Pernambuco’)

Insert Into FaixasDePlacasPorEstado Values (‘PFR0001′,’PGK9999′,’Pernambuco’)

Insert Into FaixasDePlacasPorEstado Values (‘PGL0001′,’PHE9999′,’Pernambuco’)

Insert Into FaixasDePlacasPorEstado Values (‘SAV0001′,’SAV9999′,’São Paulo’)

Go

**— Validando os dados inseridos na Tabela FaixasDePlacasPorEstado —**

Select \* From FaixasDePlacasPorEstado

Go

**— Inserindo 9999 linhas de registros lógicos na Tabela Numeracao —**

Insert Into Numeracao Default Values

Go 9999

**— Validando os dados inseridos na Tabela Numeracao —**

Select \* From Numeracao

Go

Muito bem, agora que temos nossa estrutura quase toda montada, vamos avançar mais um pouco e fazer uso da análise combinatória, conforme o ***Bloco de Código 2*** declarado abaixo:

**— Bloco de Código 2 —**

**— Montando a CTE Recursiva para Gerar todas as combinações de Letras —**

;With CTEMeuAlfabeto

As

(Select \* From (Values (‘A’),(‘B’),(‘C’),(‘D’),(‘E’),(‘F’),(‘G’),(‘H’),(‘I’),(‘J’),(‘K’),

(‘L’),(‘M’),(‘N’),(‘O’),(‘P’),(‘Q’),(‘R’),(‘S’),(‘T’),(‘U’),(‘V’),

(‘W’),(‘X’),(‘Y’),(‘Z’)) As Alfabeto (LetrasDoAlfabeto)

)

**— Inserindo as combinações de letras na Tabela LetrasCombinadas —**

Insert Into LetrasCombinadas

Select Distinct Concat(A1.LetrasDoAlfabeto, A2.LetrasDoAlfabeto, A3.LetrasDoAlfabeto) As ConcatenacaoLetrasCombinadas

From **CTEMeuAlfabeto A1**

**Cross Join CTEMeuAlfabeto A2 — Aqui que a mágia acontece —**

**Cross Join CTEMeuAlfabeto A3** **— O cross join vai cruzar e combinar todas as letras —**

Order By ConcatenacaoLetrasCombinadas Asc

Go

**— Validando os dados inseridos na Tabela LetrasCombinadas —**

Select \* From LetrasCombinadas

Go

Nosso ***Bloco de Código 2*** já deve ter sido processado, pois ele é especificamente o centro das atenções para conseguirmos criar todas as combinações possíveis de letras do nosso alfabeto, que estará criando em poucos segundos um total ***de: 17.756 (Dezessete Mil, Setecentos e Cinquenta e Seis) combinações de letras distintas.***

Ufa, estamos avançando, você vai poder notar neste post, que estou fazendo uso de diversos comandos e técnicas existentes no SQL Server desde as primeiras versões como também outros implementados nas versões mais atuais, dentre eles destaco o uso de Tabela Derivada conforme o comando **Select From (Values()) existente desde a versão 2000, e também da CTE – Common Table Expression adicionado ao Microsoft SQL Server a partir da versão 2005.**

Vamos avançar mais ainda, pois o objetivo deste post não é mostrar somente o uso da análise combinatória, ao contrário, o estudo aqui criado, me permitiu elaborar alguns cenários que me permitiram adotar formas e técnicas diferentes de realizar todas as combinações entre letras e números afim de processar, criar e armazenar todas as placas criadas justamente na tabela denominada placas.

Destaco que foram criados 3 (três) cenários de estudo de acordo com percepções e análises que realizei, fazendo uso de recursos e técnicas distintas visando identificar o que poderia ser melhor utilizada:

* **Cenário 1** – Inserindo dados na Tabela de Placas através de Loop Condicional;
* **Cenário 2** – Inserindo dados na Tabela de Placas através de CTE Recursiva com Junção Cruzada; e
* **Cenário 3** – Inserindo dados na Tabela de Placas através de Junção Cruzada.

**Nota:** Tenho a certeza que você vai poder criar outros cenários e novas análises, como também, utilizar os mais variados recursos e funcionalidades existentes no Microsoft SQL Server, reforço mais uma vez que estes cenários são meras amostras de estudo e comparações do meu entendimento.

O objetivo de ter criado estes cenários, possibilitou realizarmos comparações de tempo de processamento no que relaciona ao uso da CPU e consumo de memória RAM para cada um dos cenários.

