Administração de banco de dados

Aula 10 - Tópicos adicionais em Administração

Gustavo Maia

Gustavo.Maia@FaculdadeImpacta.com.br

Agenda

- **Tópicos adicionais em administração**
  - **System Versioned Tables**
    ou bancos de dados temporais
  - **InMemory Tables**
  - **Change Data Capture**
  - **CLR / Extended Stored Procedures**
  - **Column Store Indexes**

System Versioned Tables

“System Versioned Tables” ou de versão do sistema foram introduzidas como um recurso de banco de dados no SQL Server 2016. Nos permite criar um tipo de tabela que nos fornece informações sobre os dados que foram armazenados em qualquer horário especificado, em vez de apenas os dados que são atuais.

Limitações:

- Não é possível utilizar os comandos DDL para exclusão, adição ou exclusão de colunas.
- Não é possível utilizar FILETABLE e FILESTREAM.
- Excluir os dados através do comando TRUNCATE TABLE não é permitido.
- Linked Server não é suportado nas consultas aos dados de Temporal Tables.

Vantagens:

- Ela permite que o SQL Server mantenha e gerencie o histórico dos dados na tabela automaticamente.
- Auditoria reconstruindo os dados em caso de alterações inadvertidas projetando e relatando para análise de tendência histórica protegendo os dados em caso de perda acidental de dados

Mais detalhes: Tabelas temporais - SQL Server | Microsoft Docs
Criando uma tabela temporal com controle da versão do sistema - SQL Server | Microsoft Docs

System Versioned Tables

Demonstração:

```
--Original
CREATE TABLE consulta
  (
     id          INT NOT NULL IDENTITY(1, 1),
     id_paciente INT NOT NULL,
     id_medico   INT NOT NULL,
     numero_sala INT NOT NULL,
     datahora    DATETIME NOT NULL,
     duracao     TINYINT NOT NULL,
     CONSTRAINT pk_consulta PRIMARY KEY ( id ),
     CONSTRAINT fk_consultamedico FOREIGN KEY ( id_medico )
         REFERENCES medico(id ),
     CONSTRAINT fk_consultapaciente FOREIGN KEY ( id_paciente )
         REFERENCES     paciente( id ),
     CONSTRAINT fk_consultasala FOREIGN KEY ( numero_sala )
         REFERENCES sala(numero )
  )
go
```

System Versioned Tables

```
--Versionada
CREATE TABLE consulta_versionada
  (  id           INT NOT NULL IDENTITY(1, 1),
     id_paciente  INT NOT NULL,
     id_medico    INT NOT NULL,
     numero_sala  INT NOT NULL,
     datahora     DATETIME NOT NULL,
     duracao      TINYINT NOT NULL,
     ValidFrom    DATETIME2 GENERATED ALWAYS AS ROW START,
     ValidTo      DATETIME2 GENERATED ALWAYS AS ROW END,
     PERIOD FOR SYSTEM_TIME (ValidFrom, ValidTo),
     CONSTRAINT pk_consulta_versionada PRIMARY KEY ( id ),
     CONSTRAINT fk_consultamedico_versionada
              FOREIGN KEY ( id_medico ) REFERENCES medico( id ),
     CONSTRAINT fk_consultapaciente_versionada
              FOREIGN KEY ( id_paciente ) REFERENCES paciente( id ),
     CONSTRAINT fk_consultasala_versionada
              FOREIGN KEY ( numero_sala ) REFERENCES sala( numero )
  )
  WITH (SYSTEM_VERSIONING = ON (HISTORY_TABLE = dbo.Consulta_historica))
go
```

System Versioned Tables

```
--Tabelas vazias ( recém criadas )
SELECT * FROM   consulta_versionada
SELECT * FROM   consulta_historica
```

Resultados | Mensagens

ID	ID_Paciente	ID_Medico	Numero_Sala	DataHora	Duracao	ValidFrom	ValidTo

ID	ID_Paciente	ID_Medico	Numero_Sala	DataHora	Duracao	ValidFrom	ValidTo

```
--Inserção de alguns valores
INSERT INTO consulta_versionada(id_paciente,id_medico,numero_sala,datahora,duracao)
SELECT ....
--Alterações do número da sala
UPDATE consulta_versionada SET numero_sala = 484 WHERE  id = 1
```