A *Tabela 1* declarada abaixo apresenta um resumo dos valores obtidos **durante 5 rodadas de processamento** executadas em meu ambiente:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Cenário** | **Média de Uso de CPU** | **Média do Uso de Memória em GBs** | **Média de Uso de Disco Rígido** | **Tempo Mínimo de processamento** | **Tempo Máximo de processamento** |
| **01** | **57%** | **3.5 Gbs** | **65%** | **19 Hrs e 23 segundos** | **21Hrs e 18 segundos** |
| 02 | 34% | 2.8 Gbs | 42% | 33 minutos e 6 segundos | 42 minutos e 27 segundos |
| ***03*** | ***18%*** | ***2.0 Gbs*** | ***24%*** | ***6 minutos e 13 segundos*** | ***10 minutos e 25 segundos*** |

*Tabela 1 – Resumo comparativo do uso de CPU, Memória e Disco, em conjunto com os tempos de processamento demandados para cada cenário.*

Show, agora que temos este pequeno resumo dos tempos de processamento e uso dos principais recursos de hardware, já podemos conhecer cada um dos cenários elaborados abaixo, conforme apresenta o ***Bloco de Código 3*** *a seguir****:***

**— Bloco de Código 3 —**

**— Cenário 1 –**

**— Inserindo dados na Tabela de Placas através de Loop Condicional – 19Hrs e 33s de processamento —**

**— Limpando a Tabela de Placas —**

Truncate Table Placas

Go

**— Declarando as variáveis de controle —**

Declare @ContadorSequencialTotalPlacasInseridas Int = 1,

@ContadorSequencialParcialPlacasInseridas Int = 1,

@CodigoSequencialLetrasCombinadas Int = 1

**— Abrindo o Loop de Inserção —**

While @ContadorSequencialTotalPlacasInseridas <=(Select Max(CodigoSequencialLetrasCombinadas) From LetrasCombinadas)

Begin

Set @CodigoSequencialLetrasCombinadas=(Select CodigoSequencialLetrasCombinadas From LetrasCombinadas

Where CodigoSequencialLetrasCombinadas = @ContadorSequencialTotalPlacasInseridas)

While @ContadorSequencialParcialPlacasInseridas <=9999 — Contador a cada 9999 incrementa a variável @ContadorSequencialTotalPlacasInseridas

Begin

Insert Into Placas (CodigoSequencialLetrasCombinadas, CodigoSequencialNumeroPlacas)

Values (@CodigoSequencialLetrasCombinadas, @ContadorSequencialParcialPlacasInseridas)

Set @ContadorSequencialParcialPlacasInseridas +=1

End

Set @ContadorSequencialParcialPlacasInseridas = 1

Set @CodigoSequencialLetrasCombinadas +=1

Set @ContadorSequencialTotalPlacasInseridas +=1

End

**— Cenário 2 –**

**— Inserindo dados na Tabela de Placas através de CTE Recursiva com Junção Cruzada — 33 minutos e 40s de processamento —**

**— Limpando a Tabela de Placas —**

Truncate Table Placas

Go

**— Declarando as variáveis de controle —**

Declare @ContadorSequencialTotalPlacasInseridas Int = 1,

@CodigoSequencialLetrasCombinadas SmallInt = 1

While @ContadorSequencialTotalPlacasInseridas <=(Select Max(CodigoSequencialLetrasCombinadas) From LetrasCombinadas)

Begin

**— Realizando a Junção Cruzada entre as Tabelas LetrasCombinadas x CTENumeracao —**

;With CTENumeracao

As

(Select 1 As Numero

Union All

Select Numero + 1 From CTENumeracao

Where Numero <=9998

)

Insert Into Placas (CodigoSequencialLetrasCombinadas, CodigoSequencialNumeroPlacas)

Select LC.CodigoSequencialLetrasCombinadas, N.Numero

From LetrasCombinadas LC Cross Join **— Aqui acontece a mágia**

CTENumeracao N **— Cross Joi vai combinar todas as Letras com 9999 números —**

Where CodigoSequencialLetrasCombinadas = @CodigoSequencialLetrasCombinadas

Option (MaxRecursion 0)

Set @CodigoSequencialLetrasCombinadas +=1

Set @ContadorSequencialTotalPlacasInseridas +=1

End

Go

**— Cenário 3 –**

**— Inserindo dados na Tabela de Placas através de Junção Cruzada – 10 minutos e 27s de processamento —**

**— Limpando a Tabela de Placas —**

Truncate Table Placas

Go

**— Declarando as variáveis de controle —**

Declare @ContadorSequencialTotalPlacasInseridas Int = 1,

@CodigoSequencialLetrasCombinadas SmallInt = 1

While @ContadorSequencialTotalPlacasInseridas <=(Select Max(CodigoSequencialLetrasCombinadas) From LetrasCombinadas)

Begin

**— Realizando a Junção Cruzada entre as Tabelas LetrasCombinadas x Numeracao —**

Insert Into Placas (CodigoSequencialLetrasCombinadas, CodigoSequencialNumeroPlacas)

Select LC.CodigoSequencialLetrasCombinadas, N.CodigoNumeracao

From LetrasCombinadas LC Cross Join **— Aqui acontece a mágia**

Numeracao N **— Cross Joi vai combinar todas as Letras com 9999 números —**

Where CodigoSequencialLetrasCombinadas = @CodigoSequencialLetrasCombinadas

Set @CodigoSequencialLetrasCombinadas +=1

Set @ContadorSequencialTotalPlacasInseridas +=1

End

Go

Show, show, e show, nossos três cenários de estudo e testes já estão apresentados, basta você escolher qual deseja brincar, executar, encontrar as melhorias e possíveis falhas que podem existir.