Resultados | Mensagens

	ID	ID_Paciente	ID_Medico	Numero_Sala	DataHora	Duracao	ValidFrom	ValidTo
1	1	66	6	484	2020-03-04 08:00:00.000	15	2021-05-13 14:44:12.0357326	9999-12-31 23:59:59.9999999
2	2	285	7	652	2020-06-03 13:45:00.000	30	2021-05-13 14:43:58.9170706	9999-12-31 23:59:59.9999999
3	3	273	4	171	2018-10-14 14:15:00.000	30	2021-05-13 14:43:59.9302266	9999-12-31 23:59:59.9999999
4	4	257	2	907	2019-03-21 09:00:00.000	15	2021-05-13 14:44:00.9428846	9999-12-31 23:59:59.9999999
5	5	283	7	484	2020-07-31 15:45:00.000	15	2021-05-13 14:44:01.9502159	9999-12-31 23:59:59.9999999
6	6	319	6	543	2020-02-01 10:00:00.000	15	2021-05-13 14:44:02.9612864	9999-12-31 23:59:59.9999999
7	7	268	4	652	2019-04-01 14:30:00.000	30	2021-05-13 14:44:03.9713600	9999-12-31 23:59:59.9999999
8	8	421	4	759	2020-10-07 11:45:00.000	30	2021-05-13 14:44:04.9802142	9999-12-31 23:59:59.9999999
9	9	416	1	239	2021-03-28 08:15:00.000	30	2021-05-13 14:44:05.9851366	9999-12-31 23:59:59.9999999
10	10	278	3	907	2018-09-11 13:45:00.000	15	2021-05-13 14:44:06.9993738	9999-12-31 23:59:59.9999999

	ID	ID_Paciente	ID_Medico	Numero_Sala	DataHora	Duracao	ValidFrom	ValidTo
1	1	66	6	652	2020-03-04 08:00:00.000	15	2021-05-13 14:43:57.9067698	2021-05-13 14:44:12.0357326

System Versioned Tables

```
SELECT *
FROM   consulta_versionada
FOR system_time AS OF '2021-05-13 14:43:57.9067698'
```

	ID	ID_Paciente	ID_Medico	Numero_Sala	DataHora	Duracao	ValidFrom	ValidTo
1	1	66	6	652	2020-03-04 08:00:00.000	15	2021-05-13 14:43:57.9067698	2021-05-13 14:44:12.0357326

```
SELECT *
FROM consulta_versionada
FOR system_time BETWEEN '2021-05-13 14:43:57.9067698' AND '2021-05-13 14:44:12.0357326'
```

Resultados | Mensagens

	ID	ID_Paciente	ID_Medico	Numero_Sala	DataHora	Duracao	ValidFrom	ValidTo
1	1	66	6	484	2020-03-04 08:00:00.000	15	2021-05-13 14:44:12.0357326	9999-12-31 23:59:59.9999999
2	2	285	7	652	2020-06-03 13:45:00.000	30	2021-05-13 14:43:58.9170706	9999-12-31 23:59:59.9999999
3	3	273	4	171	2018-10-14 14:15:00.000	30	2021-05-13 14:43:59.9302266	9999-12-31 23:59:59.9999999
4	4	257	2	907	2019-03-21 09:00:00.000	15	2021-05-13 14:44:00.9428846	9999-12-31 23:59:59.9999999
5	5	283	7	484	2020-07-31 15:45:00.000	15	2021-05-13 14:44:01.9502159	9999-12-31 23:59:59.9999999
6	6	319	6	543	2020-02-01 10:00:00.000	15	2021-05-13 14:44:02.9612864	9999-12-31 23:59:59.9999999
7	7	268	4	652	2019-04-01 14:30:00.000	30	2021-05-13 14:44:03.9713600	9999-12-31 23:59:59.9999999
8	8	421	4	759	2020-10-07 11:45:00.000	30	2021-05-13 14:44:04.9802142	9999-12-31 23:59:59.9999999
9	9	416	1	239	2021-03-28 08:15:00.000	30	2021-05-13 14:44:05.9851366	9999-12-31 23:59:59.9999999
10	10	278	3	907	2018-09-11 13:45:00.000	15	2021-05-13 14:44:06.9993738	9999-12-31 23:59:59.9999999
11	1	66	6	652	2020-03-04 08:00:00.000	15	2021-05-13 14:43:57.9067698	2021-05-13 14:44:12.0357326

System Versioned Tables

Desfazendo:

```
ALTER TABLE consulta_versionada SET (system_versioning = OFF )
DROP TABLE consulta_versionada
DROP TABLE consulta_historica
```

InMemory Tables

OLTP na memória pode melhorar significativamente o desempenho de processamento de transações, inclusão de dados e carregamento de dados, e cenários de dados temporário.

Vantagens:

- Aumento considerável de desempenho do banco para processamento de:
  - Transações.
  - Ingestão de dados.
  - Cenários temporários (Transientes).

Desvantagens:

- Possibilidade de perdas potenciais de dados.(*)
- Limite no tamanho do banco de dados.
- Sem suporte para indices clusterizados
- Sem suporte para foreign key

Mais detalhes:

https://docs.microsoft.com/pt-br/sql/relational-databases/in-memory-oltp/in-memory-oltp-in-memory-optimization?view=sql-server-ver15

https://www.youtube.com/watch?v=I5l5eophmK4

InMemory Tables

Preparação:

```
USE master
go

ALTER DATABASE consultorio ADD filegroup fg_inmemory
     CONTAINS memory_optimized_data;
go

-- sp_helpdb consultorio
ALTER DATABASE consultorio ADD FILE ( NAME = 'consultorio_inMemory', filename =
'C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\MSSQL\DATA\consultorio.ndf'
) TO filegroup fg_inmemory
go
```

InMemory Tables

Demonstração:

```
CREATE TABLE consulta_inmemory
  (
     id          INT NOT NULL IDENTITY(1, 1),
     id_paciente INT NOT NULL,
     id_medico   INT NOT NULL,
     numero_sala INT NOT NULL,
     datahora    DATETIME NOT NULL,
     duracao     TINYINT NOT NULL,
     CONSTRAINT pk_consulta_inmemory PRIMARY KEY NONCLUSTERED ( id )
  )
WITH (MEMORY_OPTIMIZED = ON, DURABILITY = SCHEMA_AND_DATA)
go
```

InMemory Tables

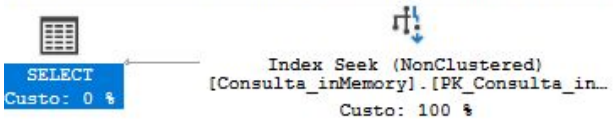
Comparação por uma consulta pela PK :

```
148  select * from Consulta_inMemory where id = 1234
149  select * from Consulta where id = 1234
150
151
```

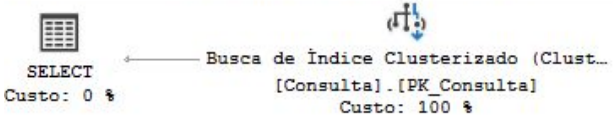
53 %

Mensagens | Plano de execução

```
Consulta 1: Custo da consulta (relativo ao lote): 12%
select * from Consulta_inMemory where id = 1234
```




```
Consulta 2: Custo da consulta (relativo ao lote): 88%
select * from Consulta where id = 1234
```



Quais suas conclusões ?