Pra finalizar nossa longa caminhada, apresento o ***Bloco de Código 4***, o qual vai ilustrar como podemos consultar nossas tabelas e obter os dados já inseridos após o processamento de um dos cenários, e por fim o ***Bloco de Código 5*** que apresenta o código utilizado no Bloco de Código 4 transformado em uma Stored Procedure denominada: ***P\_PesquisarPlacas.***

**— Bloco de Código 4 —**

**— Apresentando as 100 primeiras Placas Geradas —**

Select Top 100 Concat(LC.SequencialDeLetrasCombinadas,’-‘,

**Convert(Char(4),Case**

**When P.CodigoSequencialNumeroPlacas BetWeen 1 And 9 Then Concat(‘000’,P.CodigoSequencialNumeroPlacas)**

**When P.CodigoSequencialNumeroPlacas BetWeen 10 And 99 Then Concat(’00’,P.CodigoSequencialNumeroPlacas)**

**When P.CodigoSequencialNumeroPlacas BetWeen 100 And 999 Then Concat(‘0’,P.CodigoSequencialNumeroPlacas)**

**When P.CodigoSequencialNumeroPlacas BetWeen 1000 And 9999 Then Convert(Char(4),P.CodigoSequencialNumeroPlacas)**

**End)) As ‘Placa’,**

IsNull(F.FaixasDePlacasPorEstadoNomeDoEstado,’Sequência não atribuída…’) As ‘Nome do Estado’

From LetrasCombinadas LC Inner Join Placas P

On LC.CodigoSequencialLetrasCombinadas = P.CodigoSequencialLetrasCombinadas

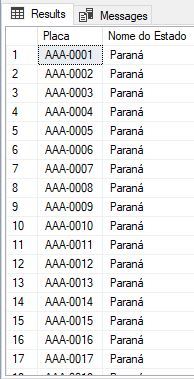
Left Join FaixasDePlacasPorEstado F

On LC.SequencialDeLetrasCombinadas Between SubString(F.CodigoSequencialFaixasDeLetrasNumerosInicial,1,3)

And SubString(F.CodigoSequencialFaixasDeLetrasNumerosFinal,1,3)

Go

A *Figura 1* apresentada abaixo, ilustra o possível resultado obtido após a execução do Bloco de Código 4:

***Figura 1 – Relação das 100 primeiras placas de carros criadas e inseridas na tabela Placas.***

**— Bloco de Código 5 —**

**— Criando uma Stored Procedure para pesquisa de placas —**

Create or Alter Procedure P\_PesquisarPlacas @LetrasCombinadas Char(3), @SequenciaNumerica SmallInt = Null

As

Begin

Set NoCount On

Select Concat(LC.SequencialDeLetrasCombinadas,’-‘,

Convert(Char(4),Case

When P.CodigoSequencialNumeroPlacas BetWeen 1 And 9 Then Concat(‘000′,P.CodigoSequencialNumeroPlacas)

When P.CodigoSequencialNumeroPlacas BetWeen 10 And 99 Then Concat(’00’,P.CodigoSequencialNumeroPlacas)

When P.CodigoSequencialNumeroPlacas BetWeen 100 And 999 Then Concat(‘0’,P.CodigoSequencialNumeroPlacas)

When P.CodigoSequencialNumeroPlacas BetWeen 1000 And 9999 Then Convert(Char(4),P.CodigoSequencialNumeroPlacas)

End)) As ‘Placa’,

IsNull(F.FaixasDePlacasPorEstadoNomeDoEstado,’Sequência não atribuída…’) As ‘Nome do Estado’

From LetrasCombinadas LC Inner Join Placas P

On LC.CodigoSequencialLetrasCombinadas = P.CodigoSequencialLetrasCombinadas

Left Join FaixasDePlacasPorEstado F

On LC.SequencialDeLetrasCombinadas Between SubString(F.CodigoSequencialFaixasDeLetrasNumerosInicial,1,3)

And SubString(F.CodigoSequencialFaixasDeLetrasNumerosFinal,1,3)

Where LC.SequencialDeLetrasCombinadas = @LetrasCombinadas

And P.CodigoSequencialNumeroPlacas = @SequenciaNumerica

End

Após a Stored Procedure estar criada, basta realizar sua execução conforme o exemplo apresentado abaixo, passando a sequência de letras e números que você deseja consultar.

**Importante:** Destaco que algumas combinações de placas de carro ainda não estão sendo utilizadas em nosso território, dentre elas as que começam com as letras: **W, X, Y e Z.**

**— Executando a Stored Procedure P\_PesquisarPlacas —**

Exec P\_PesquisarPlacas ‘FBD’,3127

Go

Sensacional, chegamos ao final, missão cumprida e entregue, acredito que este foi um dos estudos mais prazerosos e de grande obtenção de conhecimento que eu realizei nos últimos meses.