InMemory Tables

Indices:

```
-- Indices InMemory
ALTER TABLE consulta_inmemory
     ADD CONSTRAINT uq_consulta_inmemory_datahora
     UNIQUE NONCLUSTERED (datahora, id); --Unique
     go
ALTER TABLE consulta_inmemory
     ADD index IDX_HASH_CONSULTA_INMEMORY_PACIENTE
     hash (id_paciente) WITH (bucket_count = 64); -- Nonunique.
go
--Criando os “equivalentes” na versão normal
ALTER TABLE consulta
     ADD CONSTRAINT uq_consulta_datahora UNIQUE NONCLUSTERED (datahora, id);--Unique
go
CREATE INDEX idx_consulta_paciente
ON consulta (id_paciente) -- Nonunique.
go
```

InMemory Tables

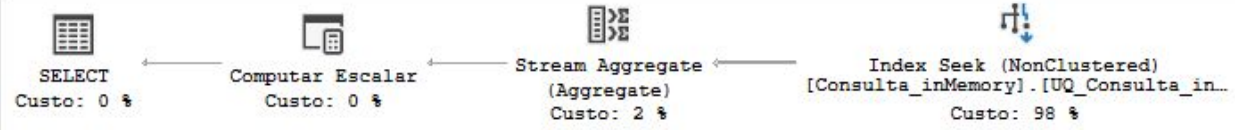
Comparação por um índice Unique :

```
180 select COUNT(*) from Consulta_inMemory
181     where datahora = @min_inmemory
182 select COUNT(*) from Consulta
183     where datahora = @min_normal
184
```

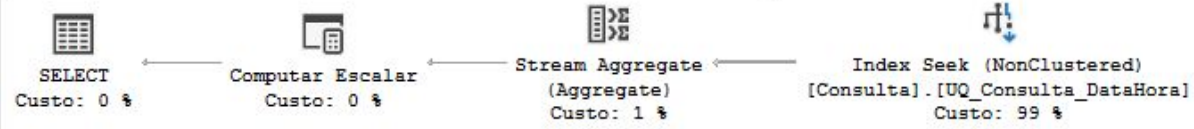
163 %

Mensagens | Plano de execução

Consulta 3: Custo da consulta (relativo ao lote): 10%
select COUNT(*) from Consulta_inMemory where datahora = @min_inmemory



Consulta 4: Custo da consulta (relativo ao lote): 32%
select COUNT(*) from Consulta where datahora = @min_normal



Quais suas conclusões ?

InMemory Tables

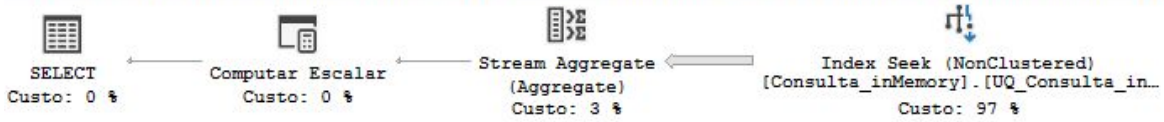
Comparação por um índice Unique :

```
181 select COUNT(*) from Consulta_inMemory
182     where datahora between @min_inmemory and @max_inMemory
183 select COUNT(*) from Consulta
184     where datahora between @min_normal and @max_normal
185
186
```

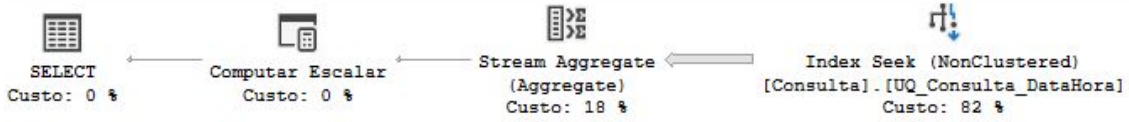
63 %

Mensagens | Plano de execução

```
Consulta 3: Custo da consulta (relativo ao lote): 85%
select COUNT(*) from Consulta_inMemory where datahora between @min_inmemory and @max_inMemory
```




```
Consulta 4: Custo da consulta (relativo ao lote): 15%
select COUNT(*) from Consulta where datahora between @min_normal and @max_normal
```



Quais suas conclusões ?

InMemory Tables

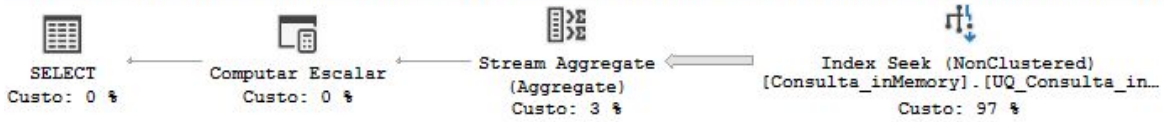
Comparação por um índice Unique :

```
181 select COUNT(*) from Consulta_inMemory
182     where datahora between @min_inmemory and @max_inMemory
183 select COUNT(*) from Consulta
184     where datahora between @min_normal and @max_normal
185
186
```

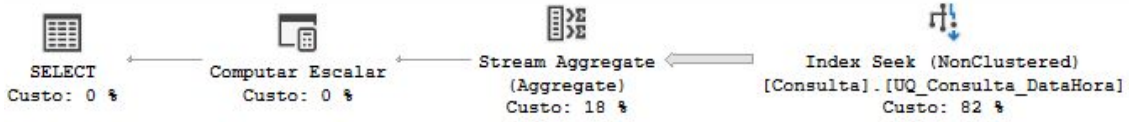
63 %

Mensagens | Plano de execução

```
Consulta 3: Custo da consulta (relativo ao lote): 85%
select COUNT(*) from Consulta_inMemory where datahora between @min_inmemory and @max_inMemory
```




```
Consulta 4: Custo da consulta (relativo ao lote): 15%
select COUNT(*) from Consulta where datahora between @min_normal and @max_normal
```



Quais suas conclusões ?

InMemory Tables

Verificando alocação da memória dos objetos:

```
--Verificando alocação em memória dos objetos
SELECT Object_name(t.object_id) AS [Table Name],
       memory_allocated_for_table_kb,
       memory_allocated_for_indexes_kb
FROM   sys.dm_db_xtp_table_memory_stats dms
       JOIN sys.tables t
         ON dms.object_id = t.object_id
WHERE  t.type = 'U'
```

	Table Name	memory_allocated_for_table_kb	memory_allocated_for_indexes_kb
1	Consulta_inMemory	85440	196800

Change Data Capture

É um processo por meio do qual é possível habilitar que as alterações de colunas de tabelas sejam rastreadas e armazenadas em uma tabela auxiliar.
O rastreamento das alterações é feito por meio da captura dos logs de alteração na tabela principal.

É possível habilitar no máximo dois CDC's para cada tabela e a persistência dos dados da tabela auxiliar é, por padrão, 3 dias.

O Server Agent deve estar habilitado pois ao habilitar o CDC para uma determinada tabela são criados dois jobs: um para efetuar a captura dos logs e outro para o processo de cleanup.

Muito utilizado por aplicativos que capturam tais mudanças para sincronização com outras bases ou serviços.

Mais detalhes:
Sobre o Change Data Capture - SQL Server | Microsoft Docs

Change Data Capture

Demonstração:

```
--habilitando o CDC no banco de dados
EXEC sys.Sp_cdc_enable_db

go

--Habilitando o trace sobre uma tabela
EXEC sys.Sp_cdc_enable_table
  @source_schema = N'dbo',
  @source_name = N'consulta',
  @role_name = NULL,
  @filegroup_name = 'primary'

go
```

Change Data Capture

Conferindo:

```
--Mudando algo
INSERT INTO consulta(id_paciente,id_medico,numero_sala,datahora,
            duracao)
SELECT ...
go
UPDATE consulta SET numero_sala = ...
WHERE  id = 1
go
DELETE TOP(1) FROM consulta
WHERE  id = 2
go
--Conferindo as entradas
SELECT * FROM   cdc.dbo_consulta_ct
```

Resultados | Mensagens

	__$start_lsn	__$end_lsn	__$seqval	__$operation	__$update_mask	ID	ID_Paciente	ID_Medico	Numero_Sala	DataHora	Duracao	__$command_id
1	0x000000770000B8BA0031	NULL	0x000000770000B8BA0004	2	0x3F	1000001	420	1	759	2019-09-08 13:45:00.000	30	1
2	0x000000770000B8D3000F	NULL	0x000000770000B8D30002	3	0x08	1	336	1	543	2020-12-07 11:00:00.000	15	1
3	0x000000770000B8D3000F	NULL	0x000000770000B8D30002	4	0x08	1	336	1	907	2020-12-07 11:00:00.000	15	1
4	0x000000770000B8D6000F	NULL	0x000000770000B8D6000D	1	0x3F	2	1	5	399	2019-05-13 15:15:00.000	15	1

Colunas da tabela que armazena as alterações:

- _$start_lsn: armazena o número sequencial de 'commit' dos logs das transações.
- _$seqval: armazena o número para ordenar as alterações dentro de uma mesma transação.
- _$operation: 1 = delete, 2 = insert, 3 = valor antes do update, 4 = valor após o update.

Change Data Capture

Outros comandos:

```
--limpeza dos jobs
EXEC sys.sp_mscdc_cleanup_job
-- Lista tabelas com CDC habilidatas
EXEC sys.Sp_cdc_help_change_data_capture
```

Desabilitando:

```
--Desabilitando o trace
EXECUTE sys.Sp_cdc_disable_table
  @source_schema = N'dbo',
  @source_name = N'consulta',
  @capture_instance = N'dbo_consulta';
go

--Desabilitando o CDC
EXEC sys.Sp_cdc_disable_dbgo
```

Extended Stored Procedures

No Common Language Runtime (CLR), os procedimentos armazenados são implementados como métodos estáticos públicos em uma classe Microsoft .NET Framework em um assembly depois importados no SQL Server como procedimentos estendidos.

```
Imports System
Imports System.Data
Imports System.Data.Sql
Imports System.Data.SqlTypes
Imports Microsoft.SqlServer.Server
Imports System.Data.SqlClient
Imports System.Runtime.InteropServices

'The Partial modifier is only required on one class definition per project.
Partial Public Class StoredProcedures
    ''' <summary>
    ''' Executes a query and iterates over the results to perform a summation.
    ''' </summary>
    <Microsoft.SqlServer.Server.SqlProcedure> _
    Public Shared Sub PriceSum( <Out()> ByRef value As SqlInt32)

        Using connection As New SqlConnection("context connection=true")
           value = 0
           Connection.Open()
           Dim command As New SqlCommand("SELECT Price FROM Products", connection)
           Dim reader As SqlDataReader
           reader = command.ExecuteReader()

           Using reader
              While reader.Read()
                 value += reader.GetSqlInt32(0)
              End While
           End Using
        End Using
    End Sub
End Class
```

SQL

```
CREATE PROCEDURE PriceSum (@sum int OUTPUT)
AS EXTERNAL NAME TestStoredProc.StoredProcedures.PriceSum
-- if StoredProcedures class was inside a namespace, called MyNS,
-- you would use:
-- AS EXTERNAL NAME TestStoredProc.[MyNS.StoredProcedures].PriceSum
```

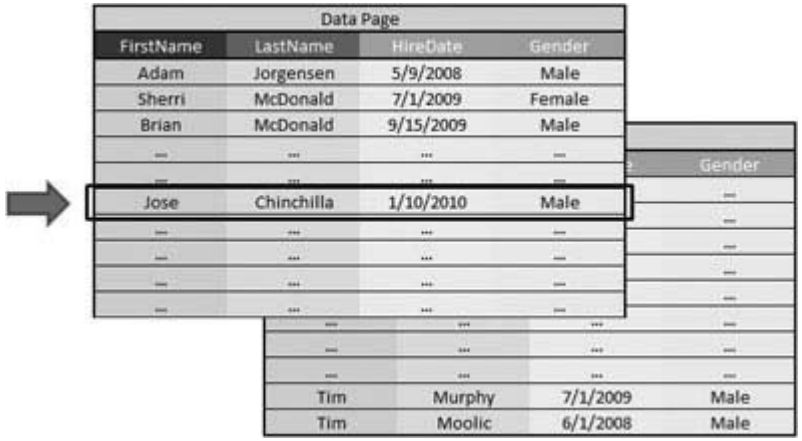
Mais detalhes:

Introdução à Integração do SQL Server CLR - ADO.NET | Microsoft Docs

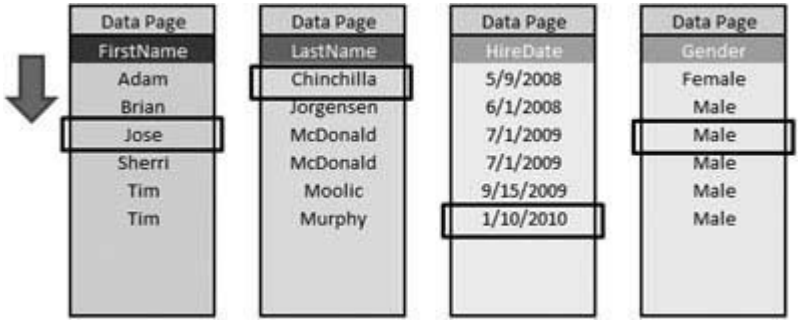
Visão geral do CLR (Common Language Runtime) - SQL Server | Microsoft Docs

Column Store Index

Tabelas e índices normais são armazenados na estrutura que chamamos de ROWSTORE ou seja, as páginas são preenchidas linha a linha



Column store permite a organização destas páginas em colunas, por isso o nome COLUMNSTORE.



Fonte das imagens: Understanding the SQL Server Columnstore Index (logicalread.com)

Mais detalhes:

Columnstore indexes: Overview - SQL Server | Microsoft Docs

CREATE COLUMNSTORE INDEX (Transact-SQL) - SQL Server | Microsoft Docs

Column Store Index

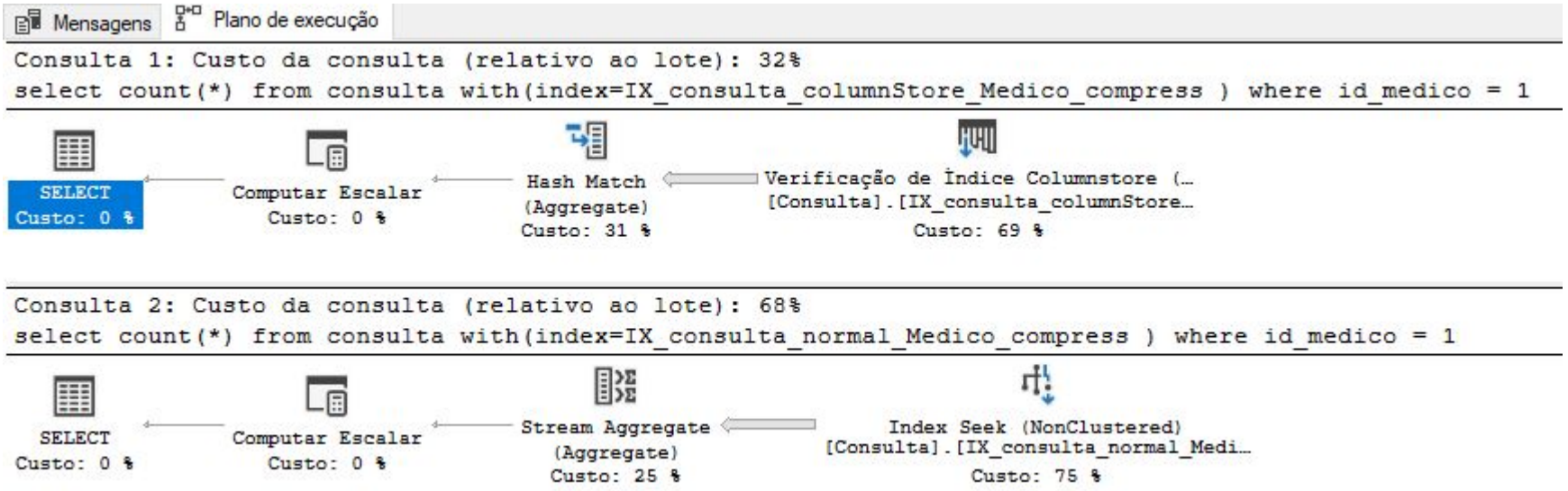
Demonstração:

```
-- Indices clusterizados deixam toda a tabela em modo COLUNAR.
-- não vamos rodar em nosso exemplo.
CREATE CLUSTERED columnstore INDEX ix_consulta_columnstore ON consulta;

--Indices não clusterizados funcionam como indices não clusterizados comuns
-- Uma das grandes vantagens de utilização de columnStore é a compressão das páginas.
CREATE NONCLUSTERED columnstore INDEX ix_consulta_columnstore_medico_compress
     ON consulta ( id_medico ) WITH ( data_compression = columnstore );

--Criando o indice normal equivalente
CREATE NONCLUSTERED INDEX ix_consulta_normal_medico_compress
     ON consulta( id_medico ) WITH( data_compression = page);--OU ROW
```

Column Store Index

Comparação:

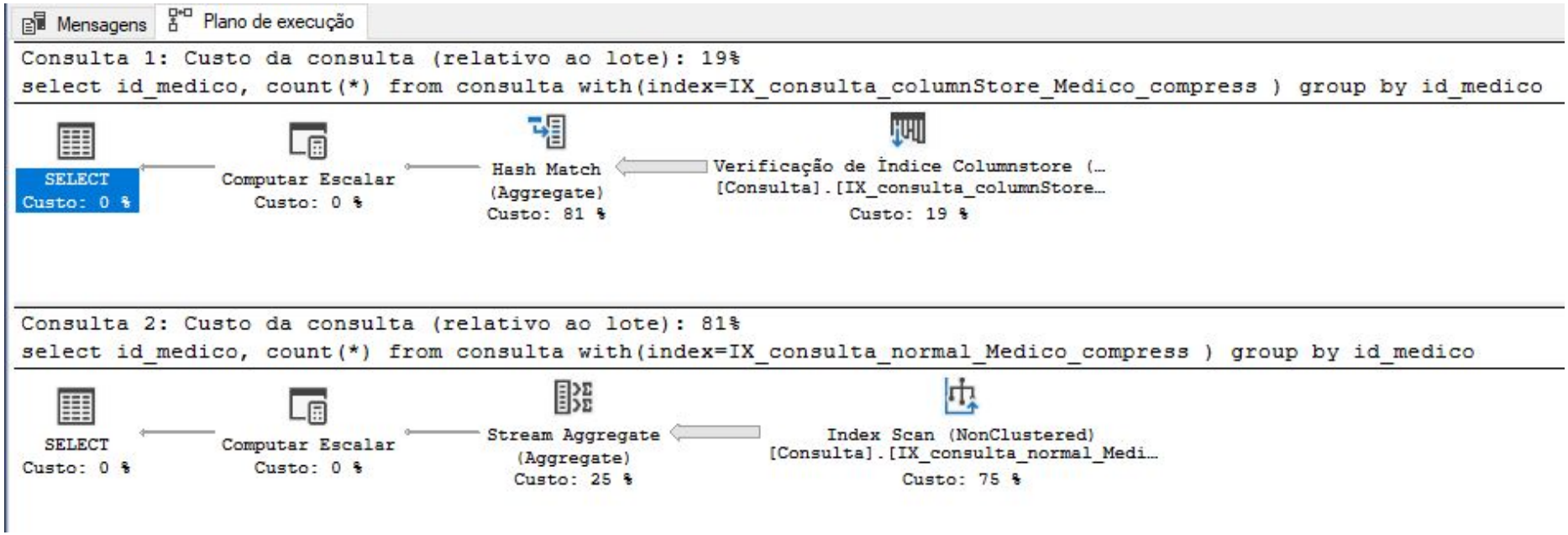
```
SELECT Count(*)
FROM   consulta WITH(INDEX=ix_consulta_columnstore_medico_compress )
WHERE  id_medico = 1

SELECT Count(*)
FROM   consulta WITH(INDEX=ix_consulta_normal_medico_compress )
WHERE  id_medico = 1
```



Column Store Index

Comparação com group by :

```
SELECT id_medico,
       Count(*)
FROM   consulta WITH(INDEX=ix_consulta_columnstore_medico_compress )
GROUP  BY id_medico

SELECT id_medico,
       Count(*)
FROM   consulta WITH(INDEX=ix_consulta_normal_medico_compress )
GROUP  BY id_medico
```



Obrigado !

Gustavo Maia
Gustavo.Maia@FaculdadeImpacta.com.br